Internal Report 2010–08 August 2013

Universiteit Leiden
Opleiding Informatica

On Derandomization

in

Evolution Strategies

Hao Wang

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The derandomized techniques are frequently applied to improve the performance of stochastic optimization algorithms. The evolution strategy (ES), a subfield of evolutionary algorithm, is one of stochastic optimization algorithms. In the ES history, the derandomization of the ES parameter tuning has already been proved to give a big performance leap. Consequently, the state-of-art ES algorithm has "evolved" from the mutative-self-adaption framework (MSC-ES) to the covariance-matrix-adaptation technique (CMA-ES). Recently, the derandomization techniques are applied to the mutation operator to improve the quality of mutation samples and reduce sampling errors. Following the recent trend, this thesis firstly discusses and analyzes the newly derandomized sampling method, mirrored sampling in detail. Then the drawbacks of mirrored sampling are explained. A possible improvement, "noisy" mirrored sampling is proposed. Secondly, the idea of mirrored sampling is generalized into two concepts, the direction derandomization and the step-size derandomization. Two new mutation operators are proposed based on these two concepts. Finally, those newly proposed mutation operators are implemented into ES algorithms and tested on BBOB-2012 benchmark to investigate and verify their abilities.

Contents

1 Introduction **1**

2 Background **3**
2.1 A hierarchical view of evolution strategy . . . 3
2.2 Review of (μ, λ)-MSC-ES . . . 5
2.2.1 Implicit covariance matrix . . . 5
2.2.2 The algorithm . . . 6
2.2.3 Drawbacks . . . 7
2.2.4 Biasness of the recombination . . . 8
2.2.5 Cumulative parameter adaptation . . . 11
2.3 From (μ, λ)-MSC-ES to $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES . . . 12
2.3.1 Covariance matrix adaptation . . . 13
2.3.2 Step-size adaptation . . . 14

3 Mirrored sampling technique **17**
3.1 Intuition and the algorithm . . . 17
3.2 Theoretical aspects of $(1, \lambda_{\mathrm{m}})$-ES . . . 21
3.2.1 The properties of the mutation vector . . . 22
3.2.2 The $(1, \lambda_{\mathrm{m}})$ evolution strategy . . . 23
3.3 Mirrored sampling and recombination . . . 27
3.4 The "noisy" mirrored sampling technique . . . 28

4 Derandomized sampling **31**
4.1 Direction derandomization . . . 31
4.1.1 A measure on mutation direction . . . 32
4.1.2 A completely derandomized direction approach . . . 33
4.1.3 Orthogonal Sampling . . . 35
4.2 Step-size derandomization . . . 37
4.2.1 Motivation . . . 37
4.2.2 The algorithm . . . 38
4.2.3 Limitation . . . 39
4.2.4 Application . . . 39
4.3 Theoretical results on step-size derandomization . . . 40
4.3.1 Prerequisite on probability distribution . . . 41
4.3.2 $(1 + 1)$-ES on Linear Model . . . 42

4.3.3 $(1+1)$-ES on Sphere Model 43

5 Empirical results **45**
5.1 Performance measure 45
5.2 Experiment details 46
5.2.1 BBOB features 46
5.2.2 Experiment objects 46
5.2.3 Test functions 47
5.2.4 Parameter Setting and Implementation Issue 48
5.3 Explanation on the BBOB figures 49
5.4 Results on BBOB-2012 benchmark 52
5.4.1 Cu-Simple-(μ, λ)-MSC-ES 52
5.4.2 Noisy$-(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES 56
5.4.3 Derandomized-stepsize-(μ, λ)-CMSA-ES 60
5.4.4 Cu-Simple-(μ, λ)-MSC-ES vs Simple-(μ, λ)-MSC-ES 64
5.4.5 Noisy$-(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES vs $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES 70
5.4.6 Orthogonal1 $-$ $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES vs $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES 76
5.4.7 Orthogonal2 $-$ $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES vs $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES 82
5.4.8 BBOB result of derandomized step-size 88

6 Conclusion **94**

List of Figures

2.1 The hierarchical structure of evolution strategy algorithm. 4
2.2 Step-size σ versus number of function evaluations of 20 runs on a purely random fitness function in dimension 10. x-axis: evaluation counts, y-axis: global step-size. 10

3.1 Sampling process of mirrored sampling. 19
3.2 The improvement after applying mirrored sampling. 20
3.3 Projection of n-dimensional sphere landscape onto the two-dimensional plane after the rotation. 23
3.4 Comparison of the progress coefficients. 26
3.5 Step-size σ versus number of function evaluations of 30 runs on a purely random fitness function in dimension 10. x-axis: evaluation counts. 30

4.1 Step-size σ versus number of function evaluations of 30 runs on a purely random fitness function in dimensions $2, 3, 5, 10, 20, 40$. x-axis: evaluations. . . 33
4.2 Convergence velocity comparison of $(1 + 1)$-ES on the sphere model. 44

5.1 An Example of running time figures. 49
5.2 An Example of Empirical cumulative distribution function figures. 50
5.3 An Example of ERT ratio figures. 51
5.4 Expected number of f-evaluations to reach $f_{\mathrm{opt}} + \Delta f$ for Cu-Simple-(μ, λ)-MSC-ES 53
5.5 Empirical cumulative distribution functions (ECDFs) for Cu-Simple-(μ, λ)-MSC-ES 54
5.6 Expected number of f-evaluations to reach $f_{\mathrm{opt}} + \Delta f$ for $(\mu/\mu_{\mathrm{w}}, \tilde{\lambda}_{\mathrm{m}})$-CMA-ES 57
5.7 Empirical cumulative distribution functions (ECDFs) for $(\mu/\mu_{\mathrm{w}}, \tilde{\lambda}_{\mathrm{m}})$-CMA-ES 58
5.8 Expected number of f-evaluations to reach $f_{\mathrm{opt}} + \Delta f$ for Derandomized-stepsize CMSA 61
5.9 Empirical cumulative distribution functions (ECDFs) for Derandomized-stepsize CMSA 62
5.10 Expected running time divided by dimension versus dimension. 65
5.11 Ratio of ERT for Cu-Simple-(μ, λ)-MSC-ES over ERT for Simple-(μ, λ)-MSC-ES versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇ 66
5.12 Expected running time of Simple-(μ, λ)-MSC-ES versus Cu-Simple-(μ, λ)-MSC-ES. 67
5.13 Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios. 68

5.14 Expected running time divided by dimension versus dimension. 71
5.15 Ratio of ERT for "Noisy" mirrored sampling over ERT for Mirrored sampling versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇ 72
5.16 Expected running time of Mirrored sampling versus "Noisy" mirrored sampling. 73
5.17 Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios. 74
5.18 Expected running time divided by dimension versus dimension. 77
5.19 Ratio of ERT for Orthogonal-Sampling1 over ERT for CMA-ES versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇ 78
5.20 Expected running time of CMA-ES versus Orthogonal-Sampling1. 79
5.21 Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios. 80
5.22 Expected running time divided by dimension versus dimension. 83
5.23 Ratio of ERT for Orthogonal-Sampling2 over ERT for CMA-ES versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇ 84
5.24 Expected running time of CMA-ES versus Orthogonal-Sampling2. 85
5.25 Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios. 86
5.26 Expected running time divided by dimension versus dimension. 89
5.27 Ratio of ERT for Derandomized-stepsize CMSA over ERT for CMSA versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇ 90
5.28 Expected running time of CMSA versus Derandomized-stepsize CMSA. . . . 91
5.29 Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios. 92

List of Algorithms

1.1 Outline of an evolutionary algorithm 1
2.1 (μ, λ)-MSC-ES 6
2.2 SIMPLE-(μ, λ)-MSC-ES 11
2.3 Cu-SIMPLE-(μ, λ)-MSC-ES 12
3.1 SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$ 18
3.2 MIRRORED-SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$ 18
3.3 NOISE-GENERATION(N) 29
3.4 NOISY-MIRRORED-SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$ 29
4.1 COMPLETE-DERANDOMIZED-DIRECTION-SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$ 34
4.2 ORTHOGONAL-SAMPLING1$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$ 36
4.3 GRAM-SCHMIDT$(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ 36
4.4 ORTHOGONAL-SAMPLING2$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$ 37
4.5 DERANDOMIZED-STEPSIZE-SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$ 39
4.6 (μ, λ)-CMSA-ES 40

Chapter 1

Introduction

Evolution strategy (ESs) is an optimization technique based on ideas of adaptation and evolution. It belongs to the general class of evolutionary algorithms (EAs). A very primitive EA algorithm structure is shown in Algorithm 1.1. The evolution strategy optimization technique was created in the early 1960s and developed further in the 1970s by Ingo Rechenberg, Hans-Paul Schwefel and their co-workers.

Evolution Strategies can be applied in all fields of optimization including continuous, discrete, combinatorial search spaces and even mixed search spaces without/with constraints. It directly uses the solution vectors of the optimization problem as representation of itself. Normally various solution vectors (offspring) are created by the mutation operator. Then their corresponding function values (fitnesses) are calculated by the object function. The selection operator are used to select some good offspring based on their fitness and the selected offspring are further recombined in order to generate offspring again. In common with evolutionary algorithms, the operators are applied in a loop. An iteration of the loop is called a *generation*. The sequence of generations is continued until a termination criterion is met. Although there is a huge diversity in this field, most of the ES algorithms share the same characteristics: they are using a mutation operator based on multivariate normal distribution and their functionalities are largely dependent on and affected by a upper level adaptation mechanism of the covariance matrix of the the normal distribution. The latter is called *self-adaptation*.

Algorithm 1.1 Outline of an evolutionary algorithm

```
1 Initialization
2 repeat
3    Recombination
4    Mutation
5    Evaluation
6    Selection
7 until termination criterion fulfilled.
```

As a special class of computer algorithms, there are some criteria to measure the performance of ESs. The most common one is the time complexity of the algorithm. This is the standard criterion to measure the performance of a computer algorithm and also appli-

cable for ESs. The computational overheads are mainly due to the numerical computation of the covariance matrix, possible time-consuming adaptation mechanism and maybe sophisticated mutation operators. However, the time complexity or the space complexity (the memory space consumed by ESs) is not the most important criterion for ESs due to the increasing computational power. Instead, the evaluations in a ES algorithm are very precious. The fitness function itself could be so complicated that it would take a while for one evaluation. Or, it could be a computational program which takes the solution vectors as its parameter and gives a real number as fitness value after a long run (Consider an ES algorithm is used to optimize the structure of a neural network). In the worst case, the fitness value could only be retrieved from a real-world experiment, like the optimization of shape of a car or a quantum control experiment. In such case, one evaluation corresponds to one real experiment, which does not only consume our time. Loosely speaking, the prominent criterion of ES performance is the expected number of evaluations it takes to give a good solution vector. The formal terminology is the convergence velocity, which is of importance in our analysis.

As the result of continuous effort towards improving the ES algorithms, a lot of well-known ES algorithms are developed, tested, analyzed and exploited in real optimization problems. The very first successful one is called the $(1+1)$-ES with 1/5-success rule, which is developed by Schwefel [36] and Rechenburg [35]. It is just a single point (one offspring) search algorithm using an external rule to tune the global step-size. Later, the significances of multi-offspring and parameter self-adaptation are discovered and the first ES able to adapt all the endogenous strategy parameters, (μ, λ)-MSC-ES was proposed by Schwefel [37] in 1981. It is a successful algorithm based on the elegant intuition: mutative self-adaptation of parameters, which is discussed in details in Section 2.2. Despite the success of MSC-ES, there are some unpleasant randomnesses which are first discussed and removed by Ostermeier et. al. [31] in 1993, resulting in the very first *derandomized* ES algorithm, DR1. The derandomization of ES algorithm, which means discovering the unpleasant or useless randomnesses and replacing them by much more deterministic logic, targets at improving the ES performance (saving evaluations) efficiently and enormously. Following such goal, the derandomized ES has also "evolved" to DR2 [32], then DR3 [26] and finally CMA-ES [25]. They have achieved great successes both experimentally and theoretically. Due to the importance of derandomization in ES, this thesis would be devoted to re-visiting the harmful randomnesses in history and their derandomized solutions, discussing and generalizing the recently development and trend in derandomization and to give some possible improvements following the trend. For more about the history of classic evolution strategies, the survey by Bäck [8] or the comprehensive introduction by Beyer and Schwefel [12] should meet you favor.

The structure of this thesis is as follows: Chapter 2 gives a detailed introduction and discussion about the knowledge basis and summarizes the essence of the CMA-ES algorithm. In Chapter 3, a new derandomized technique named *mirrored sampling* is introduced and analyzed. We give its advantages as well as its drawbacks. A possible improvement is also shown. In Chapter 4, the idea introduced by mirrored sampling is then generalized, resulting in two new ES variants. The motivation and possible analysis are performed. Finally, in Chapter 5, all the newly proposed ES variants are tested on BBOB-2012 benchmark. The detailed results are provided.

Chapter 2

Background

In this chapter, the background knowledge of the discussions in Chapter 3, 4 are introduced in detail. First, a hierarchical view of the evolution strategy is given as the basis of this chapter. Second, (μ, λ)-MSC-ES, which is the first ES able to adapt the covariance matrix successfully, is introduced and discussed in depth. In addition, a problem about the recombination in MSC-ES is proposed and a possible improvement is given to relieve the problem. Finally, the leap from MSC-ES to the completely derandomized ES, CMA-ES, is introduced in a comparable manner. Another purpose of this chapter is to summarize the history of the derandmomization in evolution strategy and to explain the functionalities of the derandomized techniques.

2.1 A hierarchical view of evolution strategy

Since the development of the derandomized evolution strategy algorithms (DR1, DR2, DR3, CMA-ES), a lot of the derandomization techniques have been proposed to improve the algorithm performance. Because of the diversity of those approaches, it is not a easy job to clearly argue which parts of randomness in ES is derandomized and why such randomness is undesirable. Therefore, it is crucial to begin with a ES framework which could characterize and categorize the different derandomization techniques.

The author argues that most of evolution strategies could be presented in a hierarchical way such that the ES algorithm is divided into two levels interacting with each others. These two levels could be formalized such that their interfaces (inputs and outputs) are standardized. In this way, different ES algorithm could be organized in a uniform structures. Such framework is shown in Figure 2.1.

Each ES algorithm can be organized in two levels, shown by level 1 and Level 2 in the figure. Prior to level 1, the level 0 stands for the fitness function. It just evaluates the vectors given by level 1 and could be even a back box. Level 1 is the body of ES and in essence is a population-based stochastic optimization using mutation, selection and recombination operators. The stochastic search is then parametric and controlled by a parameter set $\mathbf{\Psi}$. It is also a functional block which can work alone with predefined $\mathbf{\Psi}$. Level 2 is managing to tune the parameter set $\mathbf{\Psi}$. Its only information source is the ordered offspring $\{\mathbf{x}_{i:\lambda}\}_{i=1}^{\lambda}$. Then some methods are exploited to offer Level 1 a reasonable prediction $\hat{\mathbf{\Psi}}$ for the next

generation. The interactions between these three levels is also shown in Fig. 2.1.

The author also argues that a "good" ES algorithm should resemble this framework in the structure. The functionality of every component is clear and dependences between different components are reduced as much as possible so that the alternative of a component could be put in without too much violation of the overall algorithm. Due to such advantage, if we view the known algorithms in this hierarchical structure, it is easier to compare the a component in one algorithm to its counterpart in another algorithm. The following introduction and discussion will base on this hierarchical structure.

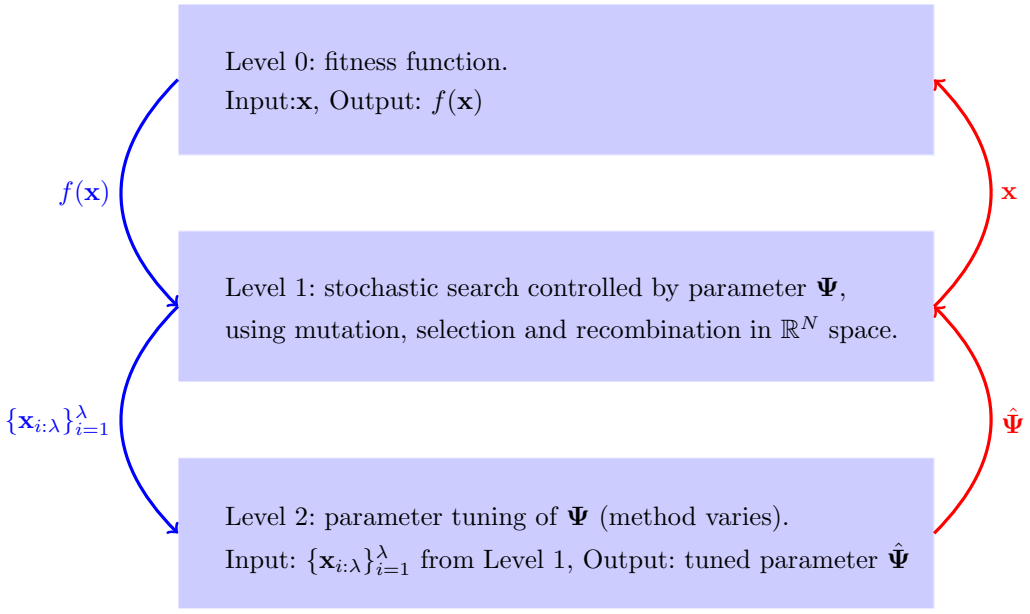

Figure 2.1: The hierarchical structure of evolution strategy algorithm.

2.2 Review of (μ, λ)-MSC-ES

2.2.1 Implicit covariance matrix

The (μ, λ)-MSC-ES (MSC is the abbreviation of mutative self-adaptation of covariances) is the first evolution strategy that successfully adapts the covariance matrix and is widely used. The algorithm is developed by Schwefel in [37]. The motivation is quite natural: if we do not know the rules to adapt the covariance matrix, we could simply treat it in the same way as the solution vector and let it evolve. This is named *self-adaptation*. In order to mutate a covariance matrix, we should manipulate it in an *implicit* form. Given the eigen-decomposition of the covariance matrix $\mathbf{C}$,

$$\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^T, \quad \mathbf{C}^{\frac{1}{2}} = \mathbf{B}\mathbf{D}\mathbf{B}^T$$

where

- $\mathbf{B}$ is an orthogonal matrix containing eigenvectors, $\mathbf{B}\mathbf{B}^T = \mathbf{I}$,
- $\mathbf{D}^2 = diag(d_1^2, \ldots, d_N^2) = diag(d_1, \ldots, d_N)^2$ is a diagonal matrix and $d_1^2, \ldots, d_N^2$ are N positive eigenvalues of C.

Then the normal mutation $\boldsymbol{\mathcal{N}}(\mathbf{0}, \sigma^2\mathbf{C})$ can be written in different ways,

$$\begin{aligned}\boldsymbol{\mathcal{N}}(\mathbf{0}, \sigma^2\mathbf{C}) &\sim \sigma\mathbf{C}^{\frac{1}{2}}\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}) \\ &\sim \sigma\mathbf{B}\mathbf{D}\mathbf{B}^T\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}) \\ &\sim \sigma\mathbf{B}\mathbf{D}\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}).\end{aligned}$$

Note that the last equivalence above is because $\mathbf{B}^T$ represents a rotation matrix while $\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$ is rotation-invariant (see Section 3.2.1 for detail). Because $\mathbf{D}$ is a diagonal matrix, the multiplication $\mathbf{D}\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$ is just multiplying the square root of the N eigenvalues, $d_1, \ldots, d_N$ to each component of the mutation vector. The effect is rescaling the mutation vector in each dimension, which is realized by the so-called individual step-size $\sigma_1, \ldots, \sigma_N$ in MSC-ES. Note that the global step-size σ above is also realized by individual step-size $\sigma_1, \ldots, \sigma_N$.

Then due to the orthogonality of $\mathbf{B}$, it represents a rotation matrix[1]. A rotation operation in high-dimensional space can be determined by $N(N-1)/2$ rotation angles which correspond to $N(N-1)/2$ 2-dimensional subspaces and dictate the rotation component at each 2-dimensional subspace. The rotation matrix R_{ij} in 2-dimensional subspace formed by axis i and j is given by an identity matrix, extended by the entries $R(i,i) = R(j,j) = \cos\alpha_{ij}$ and $R(i,j) = -R(j,i) = -\sin\alpha_{ij}$ (α_{ij} is the rotation angle in this subspace). Furthermore, the overall rotation matrix in the search space is computed as the product of all the R_{ij},

$$\mathbf{B} = \prod_{i=1}^{n-1}\prod_{j=i+1}^{n} R_{ij}. \tag{2.1}$$

Given the relation of the eigenvector B and the rotation angles above, every covariance matrix can be decomposed into two quantities: rotation angles and the square root of eigenvalues, which are manipulated much easier in ES. They are further treated the same as the solution vector and participate in the evolution process.

[1]Normally an orthogonal matrix $\mathbf{B}$ represents a rotation matrix when $|\mathbf{B}| = 1$. When $|\mathbf{B}| = -1$, it represents a reflection transformation, which also holds for our discussion here.

2.2.2 The algorithm

The (μ, λ)-MSC-ES introduced here is basically the same as the one proposed in [37]. However, only intermediate recombination operator is allowed here while some alternatives are possible in the original literature. According to the self-adaptation principle, the individual step-sizes and rotation angles also evolve every generation. If the individual step-sizes are denoted by a vector $\boldsymbol{\sigma}$ of length N, the mutation rules for it is defined as

$$\boldsymbol{\sigma}_i \leftarrow \bar{\boldsymbol{\sigma}} \exp\left(\tau' \mathcal{N}(0,1) + \tau \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})\right), \quad 1 \leq i \leq \lambda \tag{2.2}$$

where $\bar{\boldsymbol{\sigma}}$ is the mean individual step-size vector obtained in the last generation. τ and τ' are two constants controlling the strength of mutation. Note that this mutation rule is the product of two log-normal distributions. The first one is a scalar $\exp(\tau' \mathcal{N}(0,1))$ which mutates all the dimensions at the same amount and is called *global step-size*. Another one is a random vector $\exp(\tau \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}))$ mutating each dimension differently and in fact adapting the eigenvalues of the covariance matrix. The rotation angles are also packed into a vector $\boldsymbol{\alpha}$ of length $N(N-1)/2$. The mutation rule for it reads basically the same as that for the solution vector,

$$\boldsymbol{\alpha}_i \leftarrow \bar{\boldsymbol{\alpha}} + \beta \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}), \quad 1 \leq i \leq \lambda \tag{2.3}$$

where $\bar{\boldsymbol{\alpha}}$ is the mean rotation angle vector obtained in the last generation. Constant β is the standard deviation of this mutation. τ, τ' and β arre called learning rates. As recommended by Schwefel [37], the setting $\tau = \frac{1}{\sqrt{2\sqrt{N}}}, \tau' = \frac{1}{2\sqrt{N}}$ and $\beta = \frac{5}{180}\pi$ are reliable.

Algorithm 2.1 (μ, λ)-MSC-ES

```
Initialize population
P^(0) ← {(x_i, σ_i, α_i)}_{i=1}^λ
t ← 0
repeat
    t ← t + 1
    x̄ = (1/μ) Σ_{i=1}^μ x_i
    σ̄ = (1/μ) Σ_{i=1}^μ σ_i
    ᾱ = (1/μ) Σ_{i=1}^μ α_i
    for i = 1 → λ do
        η ← τ' N(0,1)
        σ_i ← σ̄ exp(η + τ N(0,I))
        α_i ← ᾱ + β N(0,I)
        B ← Π_{i=1}^{n-1} Π_{j=i+1}^{n} R_ij
        x_i ← x̄ + B · σ_i ⊕ N(0,I)
        φ_i ← f(x_i)
    end for
    Select μ best from (x_i, σ_i, α_i)
until termination criterion fulfilled.
```

In the next step, the rotation angles $\boldsymbol{\alpha}$ is used to calculate component $\mathbf{B}$ of the covariance matrix as shown in Equation 2.1. In addition, $\boldsymbol{\sigma}$ represents the square root of the eigenvalues.

Therefore, offspring are generated as

$$\mathbf{x}_i \leftarrow \bar{\mathbf{x}} + \mathbf{B} \cdot \boldsymbol{\sigma}_i \oplus \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}), \quad 1 \leq i \leq \lambda$$

Note that symbol $\oplus$ means the element-wise multiplication and $\bar{\mathbf{x}}$ is the parental vector. In the selection step, the goodness of $\boldsymbol{\alpha}_i$ and $\boldsymbol{\sigma}_i$ are determined by the fitness of $\mathbf{x}_i$. Then the pair $(\mathbf{x}_i, \boldsymbol{\alpha}_i, \boldsymbol{\sigma}_i)$ are ranked with respect to its goodness. The intermediate recombination is uniformly used for the μ best pairs, yielding the parental pair $(\bar{\mathbf{x}}, \bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\sigma}})$ of the next generation. The complete algorithm is listed in Algorithm 2.1 and should be self-explanatory.

2.2.3 Drawbacks

Although the concept of making strategy parameters mutate and survive from the environment seems a good way to adjust the strategy parameters, such approach has lots of drawbacks due to the disturbed and limited information source of the strategy parameter adaptation and an intrinsic conflict involving the mutation strength setting of parameters.

There are two evolution processes at the different levels. The mutation, selection and recombination of solution vectors is the evolution dynamics in problem space, namely level 1 in Figure 2.1. We term it as *the first level evolution process* here. This is the standard stochastic search process parameterized by strategy parameters, $(\boldsymbol{\sigma}_i, \boldsymbol{\alpha}_i)$. To control and optimize the parameters, MSC-ES facilitates the second level evolution dynamics in the parameter space (level 2 in Figure 2.1). We term it as *the second level evolution process* here. The main drawbacks are due to this level 2 evolution process, most of which are argued below:

1. **The reliability of adaptation information**. Any adaptation mechanism works based on some information source. For the first level evolution process, the information is merely the fitness values of the population, which is precise in the noiseless fitness case. However, in the second level, the real information needed is the goodness of a strategy parameter mutant, which is neither provided directly nor possibly to be inferred accurately by the first level. Instead, the goodness is determined by the fitness rank of the corresponding offspring. However, it is possible to assemble a good realized standard normal vector and a bad strategy parameter setting to make a good offspring, which implies that *a good offspring does not always imply a good parameter setting.* Therefore, the second level evolution process is working on a noisy landscape in fact.

2. **Indirect selection of strategy parameter setting**. Despite the noisy information fed to the second level, the mutative parameter control mechanism is also loosing information by its nature. A good mutation containing good strategy parameters $(\boldsymbol{\sigma}, \boldsymbol{\alpha})$ in the first level is selected not to directly adapt the strategy parameters. Instead, such information is used to adapt the parameters of the mutation distribution $(\bar{\boldsymbol{\sigma}}, \bar{\boldsymbol{\alpha}})$ in the second level so that the probability of realizing good strategy parameters $(\boldsymbol{\sigma}, \boldsymbol{\alpha})$ again is increased. In this manner, comparing two different strategy parameter settings, the better one has (only) a higher probability to be selected. Differences between these selection probabilities can be quite small.

3. **Changing landscape**. The optimal setting of strategy parameters depends on the current search location in the problem space. Fixing the search point in problem space,

the mapping from all possible strategy parameters to the performance of the first level search can be found at least theoretically, which leads to a strategy parameter landscape depending on the search point in the lower level. The second level is actually working on this landscape. As the evolution of the first level, the landscape is changing. Thus, the MSC approach has to optimize a dynamic landscape, which should be inefficient.

4. **Conflicts involving the mutation strength**. Mutation strength in the second level is controlled by the so-called learning rate and is usually kept constant throughout the search process. It is already really difficult to facilitate an effective mutation strength that is virtually independent of the actual position in strategy parameter space. In addition, the landscape of strategy parameters seems changing during the evolution of the first level, which makes the problem even harder. Another issue, the *strategy parameter change rate* defined as the difference between the strategy parameters of two consecutive generations, is an indicator of the adaptation speed. Moreover, the mutation strength in the second level that obtains optimal change rate is typically smaller than the one that obtains good diversity among the mutants, as a desired effect of the mutation operator. Such a conflict can not be solved by any tuned learning rates.

2.2.4 Biasness of the recombination

Despite the well-known unsatisfactory aspects listed above, there is another question with the recombination operator of the strategy parameters. The idea of MSC scheme seems biologically plausible and can be easily understood. However, we argue that it is very important to understand its underlying statistical properties. The statistical analysis of its mechanism can actually act as a proof to the biological intuition, may offer some clues of improvements and at least gives challenging questions which helps our understanding about itself.

The recombination of individual step-sizes is considered here. Due to Equation 2.2 the individual step-size vectors are generated as,

$$\boldsymbol{\sigma}_i^{(g+1)} \leftarrow \hat{\boldsymbol{\sigma}}^{(g)} \exp\left(\tau' \mathcal{N}(0,1) + \tau \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})\right), \quad 1 \leq i \leq \lambda$$

It is the sum of two *log-normal* distributions rescaled by $\hat{\boldsymbol{\sigma}}$. The superscript (g) is the generation index. Then individual step-sizes are ranked based on the fitness of their corresponding offspring. We will use the notation $\boldsymbol{\sigma}_{i:\lambda}^{(g)}$ to represent the i-th best parameter in such ranking. Finally the recombination takes place to guess a good parameter based on μ best individual step-size,

$$\boldsymbol{\sigma}^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} \boldsymbol{\sigma}_{i:\lambda}^{(g)}$$

This mutation-selection-recombination cycle above could be treated as a *statistics estimation procedure*, which manages to "guess" a optimal parameter setting from limited sources. In our case, we denoted the optimal individual step-size vector in generation g as $\boldsymbol{\sigma}^{*^{(g)}}$. Then the first step of the standard procedure to estimate $\boldsymbol{\sigma}^{*^{(g)}}$ is *sampling*, which generates a population distributed by a certain distribution having $\boldsymbol{\sigma}^{*^{(g)}}$ as its parameter (the log-normal distribution is used for our case). However, it is not possible to generate such population due to the unknown optimal $\boldsymbol{\sigma}^{*^{(g)}}$. Instead, a reasonable guess $\hat{\boldsymbol{\sigma}}^{(g)}$, which is obtained in

the last generation, is considered as a approximation of $\boldsymbol{\sigma}^{*^{(g)}}$ and used to generated the individual step-size population $\{\boldsymbol{\sigma}_i^{(g+1)}\}_{1\leq i\leq\lambda}$. For the standard estimation procedure, all the λ individuals are used in the estimation. However, the collection $\{\boldsymbol{\sigma}_i^{(g+1)}\}_{1\leq i\leq\lambda}$ is not directly related to $\boldsymbol{\sigma}^{*^{(g)}}$. We suppose that the samples that are highly related to the optimum should share similar behavior with the optimum, which is measured by their fitnesses. Then it is reasonable to use the best μ individual step-size vectors to make an approximate estimation. This is the essential functionality of selection operator in terms of estimation.

Given such procedure above, the rest task is to prescribe a "good" estimator. In the standard MSC, the most popular method to generate the new $\boldsymbol{\sigma}$ is the intermediate recombination, namely the sample mean, which is

$$\bar{\boldsymbol{\sigma}}^{(g+1)} = \frac{1}{\mu}\sum_{i=1}^{\mu}\boldsymbol{\sigma}_{i:\lambda}^{(g)}. \tag{2.4}$$

This estimation seems really natural and plausible because that is the same the way in which the solution vectors are recombined. However, recall that the sample of $\boldsymbol{\sigma}$ is log-normally distributed while the sample mean $\bar{\boldsymbol{\sigma}}$ is the *maximum likelihood estimator* for normal distribution instead of log normal. Furthermore, the maximum likelihood estimator of log-normal distribution for the mean is [27, Chapter 14]:

$$\boldsymbol{\sigma}_{\mathrm{MLE}}^{(g+1)} = \exp\left(\frac{1}{\mu}\sum_{i=1}^{\mu}\ln\boldsymbol{\sigma}_{i:\lambda}^{(g)}\right). \tag{2.5}$$

At this moment we would like to propose the question: which of these two estimators should we choose? The original $\bar{\boldsymbol{\sigma}}$ or $\boldsymbol{\sigma}_{\mathrm{MLE}}$. An important design criterion for stochastic search procedure (also a theoretical aspect for judging the goodness of estimator) is the *unbiasedness* of variations of strategy parameters. A parameter ϕ is considered unbiased in a given procedure if and only if its expected value remains unchanged in the next generation under random selection (e.g. the objective function $f(\mathbf{x}) = rand$ or $f(\mathbf{x}) = const$, to be independent of $\mathbf{x}$).

Under the random selection, all the samples are accepted for the recombination or estimation as argued before, which means all the samples are believed to be generated from a distribution involving $\boldsymbol{\sigma}^{*^{(g)}}$. Then the expected estimates should be $\boldsymbol{\sigma}^{*^{(g)}}$. For ES aspects, under random selection, there is no "gain" from the landscape teaching the algorithm how to evolve. The best option is to remain the parameter unchanged. From experience, it is usually the case that the logarithm of individual step-size should be unbiased. Thus, we analyze the biasness of these two estimators as below. Note that the expectations of log normal distributions we are facing are [27],

$$\mathrm{E}\left[\exp\left(\tau'\mathcal{N}(0,1)\right)\right] = \exp\left(\frac{\tau'^2}{2}\right)$$
$$\mathrm{E}\left[\exp\left(\tau\boldsymbol{\mathcal{N}}(\mathbf{0},\mathbf{I})\right)\right] = \exp\left(\frac{\tau^2}{2}\right)\mathbf{1}_{N\times 1}$$

where $\mathbf{1}_{N\times 1}$ is the a length N column vector with all its entries equal to 1. We first calculate the conditional expectation of $\ln \bar{\boldsymbol{\sigma}}$:

$$\begin{aligned}
\mathrm{E}\left[\ln \bar{\boldsymbol{\sigma}}^{(g+1)} \mid \bar{\boldsymbol{\sigma}}^{(g)}\right] &= \mathrm{E}\left[\ln\left(\frac{1}{\mu}\sum_{i=1}^{\mu}\boldsymbol{\sigma}_{i:\lambda}^{(g)}\right) \;\middle|\; \bar{\boldsymbol{\sigma}}^{(g)}\right] \\
&\leq \ln \mathrm{E}\left[\frac{1}{\mu}\sum_{i=1}^{\mu}\boldsymbol{\sigma}_{i:\lambda}^{(g)} \;\middle|\; \bar{\boldsymbol{\sigma}}^{(g)}\right] \text{ (Jensen's inequality)} \\
&= \ln\left\{\frac{1}{\mu}\sum_{i=1}^{\mu}\bar{\boldsymbol{\sigma}}^{(g)}\,\mathrm{E}\left[\exp\left(\tau'\mathcal{N}(0,1)\right)\right]\mathrm{E}\left[\exp\left(\tau\boldsymbol{\mathcal{N}}(\mathbf{0},\mathbf{I})\right)\right]\right\} \\
&= \ln \bar{\boldsymbol{\sigma}}^{(g)} + \frac{\tau'^2+\tau^2}{2}.
\end{aligned}$$

Thus, $\ln \bar{\boldsymbol{\sigma}}$ is not guaranteed to be unbiased. On the contrary, $\ln \boldsymbol{\sigma}_{\mathrm{MLE}}$ is unbiased because

$$\begin{aligned}
\mathrm{E}\left[\ln \boldsymbol{\sigma}_{\mathrm{MLE}}^{(g+1)} \mid \boldsymbol{\sigma}_{\mathrm{MLE}}^{(g)}\right] &= \mathrm{E}\left[\frac{1}{\mu}\sum_{i=1}^{\mu}\ln \boldsymbol{\sigma}_{i:\lambda}^{(g)} \;\middle|\; \boldsymbol{\sigma}_{\mathrm{MLE}}^{(g)}\right] \\
&= \frac{1}{\mu}\sum_{i=1}^{\mu}\mathrm{E}\left[\ln \boldsymbol{\sigma}_{\mathrm{MLE}}^{(g)} + \tau'\mathcal{N}(0,1) + \tau\boldsymbol{\mathcal{N}}(\mathbf{0},\mathbf{I})\right] \\
&= \ln \boldsymbol{\sigma}_{\mathrm{MLE}}^{(g)}.
\end{aligned}$$

The arguments above can also be verified from experimental aspect. To compare the

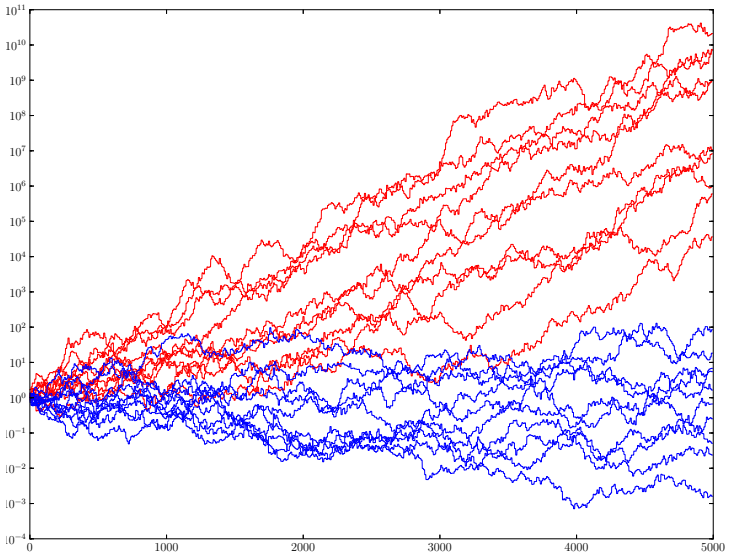

Figure 2.2: Step-size σ versus number of function evaluations of 20 runs on a purely random fitness function in dimension 10. x-axis: evaluation counts, y-axis: global step-size.

behaviors of two estimators experimentally, a simplified (μ, λ)-MSC-ES algorithm (shown in Algorithm 2.2), in which the covariance matrix does no exist and only global step-size σ is self-adaptive, is tested on a random fitness function and the random walk of $\log \sigma$ is recorded and plotted in Figure 2.2. The dynamics of $\log \sigma$ under recombination operator $\bar{\sigma}$ is shown in 10 red curves while that under recombination operator σ_{MLE} is shown in 10 blue curves. It is pretty clear that there is a huge positive bias when using $\bar{\sigma}$ for recombination. Such observation shows the original suggestion of recombination operator may not behave as what we want despite its great success. The difference between the effects of these two recombination operators on performance needs to be investigated further.

2.2.5 Cumulative parameter adaptation

As discussed in the last section, the adaptation mechanism of global step-size σ suffers a positive bias. The newly obtained σ after recombination is getting larger quickly. In order to reduce such unsatisfactoriness, we could update the global step-size cumulatively instead of replacing the old one with the new one, like the updating rules in CSA and CMA techniques (see the next section). The cumulative updating rule reads,

$$\bar{\sigma}^{(g+1)} = (1 - c)\bar{\sigma}^{(g)} + \frac{c}{\mu} \sum_{i=1}^{\mu} \sigma_{i:\lambda}^{(g)}. \tag{2.6}$$

The notations above is the same as the last section. The changing speed of $\bar{\sigma}$ is controlled by a constant c. Using this updating rule, the variation of the newly recombined parameter has relatively small effect on $\bar{\sigma}$, compared to the original updating method. Such updating rule could also be applied to the individual step-size $\bar{\boldsymbol{\sigma}}$. However, in order to make a quick, simple investigation of the functionality of this new updating rule, we just apply Eq. 2.6 to the simplest self-adaptive ES, in which the covariance matrix does not exits at all. This ES algorithm is named Simple-(μ, λ)-MSC-ES here and is shown in the following algorithm.

Algorithm 2.2 SIMPLE-(μ, λ)-MSC-ES

```
Initialize population
t ← 0
repeat
    t ← t + 1
    x̄ = (1/μ) Σ_{i=1}^{μ} x_i
    σ̄ = (1/μ) Σ_{i=1}^{μ} σ_i
    for i = 1 → λ do
        σ_i ← σ̄ exp(τ N(0,1))
        x_i ← x̄ + σ_i N(0, I)
        φ_i ← f(x_i)
    end for
    Select μ best from (x_i, σ_i)
until termination criterion fulfilled.
```

The suggested setting of τ is $1/\sqrt{N}$. The cumulative version of it is termed Cumulative-Simple-(μ, λ)-MSC-ES, or Cu-Simple-(μ, λ)-MSC-ES for short and is listed in Algorithm 2.3 below.

Algorithm 2.3 Cu-SIMPLE-(μ, λ)-MSC-ES

Initialize population
$t \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  $\bar{\mathbf{x}} = \frac{1}{\mu} \sum_{i=1}^{\mu} \mathbf{x}_i$
  $\bar{\sigma} = (1 - c)\bar{\sigma} + \frac{c}{\mu} \sum_{i=1}^{\mu} \sigma_i$
  for $i = 1 \rightarrow \lambda$ **do**
    $\sigma_i \leftarrow \bar{\sigma} \exp(\tau \mathcal{N}(0, 1))$
    $\mathbf{x}_i \leftarrow \bar{\mathbf{x}} + \sigma_i \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$
    $\phi_i \leftarrow f(\mathbf{x}_i)$
  end for
  Select μ best from $(\mathbf{x}_i, \sigma_i)$
until termination criterion fulfilled.

The parameter τ is still $1/\sqrt{N}$ and c should be set to $1/\sqrt[4]{N/2}$ by the author. There are two other motivations for this new updating rule. First, the local information is not reliable as discussed in Section 2.2.3. However, cumulated information over history sometimes gives us a improvement on the accuracy of the information. Second, the standard step-size updating rule somehow suffers an *over fit* effect (discussed in detail in Section 2.3.1). The newly recombined global step-size fits the current selected step-size population most. If we just replace the old parameter with the new one, the information contained in the old parameter is lost. Then it is reasonable to combine both of the parameters. The empirical comparison of the two algorithms is shown in Chapter 5.

2.3 From (μ, λ)-MSC-ES to $(\mu/\mu_{\mathbf{w}}, \lambda)$-CMA-ES

As the result of continuing efforts to rectify the unpleased aspects in (μ, λ)-ES, $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES was proposed in 1996 [25], and is considered as the state-of-the-art evolution strategy algorithm. This algorithm tries to adjust and adapt the strategy parameters based on much reliable information even the population is small. The history from (μ, λ)-MSC-ES to $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES is quite important and reflects the evolution of people's intuitions and viewpoints on ES adaptation. Although there are three other ESs between them, DR1, DR2, DR3, CMA-ES is considered as the "ultimate" version of DR-ES series and their ideas are consistent. Therefore, only the CMA-ES is discussed here. The introduction of CMA-ES here is built on the comparison with the essence of MSC-ES. The purpose is to make the difference between MSC-ES and CMA-ES clear and to facilitate the the discussion of new ES variants later in this thesis. There are already some comprehensive introduction to CMA-ES. For detailed explanation of CMA-ES, please read [24].

The key mechanism in evolution strategy is how to adjust the strategy parameters properly. The strategy parameters for most of the ES algorithm is the global step-size σ and the covariance matrix $\mathbf{C}$, as the mutation of solution vector is usually generated by $\boldsymbol{\mathcal{N}}(\mathbf{0}, \sigma^2\mathbf{C})$. The adaptation mechanism of $(\sigma, \mathbf{C})$ always begins with obtaining the information related to the parameters, and then decides the new parameters based on the information. The performance of such procedure is highly dependent on two questions:

1. How to obtain the information that the adaptation mechanism needed precisely?
2. How to use to information to guide the parameter adjustment reasonably?

Indeed, the method to get the information is highly dependent on the adaptation mechanism we choose. Note that the only precise and reliable information is the fitness values. If other type of information is needed, it could only be inferred from the raw fitness values.

2.3.1 Covariance matrix adaptation

In MSC-ES, the mechanism of covariance matrix adaptation is to construct a evolution process for the covariance matrices as illustrated in Equation 2.2, 2.3. Then the fitness information of the covariance matrices are necessary for the later selection operator. However, the fitness values of the solution vectors is the only information source, which are trusted as the fitness of covariance matrices and used again by the evolution dynamics of covariance matrices. As discussed before, such fitness information for covariance matrices is highly disturbed. The mutative-self-adaptation mechanism simply fails to obtain reliable information for the covariance adaptation at the beginning. Building a second level evolution dynamics (an evolution process of endogenous strategy parameters) within a evolution strategy algorithm seems not a quite good approach because the first level evolution dynamics (the evolution of the solution vectors) could not provide the fitness value required by the second level.

In $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES, σ and $\mathbf{C}$ does not mutate at all and is fixed for all the solution vectors generation within one generation. The solution vector read,

$$\mathbf{x}_i^{(g+1)} = \mathbf{x}^{(g)} + \sigma^{(g)}\mathbf{z}_i^{(g)}, \quad 1 \leq i \leq \lambda$$

where $\mathbf{z}_i^{(g)} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{C}^{(g)})$, $\mathbf{x}^{(g)}$ is the parent and the superscript (g) is the generation index. After the evaluation, we could order the solution vectors as $\{\mathbf{x}_{i:\lambda}\}_{1\leq i\leq\lambda}$ based on their corresponding fitness value. At this time, the mechanism in CMA-ES manages to investigate what information the best μ solution vectors can give. By the maximal likelihood estimation method, the covariance generating the best μ mutations of solution vectors with maximal probability reads [25],

$$\mathbf{C}_\mu^{(g+1)} = \frac{1}{\mu}\sum_{i=1}^{\mu}\left(\mathbf{x}_{i:\lambda}^{(g)} - \mathbf{x}^{(g)}\right)\left(\mathbf{x}_{i:\lambda}^{(g)} - \mathbf{x}^{(g)}\right)^T, \tag{2.7}$$

where

$$P(\mathbf{x}_{1:\lambda}^{(g)}, \ldots, \mathbf{x}_{\mu:\lambda}^{(g)} \mid \mathbf{C}_\mu^{(g+1)}) \longrightarrow \max.$$

The information needed by estimation Equation 2.7 is just the goodness ordering of the solution vectors, which is directly obtained from evaluation and reliable. Compared to MSC, the guess for good covariance matrices is done by choosing the covariance matrices related to the μ best solution vectors, which is based on the unreliable information as discussed before. At this moment, the new estimation method is already better.

The next question is how to use the estimation $\mathbf{C}_\mu^{(g)}$ to adjust the current covariance. Although $\mathbf{C}_\mu^{(g)}$ is the optimal choice for the μ best solutions, there are many other possible good solutions which are not realized in one generation. The current covariance matrix may generate other possible good mutations at high probability. Thus it is reasonable to add $\mathbf{C}_\mu^{(g)}$ to the current covariance to keep both the new information and the old one. The updating rule reads,

$$\mathbf{C}^{(g+1)} = (1 - c_{\mathrm{cov}})\mathbf{C}^{(g)} + c_{\mathrm{cov}}\frac{1}{\sigma^2}\mathbf{C}_\mu^{(g+1)}, \tag{2.8}$$

where the good choice for c_{cov} is $min(1, \mu/N^2)$ according to [25]. This mechanism is termed as *covariance matrix adaptation*. Let's turn to another viewpoint to compare this new method to the covariance matrix adaptation in MSC-ES. The covariance matrix can be treated as

a probabilistic model for mutation generation. The target of fitting this model is to find a good covariance matrix which could generate a possible good mutation at relatively high probability. The μ best mutations is the training set. In this sense, Equation 2.7 is the simple learning rule and it is *over fitting* the model to the training set. Cumulating the over fitted parameter to the current parameters can largely relieve the over fitting problem. Thus, the updating rule above seems reasonable.

For MSC-ES, its adaptation mechanism suffers from the over fitting problem. As the intermediate recombination is also applied to the covariance matrix (individual step-size and rotation angles), the recombined covariance matrix is simply the maximal likelihood estimation for μ best covariance matrix mutants and faces the same problem with Equation 2.7.

There is actually another component in the Equation 2.8, named *rank-one-update*, which cumulates the selected mutation vectors and increases the reliability of the covariance matrix adaptation. It is not relevant to the comparison here and is omitted.

2.3.2 Step-size adaptation

In MSC-ES, the adaptation mechanism of global step-size is the same with that of covariance matrix and is realized as part of individual step-size (see Section 2.2.2). Thus, it suffers from the same problem. As for the CMA-ES, the difference of two consecutive parental vectors disregarding step-size is termed as *step* and reads,

$$\frac{\mathbf{x}^{(g+1)} - \mathbf{x}^{(g)}}{\sigma^{(g)}} = \frac{1}{\mu} \sum_{i=1}^{\mu} \mathbf{z}_{i:\lambda}^{(g)}$$

The relation above is established under random selection because summation on a sequence of the independent normal vector is summing their mean and covariance in effect. The step-size adaptation based on the following observations on steps [24]:

1. If the sequence of successive steps are anti-correlated, the length of the sum sequential steps would be relatively small. Because the anti-correlated steps are canceling each other. In this situation, the algorithm is inefficient and the step-size should be decreased.

2. If the sequence of successive steps are correlated, the length of the sum sequential steps would be relatively large. Because they are roughly pointing to the similar directions. This indicates the same distance can be covered by fewer but longer steps into the same directions. Thus, the step-size should be increased.

3. If the sequence of successive steps are approximately perpendicular to each other, namely uncorrelated, then the situation is desired and there is no need to adjust the step-size.

Thus, the sum of sequence of successive steps is regarded as the indicator of how ES performs on landscapes in the recent history. It is termed as (conjugate) *evolution path*. To decide whether the evolution path is "long" or "short", we compare the length of the evolution path to the *expected* path length situation 3 above. Under random selection (or random fitness) consecutive steps are independent and uncorrelated. Therefore, the expected evolution path length under random selection is used to represent the situation 3. In order to make the

steps realized by different covariance matrices comparable, the steps are first conjugated with $\mathbf{C}^{(g)^{-\frac{1}{2}}}$ then added to the evolution path [25],

$$\boldsymbol{p}_\sigma^{(g+1)} = (1-c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu}\ \mathbf{C}^{(g)^{-\frac{1}{2}}} \frac{\mathbf{x}^{(g+1)} - \mathbf{x}^{(g)}}{\sigma^{(g)}}. \tag{2.9}$$

The factor $\sqrt{c_\sigma(2-c_\sigma)\mu}$ is chosen such that

$$\boldsymbol{p}_\sigma^{(g+1)} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$$

if the right-hand side of Equation 2.9 satisfies the following conditions,

$$\boldsymbol{p}_\sigma^{(g)} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}) \quad \text{and} \quad \frac{\mathbf{x}^{(g+1)} - \mathbf{x}^{(g)}}{\sigma^{(g)}} \sim \frac{1}{\sqrt{\mu}}\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{C}^{(g)}),$$

which could only be achieved under the random selection. Therefore, the evolution path computed by Equation 2.9 under random selection is a standard normal vector and its expectation is $\mathrm{E}\,||\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})|| = \sqrt{2}\Gamma(\frac{N+1}{2})/\Gamma(\frac{N}{2})$. Therefore we can update $\sigma^{(g)}$ by comparing $||\boldsymbol{p}_\sigma^{(g+1)}||$ with $\mathrm{E}\,||\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})||$, which is [25],

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\boldsymbol{p}_\sigma^{(g+1)}||}{\mathrm{E}\,||\mathcal{N}(\mathbf{0}, \mathbf{I})||} - 1\right)\right) \tag{2.10}$$

where the standard setting of parameter c_σ, d_σ is,

$$c_\sigma = \frac{\mu+2}{N+\mu+3}, \quad d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu-1}{N+1}}\right) + c_\sigma$$

This elegant technique is well-known as *cumulative step length adaptation*(CSA). Unlike the step-size adaptation in MSC, the CSA technique completely is built on reliable information and therefore considered more advanced. In addition, the updating rule (Equation 2.10) is unbiased under random selection due to,

$$\begin{aligned}
\mathrm{E}\left[\ln \sigma^{(g+1)} \mid \sigma^{(g)}\right] &= \mathrm{E}\left[\ln \sigma^{(g)} + \left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\boldsymbol{p}_\sigma^{(g+1)}||}{\mathrm{E}\,||\mathcal{N}(\mathbf{0}, \mathbf{I})||} - 1\right)\right) \;\middle|\; \sigma^{(g)}\right] \\
&= \ln \sigma^{(g)} + \frac{c_\sigma}{d_\sigma}\left(\frac{\mathrm{E}\,||\boldsymbol{p}_\sigma^{(g+1)}||}{\mathrm{E}\,||\mathcal{N}(\mathbf{0}, \mathbf{I})||} - 1\right) \\
&= \ln \sigma^{(g)}
\end{aligned}$$

Compared to the biased adaptation of MSC argued in Section 2.2.4, the CSA technique seems more plausible again.

Chapter 3

Mirrored sampling technique

All the derandomization techniques discussed in Chapter 2 are working and designed for the level 2 of a ES, namely the control mechanism. These techniques have already been extensively exploited to improve the ES performance. Furthermore, the randomness in level 1 of the ES, which mainly involves the mutation operator, can also been reduced in certain degree to improve the performance.

The general idea is that the realized mutations in high dimensional space could be ill-generated such that the search direction and search step-sizes implied by the covariance matrix are not realized with satisfaction, due to the nature of random sampling. Such bad sampling case will be illustrated in this chapter. The "mirrored" sampling technique serves as the first example of level 1 derandomization technique, aiming at improving the mutation sampling.

The mirrored sampling technique has been formalized and extensively analyzed under the extremely simple case ($(1, \lambda)$-ES) in [15] and benchmarked in the special case of the $(1, 2)$- and the $(1, 4)$-CMA-ES in [1, 2, 3, 4]. In this chapter, the basic mirrored sampling technique is introduced, and then its advantages and disadvantages are summarized. Some theoretical results on simple fitness function is also given and compared to the that of the standard ES. Finally, a possible improvement of mirrored sampling is proposed.

3.1 Intuition and the algorithm

The mirrored sampling, a modification made on the offspring generation process (or sampling, mutation), is quite a simple and elegant idea in which a single random vector is used to create two offspring, one by adding and the other by subtracting the vector. In order to compare the mirrored idea to the standard offspring generation in ES, it is better to formalize them both with a uniform representation and make themselves independent of the rest of the ES algorithm. In addition, by doing this, we could construct a pool of basic operators having the same purpose, like what we are going to build, a pool of mutation operators. Such pools may act as the repertoire for an algorithm designer. Once we have our pool of mutation operators, of recombination and of selection, the whole ES algorithm could be realized as the combination of the operators from these pools. In this way, the standard mutation and the mirrored alternative are formalized in Algorithm 3.1 and 3.2 uniformly.

Algorithm 3.1 SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$

given: $\mathbf{z}_i, \mathbf{x}_i \in \mathbb{R}^d$
$\mathbf{B}, \mathbf{D} \leftarrow$ EIGEN-DECOMPOSITION$(\mathbf{C})$
for $i = 1 \rightarrow \lambda$ **do**
  $\mathbf{z}_i \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_i \leftarrow \mathbf{x} + \sigma \mathbf{B}\mathbf{D}\mathbf{z}_i$
end for

Algorithm 3.2 MIRRORED-SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$

given: $\mathbf{z}_i, \mathbf{x}_i \in \mathbb{R}^d$
$\mathbf{B}, \mathbf{D} \leftarrow$ EIGEN-DECOMPOSITION$(\mathbf{C})$
if $\lambda \neq 0 \pmod 2$ **and** $\mathbf{z}_{\text{last}}$ is present **then**
  $\mathbf{x}_1 \leftarrow \mathbf{x} - \sigma \mathbf{B}\mathbf{D}\mathbf{z}_{\text{last}}$
  $\lambda \leftarrow \lambda - 1$
  Delete static variable $\mathbf{z}_{\text{last}}$.
end if
for $i = 1 \rightarrow \lambda$ **do**
  if $i \equiv 0 \pmod 2$ **then**
    $\mathbf{x}_i \leftarrow \mathbf{x} - \sigma \mathbf{B}\mathbf{D}\mathbf{z}_{i-1}$
  else
    $\mathbf{z}_i \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x} + \sigma \mathbf{B}\mathbf{D}\mathbf{z}_i$
  end if
end for
if $\lambda \neq 0 \pmod 2$ **then**
  Create static variable $\mathbf{z}_{\text{last}}$.
  $\mathbf{z}_{\text{last}} \leftarrow \mathbf{z}_\lambda$
end if

The Algorithm 3.1 gives the process of the standard mutation operator. It takes the parental vector $\mathbf{x}$, the global step-size σ, the current covariance matrix $\mathbf{C}$ and the number of the offspring λ as inputs, samples λ i.i.d (independently and identically distributed, we will use such abbreviation in the context) mutation vectors following $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{C})$ and returns the sum of $\mathbf{x}$ and each mutation vector. The functionality of the standard mutation operator, as shown in this sampling procedure, is decomposed and decoupled from other components of ESs. Furthermore, its interfaces (inputs and outputs) with other components are formalized such that it could be compared to or replaced by a alternative formalized in the same manner without violating the rest of ES algorithms. Note that the EIGEN-DECOMPOSITION in the algorithm stands for the numerical eigen decomposition procedure which is irrelevant to our concerns here.

As an alternative, the mirrored sampling procedure is shown in Algorithm 3.2. During each ES generation, only half of the vectors are sampled, namely $\{\mathbf{x}_{2i-1}\}_{1 \leq i \leq \lambda/2}, \mathbf{x}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{C})$ for even λ. Each random vector $\mathbf{x}_{2i-1}$ is used to generate two offspring, the usual one $\langle \mathbf{x} \rangle + \mathbf{x}_{2i-1}$ and the mirrored offspring $\langle \mathbf{x} \rangle - \mathbf{x}_{2i-1}$. Those two offspring are *symmetric* or *mirrored* to the current centre mass $\langle \mathbf{x} \rangle$, from which this technique gets its name. In order to make the argument here clearer, the mutations sampled from the distribution are termed as *realized* mutations while the mirrored mutations are termed as *mirrored* mutations. For odd λ, we begin to generate $\lceil \lambda/2 \rceil$ realized offspring in the first generation and results in $\lceil \lambda/2 \rceil$ mirrored offspring. Then all of the realized offspring and $\lceil \lambda/2 \rceil - 1$ mirrored ones are used immediately while the extra one mirrored mutation is kept to the next generation. Then in the next generation the extra mirrored offspring is used and only $\lfloor \lambda/2 \rfloor$ realized

mutations are needed to be drawn. The following generations would repeat this procedure. Please check the static variable $\mathbf{z}_{\text{last}}$ in Algorithm 3.2, which facilitates the extra realized mutation vector. The pseudo-code is pretty much self-explanatory. We followed the notation proposed in [15] such that any ES algorithm using mirrored sampling would be denoted as (μ, λ_m)-ES. Consequently, in the $(1+1_m)$-ES, a mirrored mutation is used if and only if the iteration index is even. Note that in the (μ, λ_m)-ES an offspring and its mirrored counterpart are entirely dependent. In order to convey the mirrored idea clearer, a typical situation of ES evolution when applying mirrored sampling is shown below.

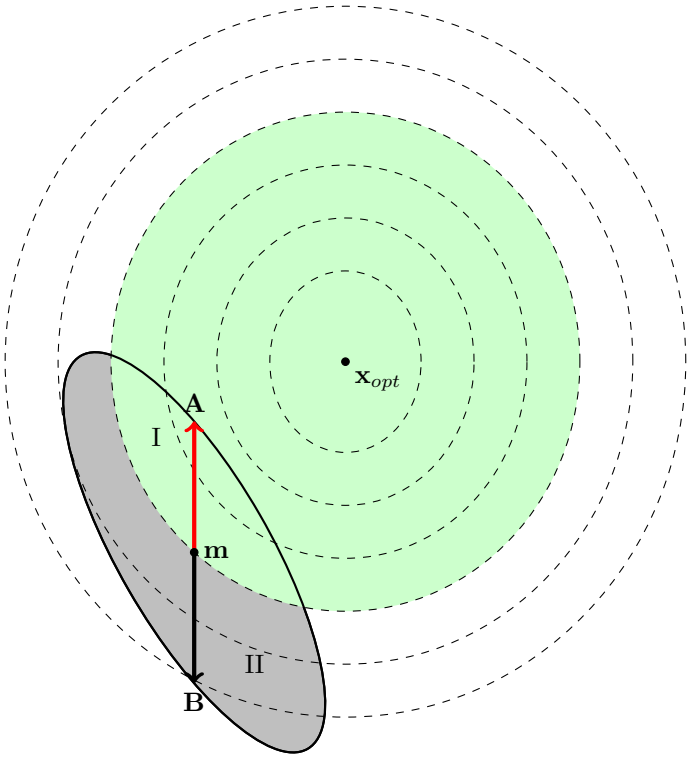

Figure 3.1: Sampling process of mirrored sampling.

In Figure 3.1 we draw the sampling process of a ES algorithm on a 2-dimensional unimodal function. The landscape (or contours) of 2-dimensional unimodal object function is shown with the dashed ellipses. The solid black ellipse centred at $\mathbf{m}$ is one example of the lines of equal mutation probability density and also represents the shape of the current covariance matrix. A mirrored pair of offspring are shown by $\mathbf{A}$ and $\mathbf{B}$. If a new offspring is generated in the light green area marked as I (like the offspring $\mathbf{A}$ in the figure), then the fitness of the offspring is better than the parent. Otherwise, any new offspring drawn on light gray area marked by II is going to be worse. Let us term region I as *progress region*.

The intuition of the idea comes from the observation of the following extreme case of offspring generation. The left part of Figure 3.2 illustrates a glimpse of the bad realized high-dimensional mutations of a $(1, 4)$-ES served as our "extremely bad case". The progress

region is the light gray area while all four offspring are out of the progress region. After crushing the high dimension search space into a plane, the mutations are actually uniformly distributed on the the solid black ellipse which represents the covariance matrix (see Section 3.2.1 for detail). In such extreme case no mutation makes progress and renders this generation inefficient. In addition, the chance of being in this situation is not as low as expected. If the length of fraction of the ellipse surrounding the progress region is l and the total length of the ellipse is L, the probability of being in the bad case is $(1 - l/L)^\lambda$, due to the uniformity on hyper ellipse of mutations. Given the progress region is usually small, the probability simply can not vanish.

Then the mirrored sampling gives a simple and elegant trick. The reason is as follows: the realized random vectors and their mirrored counterparts would locate in the space more evenly. Therefore the population mixture of realized offspring and their mirrored counterparts would increase the probability of hits in the progress region even without any information about the progress region. The right part of Figure 3.2 provides a brief situation after applying mirrored sampling to the left part. The red mutations $\mathbf{x}_3$ and $\mathbf{x}_4$ are the mirrored mutations from ill-generated $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_4$ hits the progress region. The actual effects can be thought as the progress region is "mirrored" (region I is mirrored to II in the figure) so that the probability to generate a good offspring is increased.

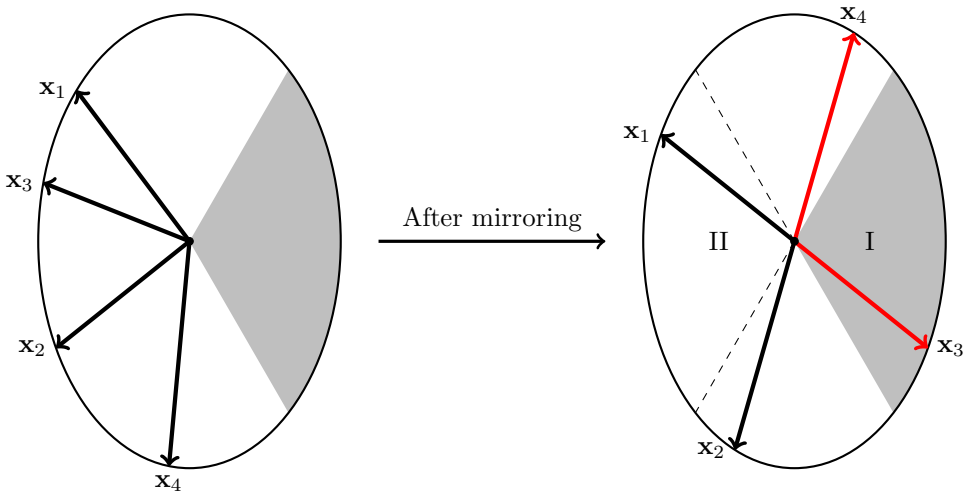

Figure 3.2: The improvement after applying mirrored sampling.

All the discussions above are based on small λ value. When the offspring population is really big, the realized population would converge to the expectation. If $\lambda/2$ offspring (take the even population as the simplest cases, for odd population, the following argument still holds) are realized from the distribution, then the realized offspring would contribute $\lambda l/2L$ good offspring. Then again the realized offspring are mirrored, which transforms $\lambda l/2L$ bad offspring from $\lambda(1 - l/S)/2$ to good one. Combining these two contributions, the total expected good offspring would be

$$\frac{\lambda}{2}\frac{l}{L} + \frac{\lambda}{2}\frac{l}{L} = \frac{\lambda l}{L},$$

which has no difference with standard sampling. Therefore, the performance of mirrored sampling ES is expected to better than that of standard ES in case of small offspring population and converge to that of standard ES when λ gets large.

Mirrored sampling could adjust the bad-realized samples to much more reasonable cases so that the probability of drawing samples in the "good" area is increased. When λ value is very large, the mirrored sampling method makes no difference with the standard sampling method. Due to the convexity of the unimodal contour lines, the size of the progress region is always much smaller than the non-progress region. Consequently, the probability of drawing all the offspring in the non-progress region can not be neglected. This observation could lead to the result that mirrored sampling could work at any unimodal fitness function under a small offspring population to improve the convergence velocity.

3.2 Theoretical aspects of $(1, \lambda_{\mathrm{m}})$-ES

The theoretical analysis of $(1, \lambda_{\mathrm{m}})$-ES on the sphere function has already been developed by Auger et. al. [15], their sophiscated derivation is based on an artificial step-size (global σ) setting named *scale-invariant step-size*, where $\sigma^{(g)}$ of the g-th generation is proportional to the distance to the optimum, namely, $\sigma^{(g)} = \sigma \cdot \left\|\mathbf{X}^{(g)} - \mathbf{x}_{opt}\right\|$ ($\mathbf{X}^{(g)}$ is the parent at generation g) for $\sigma > 0$. In addition, they argue that the logarithm of progress on the distance to the optimum since the first generation is linearly related to the number of function evaluations consumed so far. Let T_k be the number of the evaluations consumed until generation k and suppose the optimum is $\mathbf{0}$ w.l.o.g, the almost sure (a.s.) linear convergence looks like [7, Eq. 10.10]

$$\frac{1}{T_k} \ln \frac{\left\|\mathbf{X}^{(k)}\right\|}{\left\|\mathbf{X}^{(0)}\right\|} \to c \quad a.s.$$

Note that the valid constant c should be negative to make "convergence" meaningful and the smaller c is, the faster the ES algorithm performs. Furthermore, the c value for $(1, \lambda_m)$-ES on sphere function is derived as [15]

$$\frac{1}{\lambda k} \ln \frac{\left\|\mathbf{X}^{(k)}\right\|}{\left\|\mathbf{X}^{(0)}\right\|} \xrightarrow[k\to\infty]{} \frac{1}{2\lambda} \mathrm{E}\left[\ln\left(1 + \sigma \min_{1\le i\le \lambda/2}\left(-2|[\boldsymbol{\mathcal{N}}^i]_1| + \sigma\left\|\boldsymbol{\mathcal{N}}^i\right\|^2\right)\right)\right] \quad a.s.,$$

where $(\boldsymbol{\mathcal{N}}^i)_{1\le i\le \lambda/2}$ are $\lambda/2$ i.i.d. Gaussian vectors and $[\boldsymbol{\mathcal{N}}^i]_1$ represents the first dimension of random vector $\boldsymbol{\mathcal{N}}^i$. In order to make a reasonable comparison with the standard $(1, \lambda)$-ES, the convergence formula of $(1, \lambda)$-ES on sphere is also given in [15]. The summary of their results is not going to be presented in the following. For more detail of their approach and proofs, we suggest to read [15] and they also provide the theoretical result of $(1 + \lambda_m)$-ES in [5].

Despite the elegance of their theoretical work, some unpleasant facts still remain. For example, whether such artificial step-size setting can approximate the real evolution adaptation process in ES is not clear yet. Besides this unsatisfactoriness, we are much more interested in how mirrored sampling actually effects an evolution cycle treating global step-size σ as an variable or a parameter of our progress model. The pre-set scale-invariant step-size rule simply contradicts our purpose, which makes it difficult to compare their theoretical results to the classical results of standard ES by Beyer [10]. In addition and what is most important, through their derivation, how mirrored sampling outperforms the standard sampling

in essence is still not very clear. Thus, it is quite necessary to use standard progress rate theoretical framework to analyze.

3.2.1 The properties of the mutation vector

It is important to get familiar with some asymptotic properties of the N-dimensional Gaussian mutation vector $\mathbf{z} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \sigma^2\mathbf{I})$. The density function of $\mathbf{z}$ is [29]

$$p(\mathbf{z}) = \frac{1}{(2\pi)^{n/2}\sigma} \exp\left(-\frac{\mathbf{z}^T\mathbf{z}}{2\sigma^2}\right). \tag{3.1}$$

The first nice property is that $\mathbf{z}$ is spherical symmetry invariant against any rotation matrix $\mathbf{O}$. In other words, the multivariate Gaussian distribution depends only on the length of $\mathbf{z}$ instead of the direction. The random variable $\mathbf{z}' = \mathbf{O}\mathbf{z}$ has the same distribution as $\mathbf{z}$. The proof is quite simple: Using the *Jacobian transformation* [29] of the probability distribution, we could substitute $\mathbf{z}$ by $\mathbf{O}^{-1}\mathbf{z}'$ into Equation 3.1 and then the distribution of $\mathbf{z}'$ reads,

$$p(\mathbf{z}') = \frac{1}{(2\pi)^{n/2}\sigma|J|} \exp\left(-\frac{(\mathbf{O}^{-1}\mathbf{z}')^T\mathbf{O}^{-1}\mathbf{z}'}{2\sigma^2}\right) \quad |J| = \left|\det\left(\frac{\partial\mathbf{z}'}{\partial\mathbf{z}}\right)\right| = |\det\mathbf{O}|,$$

where $|J| = |\det\mathbf{O}|$ is the absolute value of Jacobian determinant. Due to the orthogonality of $\mathbf{O}$, $|J| = 1$ and $\mathbf{O}^{-1} = \mathbf{O}^T$, we have

$$p(\mathbf{z}') = \frac{1}{(2\pi)^{n/2}\sigma} \exp\left(-\frac{\mathbf{z}'^T\mathbf{z}'}{2\sigma^2}\right).$$

Such *rotation-invariant* property would be critical to our analysis. The length of mutation vector $\mathbf{z}/\sigma$, $\chi = \sqrt{\mathbf{z}^T\mathbf{z}}/\sigma$, follows the χ-distribution [29]

$$p(\chi; N) = \frac{2\chi^{N-1}e^{\chi^2/2}}{2^{N/2}\Gamma(\frac{N}{2})}$$

where N is the dimensionality of $\mathbf{z}$ and is called *degrees of freedom*. Its expectation and variance are,

$$\mathrm{E}[\chi] = \sqrt{2}\sigma\frac{\Gamma\left(\frac{N+1}{2}\right)}{\Gamma\left(\frac{N}{2}\right)} \simeq \sqrt{N}, \quad \mathrm{Var}[\chi] \simeq \frac{1}{2}$$

If the degree of freedom is big enough ($N \to \infty$), the asymptotic form of the expectation and the variance of χ-distribution above are valid [11]. Consequently, the mean and variance of mutation vector $\mathbf{z}$ reads,

$$\mathrm{E}[||\mathbf{z}||] = \mathrm{E}[\sigma\chi] \simeq \sigma\sqrt{N}, \quad \mathrm{Var}[||\mathbf{z}||] = \mathrm{Var}[\sigma\chi] \simeq \frac{1}{2}\sigma^2.$$

This is the second property we need in the following: compared to the expectation of mutation length, the standard deviation $\sigma/\sqrt{2}$ is relatively small and independent of dimensionality N. Therefore, for approximation purpose, the standard deviation could be ignored so that mutation vectors end at the hypersphere with radius $\sigma\sqrt{N}$ when N is large enough.

3.2.2 The $(1, \lambda_{\mathbf{m}})$ evolution strategy

The following derivations are based on the framework by Beyer [10, 9]. The general approach of $(1, \lambda)$-ES is introduced at first. Then some modifications are made to apply the analysis to the mirrored situation. Finally, the comparison of the results of mirrored sampling and the standard $(1, \lambda)$ algorithm is shown. The basis of the analysis is shown in Figure 3.3. Let $\mathbf{m}$ be the current parent which is at a distance R from the optimum $\mathbf{x}_{\text{opt}}$. The hypersphere centered at $\mathbf{m}$ has a radius of $\sigma\sqrt{N}$ and represents all the possible offspring. A mutation is indicated by vector $\mathbf{z}$. Because of the rotation-invariant property of multivariate Gaussian distribution, the vector $\mathbf{p} = \mathbf{m} - \mathbf{x}_{\text{opt}}$ could be rotated around $\mathbf{x}_{\text{opt}}$ such that $\mathbf{p}$ is parallel to the first canonical basis $\mathbf{e}_1$ (other bases are the same) of the space, namely $\mathbf{p}^T\mathbf{e}_1 = \pm\|\mathbf{p}\|$. After such rotation, the projection of any mutation vector $\mathbf{z}$ onto $\mathbf{p}$ is exactly the first element of the mutation vector which is denoted by z in the figure. As a basic property of multivariate normal vector, every element of the vector is again normally distributed. This can be proved simply by marginalizing out all other dimensions except the dimension we want, in the density function in Equation 3.1. Thus, projection z is $\mathcal{N}(0, \sigma^2)$ distributed with the density,

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{z^2}{2\sigma^2}\right). \tag{3.2}$$

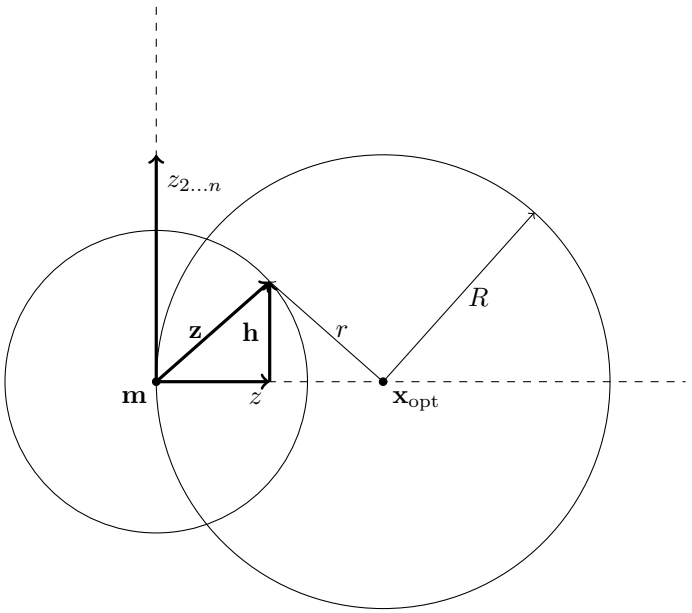

Figure 3.3: Projection of n-dimensional sphere landscape onto the two-dimensional plane after the rotation.

After the rotation of landscape and the mutations, the following relations hold for any mutations,

$$\mathbf{z} = \mathbf{h} + z\mathbf{e}_1 \quad \text{and} \quad \mathbf{e}_1^T\mathbf{h} = 0,$$

based on which the distance of the offspring to the optimum r can be calculated as

$$\begin{aligned} r &= \sqrt{\mathbf{h}^2 + (R - z)^2} \\ &= \sqrt{\mathbf{z}^2 + R^2 - 2Rz} \end{aligned}$$

However, $\mathbf{z}$ is χ distributed while z is normal distributed, which makes it difficult to determine the distribution of r. Therefore we replace $\mathbf{z}$ by its expectation $\sigma\sqrt{N}$. The asymptotic r reads

$$r \simeq \sqrt{\sigma^2 N + R^2 - 2Rz}. \tag{3.3}$$

The validity of this approximation can be found in [11, Page 63]. This asymptotic form also brings another benefit in $(1, \lambda)$-ES. Consider the collection of the distance to the optimum of each offspring, $\{r_i\}_{1 \leq i \leq \lambda}$. The selected offspring would be the one with smallest r value, namely the smallest order statistic $r_{1:\lambda}$. Due to Equation 3.3, selecting the offspring with the largest projection z is to select that with $r_{1:\lambda}$. Therefore, the distribution and the order statistics of r are uniquely determined by z, which is much easier to manipulate. The convergence velocity (or progress rate) is defined as the expected change of fitness value or distance to the optimum in one generation. For the sphere function, it reads

$$\begin{aligned} \varphi_{1,\lambda} &= \mathrm{E}[R^2 - r_{1:\lambda}^2] \\ &= \mathrm{E}[2Rz_{\lambda:\lambda} - \sigma^2 N] \end{aligned} \tag{3.4}$$

In order to calculate the expectation above, the integral limits and the p.d.f of $z_{\lambda:\lambda}$ needs to be determined. For $(1, \lambda)$-ES the domain of $z_{\lambda:\lambda}$ is $(-\infty, \infty)$. The p.d.f of $z_{\lambda:\lambda}$ is [17, Page 8]

$$p_{\lambda:\lambda}(z) = \lambda p(z) \Phi \left(\frac{z}{\sigma}\right)^{\lambda - 1}, \tag{3.5}$$

where $\Phi(z)$ is the cumulative probability distribution of Gaussian distribution having the form,

$$\Phi(z) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{1}{2}t^2} \, \mathrm{d}t.$$

Putting all the ingredients together, the convergence velocity of $(1, \lambda)$-ES can be obtained. As for the $(1, \lambda_{\mathrm{m}})$-ES, all the arguments for $(1, \lambda)$-ES above are still valid expect the density function of $z_{\lambda:\lambda}$. In $(1, \lambda_{\mathrm{m}})$-ES, only $\lambda/2$ of the mutation vectors are realized from the distribution and the formula of $p_{\lambda:\lambda}(z)$ above only works for this half of the population. Obviously, we need another equation in the mirrored case.

We derived the largest order statistics of the mirrored Gaussian population of size λ as follows. We denote the cumulative probability distribution of the largest order statistic as $\hat{P}_{\lambda:\lambda}(Z \leq z)$. Suppose for every $z \geq 0$, in order to facilitate condition in $\hat{P}_{\lambda:\lambda}(Z \leq z)$, namely the largest order statistic is less than or equals to z, all the realized mutation points are required to be realized less than or equal to z. In addition, consider, in the final population, that the remaining mirrored mutations are generated by reversing the signs. All the realized mutations also need to be bigger than $-z$, otherwise the mirrored counterpart of one outlier

would be larger than z and fails the condition of largest order statistic. The argument reads,

$$\begin{aligned}\hat{P}_{\lambda:\lambda}(Z \leq z) &= P^{\lambda/2}(-z < Z \leq z) \\ &= \left[\Phi\left(\frac{z}{\sigma}\right) - \Phi\left(-\frac{z}{\sigma}\right)\right]^{\lambda/2} \\ &= \left[2\Phi\left(\frac{z}{\sigma}\right) - 1\right]^{\lambda/2}, \quad \forall z \geq 0.\end{aligned}$$

Then in case of $z < 0$, the cumulative probability should be always 0. The reason is if a realized mutation is sampled negative, then its mirrored counterpart would be positive. Therefore you could never make the largest order statistics below 0. In total, the cumulative probability function of the largest order statistic is summarized as,

$$\hat{P}_{\lambda:\lambda}(Z \leq z) = \begin{cases} \left[2\Phi\left(\frac{z}{\sigma}\right) - 1\right]^{\lambda/2} & \forall z \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

And its probability density function is

$$\hat{p}_{\lambda:\lambda}(z) = \begin{cases} \lambda p(z)\left[2\Phi\left(\frac{z}{\sigma}\right) - 1\right]^{\lambda/2-1} & \forall z \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

Compared to Equation 3.4, the key difference made by mirrored sampling is the distribution of the order statistics of z. Putting Equations 3.2, 3.4, 3.6 together, the convergence velocity of $(1, \lambda_\mathrm{m})$-ES reads,

$$\begin{aligned}\varphi_{1,\lambda_\mathrm{m}} &= \int_{-\infty}^{\infty} \left(2Rz - \sigma^2 N\right) \hat{p}_{\lambda:\lambda}(z)\,\mathrm{d}z \\ &= 2R\int_0^{\infty} \lambda z p(z)\left[2\Phi\left(\frac{z}{\sigma}\right) - 1\right]^{\lambda/2-1}\mathrm{d}z - \sigma^2 N\int_0^{\infty}\mathrm{d}\left(\left[2\Phi\left(\frac{z}{\sigma}\right) - 1\right]^{\lambda/2}\right) \\ &= 2R\sigma\int_0^{\infty} \lambda z' p(z')[2\Phi(z') - 1]^{\lambda/2-1}\,\mathrm{d}z' - \sigma^2 N\left[2\Phi\left(\frac{z}{\sigma}\right) - 1\right]^{\lambda/2}\Big|_0^{\infty} \quad (\text{let } z = \sigma z') \\ &= 2R\sigma c_{1,\lambda_\mathrm{m}} - \sigma^2 N,\end{aligned}$$

where $z' \sim \mathcal{N}(0,1)$ and c_{1,λ_m} is the expectation of the largest order statistic of a mirrored standard normal population and is well-known as *progress coefficient*, having the specific form,

$$c_{1,\lambda_\mathrm{m}} = \frac{\lambda}{\sqrt{2\pi}}\int_0^{\infty} te^{-\frac{t^2}{2}}[2\Phi(t) - 1]^{\lambda/2-1}\,\mathrm{d}t. \tag{3.7}$$

By the usage of normalized quantities

$$\varphi^* = \frac{N}{2R^2}\varphi \quad \text{and} \quad \sigma^* = \frac{N}{R}\sigma,$$

the final normalized form of convergence velocity of $(1, \lambda_\mathrm{m})$-ES reads,

$$\varphi^*_{1,\lambda_\mathrm{m}} = c_{1,\lambda_\mathrm{m}}\sigma^* - \frac{(\sigma^*)^2}{2} \tag{3.8}$$

This form is already comparable to the progress rate of $(1, \lambda)$-ES, which is given in the equation below [9].

$$\varphi^*_{1,\lambda} = c_{1,\lambda}\sigma^* - \frac{(\sigma^*)^2}{2} \tag{3.9}$$

Note that the only difference to Equation 3.8 is the progress coefficient which is the expectation of the largest order statistic of a standard normal population,

$$c_{1,\lambda} = \frac{\lambda}{\sqrt{2\pi}} \int_{-\infty}^{\infty} te^{-\frac{t^2}{2}} \Phi(t)^{\lambda-1} \, \mathrm{d}t. \tag{3.10}$$

In the convergence velocity formula, the progress coefficient acts as the "gain" while the "loss" term $-(\sigma^*)^2/2$ is the same in Equation 3.8 and 3.9. Therefore, the comparison of the convergence velocities can be replaced by the comparison of the progress coefficients. The analytical treatments of both Equation 3.7 and 3.10 are difficult so that we just plot the two progress coefficients as a function of λ numerically in Figure 3.4. Both two progress coefficients are slowly increasing as λ increases.

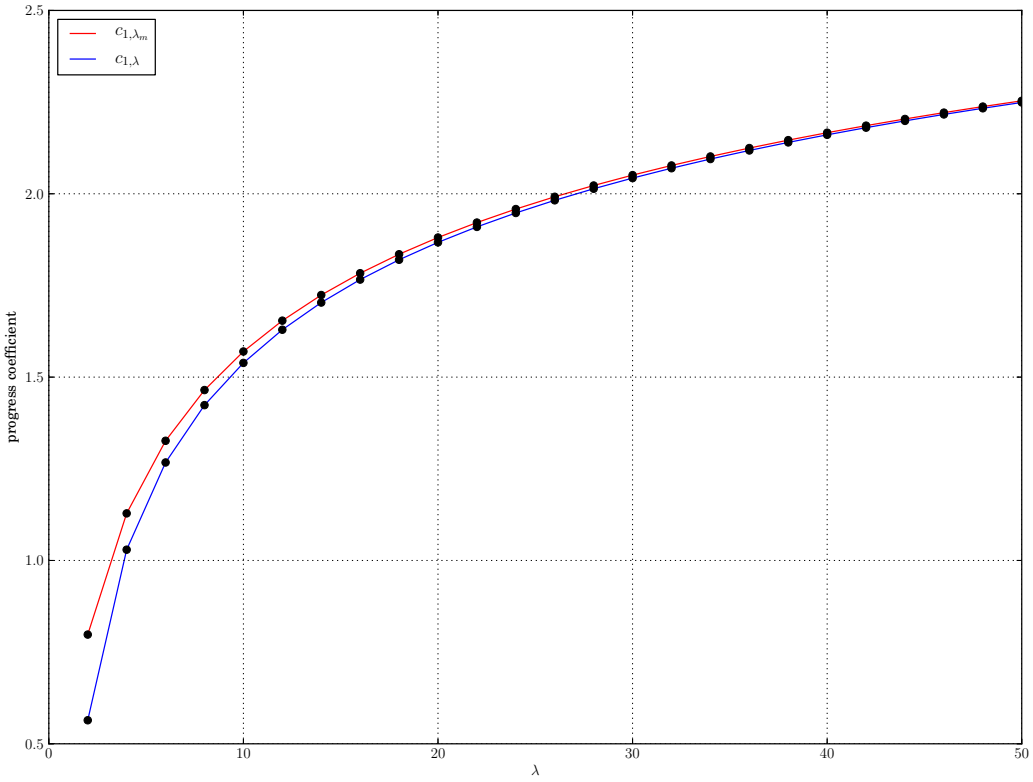

Figure 3.4: Comparison of the progress coefficients.

Due to the investigation by Beyer [9], The progress coefficient $c_{1,\lambda}$ can be approximated

by,

$$c_{1,\lambda} \sim \sqrt{2 \ln \lambda}$$

However, there is no approximation for $c_{1,\lambda_{\mathrm{m}}}$ currently. From the numerical comparison, $c_{1,\lambda_{\mathrm{m}}}$ is obviously better than $c_{1,\lambda}$ if λ is small. When λ is larger than 30, the advantage of $c_{1,\lambda_{\mathrm{m}}}$ begins to vanish and finally these two coefficients become roughly the same after 40. This theoretical result actually coincides with our argument of mirrored sampling in Section 3.1. In summary, the mirrored sampling technique changes the distribution of the largest order statistic of standard normal population, leading to a improvement on the progress coefficient under small population. When the population size is greater than 30, the improvement of mirrored sampling can be neglected. In addition, the fast evolution search (e.g. CMA-ES) always exploits small populations. The mirrored sampling could bring a satisfactory improvement on the performance of those fast searches.

3.3 Mirrored sampling and recombination

Despite the great success of $(1, \lambda_{\mathrm{m}})$, the mirrored sampling has encountered a critical bottleneck when applied to $(\mu/\mu_{\mathrm{w}}, \lambda)$-ES algorithm where the μ best out of the λ offspring are used to compute the new search point via weighted recombination. The direct application in such algorithm always results in an undesired bias on the step-size. In recombination, a mirrored pair of offspring could be selected (it is possible specially in the multi-modal landscape) and neutralize each other so that their contribution to the recombined search point is either none (equal weights) or smaller than what we expected (unequal weights). As one design principle of ES, unbiasedness dictates that ES states should remain unchanged in expectation under random selection, which is reasonable because random selection means the search space is identical everywhere so that the best evolving decision should be stay where we are. However, the behavior of mirrored pair in recombination is undesirable because it leads to a systematic reduction of the recombination *variance* under random selection (random fitness, equivalently). To see this, consider the simplest case: we have a population of mutations $\{\mathbf{z}_i\}_{1 \le i \le \lambda}$ where $\mathbf{z}_i \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$ and the equal weights are used in recombination. Under random selection, the distribution of selected mutations are still normal so that the recombined mutation $\langle \mathbf{z} \rangle$ is still normally distributed as,

$$\langle \mathbf{z} \rangle = \frac{1}{\mu} \sum_{i=1}^{\mu} \mathbf{z}_{i:\lambda} \sim \frac{1}{\mu} \boldsymbol{\mathcal{N}}(\mathbf{0}, \mu^2 \mathbf{I}) \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$$

Then in the mirrored case, if one pair of mirrored mutations are selected, then such pair would disappear in the summation above. The recombined mutation $\langle \mathbf{z}_m \rangle$ is then distributed as follows:

$$\langle \mathbf{z}_m \rangle = \frac{1}{\mu} \sum_{i=1}^{\mu-2} \mathbf{z}_{i:\lambda} \sim \boldsymbol{\mathcal{N}} \left(\mathbf{0}, \left(1 - \frac{2}{\mu} \right)^2 \mathbf{I} \right)$$

It is now obvious to see that the variance of recombined mutation is reduced under random selection. The more pairs of mirrored mutations are selected, the more undesirable bias we would have. In addition, if mirrored sampling is implemented into $(\mu/\mu_w, \lambda)$-CMA-ES or any other ES algorithm using CSA technique, there is also a reduction of the expected step-size under random selection due to the pairwise cancellation in recombination, as was shown

in [15, Fig. 4].

To fix this unsatisfactory effect, two further heuristics are introduced in [6]: *pairwise selection* and *selective mirroring*. Pairwise selection allows only the better one of a mirrored offspring pair to possibly contribute to the weighted recombination such that the pairwise cancellation during recombination is avoided.

Another improvement, selective mirroring sets the number of the realized offspring to λ_{iid} and the number of mirrored offspring to λ_{m}. The mirrored offspring are created from the λ_{m} worst realized offspring. The parameter λ_{iid} is larger than λ_{m} and they are both exogenous strategy parameters. The idea is that: First, if we limit the number of mirrored pairs, the probability of recombining such pair is decreased. Second, the best of λ_{iid} offspring is not expected to be improved by mirrored sampling on fitness landscapes with convex contours. In such case, the realized offspring and its mirrored counterparts can not be better than the parent at the same time. Thus, mirroring the λ_{m} worst offspring from λ_{iid} seems a good design. The detailed discussion and analysis of those two improvement of mirrored sampling are presented in [6].

3.4 The "noisy" mirrored sampling technique

As to solve the bottleneck problem in the last section, we also try to make the mirroring work with recombination with our new variant, the "noisy" mirrored sampling. The idea is really simple: Due to the unpleasant behaviour of mirroring actually is the possible pairwise cancellation during recombination, if we rotate the mirrored vector a little bit, a mirrored pair is not cancelling each other any longer. The idea looks like there is a noise interfering the mirroring process. That is why it is named "noisy" mirroring and we use the notation $(\mu, \tilde{\lambda}_{\text{m}})$-ES to denoted such ES variant. However, the amount of the rotation is really critical here. The rotations could neither be too small so that there is no difference with mirroring, or be too large so that a possible "good" mirrored offspring is rotated out of the progress region. The information on high dimensional progress region depends on many facts and can not be inferred from the fitness. Therefore, it seems there is no reasonable basis on which the amount of rotation is determined. Instead, we try to generate small random rotations.

The rotation generation is exactly the same approach used in (μ, λ)-MSC-ES to generate correlated mutations. The vector rotation in N-dimensional space can be represented by $N(N-1)/2$ rotation angles corresponding to $N(N-1)/2$ 2-dimensional subspaces. The rotation matrix R_{ij} for a rotation angle α_{ij} between axis i and j, is given by an identity matrix, extended by the entries $R(i,i) = R(j,j) = cos\alpha_{ij}$ and $R(i,j) = -R(j,i) = -\sin\alpha_{ij}$. Then the overall rotation matrix, or noise matrix C for the vector is simply the multiplication of all $N(N-1)/2$ R_{ij} matrices. The rotation angles is generated by uniform distribution over $[-1,1]$ rescaled by the rotation range η. This rotation generation procedure is shown in Algorithm 3.3 named NOISE-GENERATION(N).

For practical purpose, the computational complexity is also a import criterion of ES algorithm. However, the "noisy" mirrored sampling method has to call the NOISE-GENERATION(N) procedure which normally makes $N(N-1)/2-1$ matrix multiplication of size N, at least once a generation. In order to reduce such overheads, a very low change rate p_m is exploited to control the process. Before the generation of each rotation angle, a uniform number a over $[0,1]$ is generated and if $a < p_m$, the rotation angle is generated, otherwise this rotation angle is skipped so that one matrix multiplication is saved (check lines 6 and 7 in Algorithm 3.3).

Algorithm 3.3 NOISE-GENERATION(N)

$\eta \leftarrow \pi/\sqrt{N/2}$
$p_m \leftarrow 0.2$
$\mathbf{M} \leftarrow$ IDENTITY-MATRIX(N)
for $i = 1 \rightarrow N - 1$ **do**
  for $j = i + 1 \rightarrow N$ **do**
    $a \leftarrow \mathcal{U}(0, 1)$
    if $a < p_m$ **then**
      $z \leftarrow \eta \cdot \mathcal{U}(-1, 1)$
      $\mathbf{R} \leftarrow$ IDENTITY-MATRIX(N)
      $\mathbf{R}_{ii} \leftarrow \cos z$
      $\mathbf{R}_{jj} \leftarrow \mathbf{R}_{ii}$
      $\mathbf{R}_{ij} \leftarrow -\sin z$
      $\mathbf{R}_{ji} \leftarrow -\mathbf{R}_{ij}$
      $\mathbf{M} \leftarrow \mathbf{RM}$
    end if
  end for
end for
return $\mathbf{M}$

Algorithm 3.4 NOISY-MIRRORED-SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$

given: $\mathbf{z}_i, \mathbf{x}_i \in \mathbb{R}^d$
$\mathbf{B}, \mathbf{D} \leftarrow$ EIGEN-DECOMPOSITION$(\mathbf{C})$
$\mathbf{N} \leftarrow$ NOISE-GENERATION(d)
if $\lambda \neq 0 \pmod 2$ **and** $\mathbf{z}_{\text{last}}$ is present **then**
  $\mathbf{x}_1 \leftarrow \mathbf{x} - \sigma\mathbf{BDN}\mathbf{z}_{\text{last}}$
  $\lambda \leftarrow \lambda - 1$
  Delete static variable $\mathbf{z}_{\text{last}}$.
end if
for $i = 1 \rightarrow \lambda$ **do**
  if $i \equiv 0 \pmod 2$ **then**
    $\mathbf{x}_i \leftarrow \mathbf{x} - \sigma\mathbf{BDN}\mathbf{z}_{i-1}$
  else
    $\mathbf{z}_i \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x} + \sigma\mathbf{BD}\mathbf{z}_i$
  end if
end for
if $\lambda \neq 0 \pmod 2$ **then**
  Create static variable $\mathbf{z}_{\text{last}}$.
  $\mathbf{z}_{\text{last}} \leftarrow \mathbf{z}_\lambda$
end if

Finally, The "noisy" mirrored sampling algorithm is given in Algorithm 3.4 to show how to use the "noise" matrix. In case the covariance matrix or correlated mutation is incorporated, the key operation order is that we should first generate standard normal mutations,

then noisily mirror them, finally rescale and rotate them by the covariance matrix (check lines 8 and 12 in Algorithm 3.4). The reason is that any rotation matrix only makes the end points of vectors transformed on hypersphere. If we first generate the correlated mutations, they are essentially on a hyper-ellipsoid which is not suitable for a rotation operation.

Prior to the benchmark results of performance in Chapter 5, we would like to verify if our new variant actually solve the problem in the last section. After both the mirrored sampling and "noisy" mirrored sampling method are implemented into CMA-ES, we test both $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES and $(\mu/\mu_{\mathrm{w}}, \tilde{\lambda}_{\mathrm{m}})$-CMA-ES on random fitness function and plot the logarithm of the global step-size σ against each other in Figure 3.5.

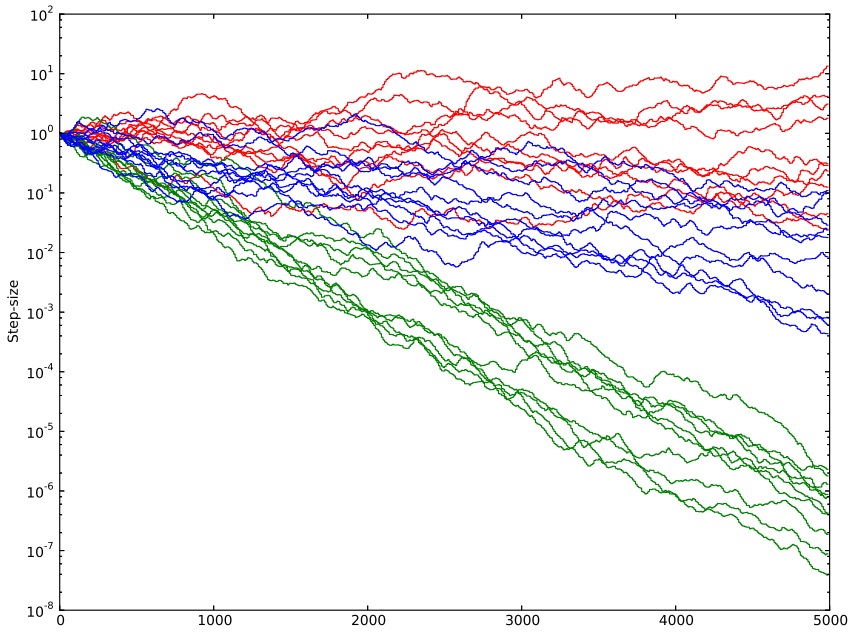

Figure 3.5: Step-size σ versus number of function evaluations of 30 runs on a purely random fitness function in dimension 10. x-axis: evaluation counts.

The random walk of $\log \sigma$ of CMA-ES is shown in 10 red curves. Another 10 green curves show the situation in $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES and reveal a strong bias. Such strong bias is rectified for some degree in the last ten blue curves which are for the $(\mu/\mu_{\mathrm{w}}, \tilde{\lambda}_{\mathrm{m}})$-CMA-ES. In addition, the experiment also shows the degree of such rectification is controlled by parameters η and a. $\eta = \pi/\sqrt{N/2}, a = 0.2$ are used to generate Figure 3.5 and should be manually tuned to maximize the ES performance. As explained in the last section and in [6] before, the expected behaviour of σ remain unchanged in expectation. The more bias observed in σ, the more undesirable recombination we would get. Thus our new method has certain degree of correction of the biases. The detailed performance measurement will be done in Chapter 5.

Chapter 4

Derandomized sampling

The mirroring intuition in the last chapter, which distributes random samples much evenly, is a special case of the much more general concept: *derandomized sampling*. The task is the same as mirrored sampling while the approaches are more general and diverse. Much of the current progresses on derandomized sampling are based on quasi-random-search in which the quasi-random numbers are used in the evolution, such as the application in GA (genetic algorithm) [28] and in CMA-ES [39]. These techniques usually take some complicated methods for quasi-random number generation and will not be discussed here. The goal of this chapter is to introduce two new intuition to generate the derandomized samples by simply manipulating the normal random numbers. The starting point of everything in this chapter is based on the following consideration: The functionality of search space exploration in evolution strategy can be separated to,

1. Exploration of a good local direction to move the parent in $\mathbb{R}^n$ search space, namely the adjustment of the eigenvectors of covariance matrix.

2. Exploration of a good local step-size to move the parent in $\mathbb{R}^n$ search space, namely the tuning of eigenvalues and the global step-size.

Intuitively, the sampling (mutation) process can be separated into two components, the determination of the direction and determination of step-size.

4.1 Direction derandomization

We first concern the direction aspect in sampling. The previous derandomized sampling method, mirrored sampling renders the directions of half of the mutations (mirrored vectors) opposite to the other half (realized normal vectors). It is trying to avoid all the mutations in one generation roughly head to the similar direction, by derandomizing the direction of half the mutations. As the consequence of such direction derandomization process, the mutations are distributed in the $\mathbb{R}^n$ Euclidean space much more evenly so that the exploration of search space is seemingly much better. Despite its success, the direction of the realized mutations is still completely random while the direction of the mirrored mutations are too deterministic.

4.1.1 A measure on mutation direction

In the effort to compare the strategy parameter adaptation mechanisms, lots of measures or principles (unbiasedness e.g.) are defined to judge the goodness. Following the previous research method, it is very important to establish a quality measure of the mutation directions before any discussion or improvement on the direction derandomization method. Such measure would make any comparison based on quantities instead of blurred intuition. The effect of the derandomized direction could be measured as the mean of the angles formed by each pair of the mutation vectors. Let angle ϕ_{ij} represents the angle formed by mutation vector $\mathbf{x}_i$ and $\mathbf{x}_j$, where

$$\cos\phi_{ij} = \frac{\mathbf{x}_i^T \mathbf{x}_j}{||\mathbf{x}_i||\,||\mathbf{x}_j||}.$$

Then the measure of the degree of derandomized direction read,

$$M(P) = \frac{1}{\lambda}\sum_{i=1}^{\lambda-1}\sum_{j=i+1}^{\lambda}\frac{\mathbf{x}_i^T \mathbf{x}_j}{||\mathbf{x}_i||\,||\mathbf{x}_j||}, \quad \mathbf{x}_i, \mathbf{x}_j \in P. \tag{4.1}$$

Not that the P is a population of realized mutations and therefore is not a measure on the expected direction derandomization ability of a specific distribution. This measure would be meaningless if we take the expectation. Consider that the multivariate Gaussian distribution is used in the sampling. Then the expected measure would be

$$\begin{aligned}
\mathrm{E}[M] &= \mathrm{E}\left[\frac{1}{\lambda}\sum_{i=1}^{\lambda-1}\sum_{j=i+1}^{\lambda}\frac{\mathbf{x}_i^T \mathbf{x}_j}{||\mathbf{x}_i||\,||\mathbf{x}_j||}\right] \\
&\approx \frac{1}{\lambda}\sum_{i=1}^{\lambda-1}\sum_{j=i+1}^{\lambda}\frac{1}{||\mathbf{x}_i||\,||\mathbf{x}_j||}\,\mathrm{E}\left[\sum_{k=1}^{N} x_{ik}x_{jk}\right] \\
&= 0
\end{aligned}$$

Because the measure is the mean cos values, the result 0 above means the mutation vector are approximately perpendicular to each other in expectation, which is the basic property of Gaussian distribution. Therefore, M should be applied to realized mutations to observe the worst case that a certain sampling technique would lead to.

During the execution of an ES algorithm, we could calculate such measure generation by generation and record the worst value ever happened. After multiple trials, the worst measure can be averaged, yielding a quality measure on how bad a mutation operator would be in the sense of direction derandomization.

For a population of realized mutations, we examine the worst measure value M_w a mutation operator would give. Then the closer such worst case is to 0, the more desirable the mutation vectors are arranged. If $M_w = 0$, the mutations are guaranteed to be orthogonal and thus sparse in the search space. such mutations are considered as *completely derandomized*. If $M_w > 0$, then the mutations are supposed to roughly concentrate to the same direction and is named as *under derandomized* or *correlated*. The closer M_w to 1, the are heading to the similar direction. If the measure is negative, then most of the angles formed by mutations are greater than $\frac{\pi}{2}$, which indicates that the mutations are approximately anti-parallel to each other (consider the mirrored sampling). This situation is called *over*

derandomized.

For example, the situation and degree of direction derandomization are explored by the measure above. The expected worst measure for CMA-ES and its mirrored version are computed by 15 trials each and compared in Figure 4.1

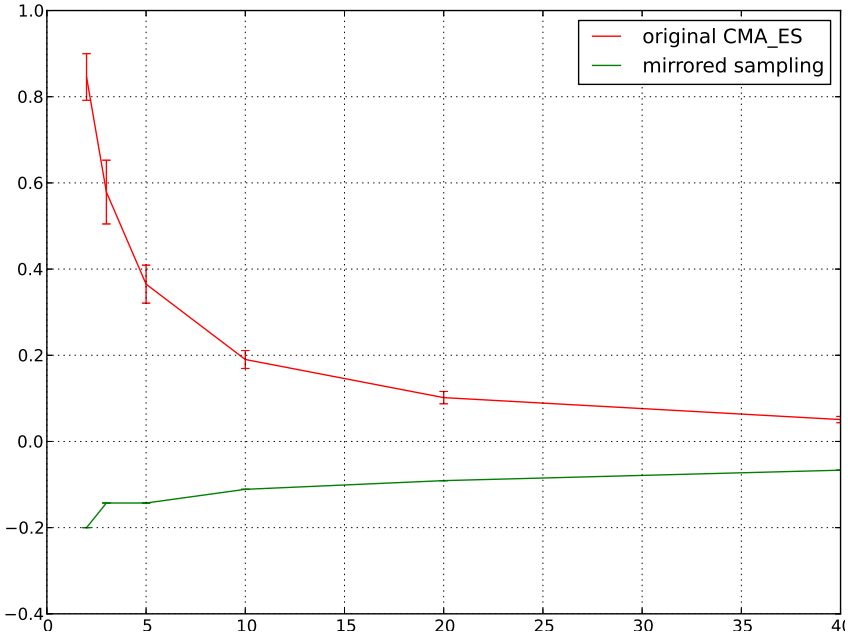

Figure 4.1: Step-size σ versus number of function evaluations of 30 runs on a purely random fitness function in dimensions $2, 3, 5, 10, 20, 40$. x-axis: evaluations.

From the figure, the value of mirrored sampling method (the green curve) is always negative. This is because half of the mutations are anti-parallel to another half, which contributes $\frac{\lambda}{2} - 1$s to the measure calculation and therefore reduce the worst measure.

4.1.2 A completely derandomized direction approach

As a naive investigation, the author is thinking whether it is possible to distribute the directions of mirrored mutations more evenly. A possible idea can be formalized as follows. If we completely derandomize the direction, in other words, prescribe a rule to setting the directions of all λ mutations in a deterministic may, the measure on such mutations would be very good. A native idea is to align the mutation vectors with the *eigenvectors* of the covariance matrix $\mathbf{C}$. As the eigendecomposition is usually needed in the derandomized ES, such approach seems quite cheap in computational cost.

The eigendecomposition of covariance matrix $\mathbf{C}$ reads,

$$\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^T, \quad \mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_N]$$

where $\mathbf{b}_1, \ldots, \mathbf{b}_N$ are N eigenvectors, representing the principal axises of the hyper ellipsoid contour of the distribution. The directions of the N mutation vectors are exactly the

directions of eigenvectors. First we assign the N mutation vectors to the N canonical bases,

$$[\mathbf{s}_1, \ldots, \mathbf{s}_N] = \mathbf{I}_{N \times N},$$

$\mathbf{I}_{N \times N}$ is a $N \times N$ identity matrix. Then N χ-distributed random numbers (its degree of freedom is N) are generated and used to rescale N mutations,

$$\mathbf{z}_i = a_i \eta_i \mathbf{s}_i, \quad a_i \sim \mathcal{U}, \quad \eta_i \sim \chi(N), \quad 1 \leq i \leq N$$

Note that a_i is sampled from a discrete uniform distribution on points $-1, 1$ and used to make possible reverse of the direction of the vector. After this operation, $\mathbf{z}_i$ is again normally distributed due to the argument in Section 4.3.1. Finally, we use matrices $\mathbf{D}$ and $\mathbf{B}$ to rescale and rotate the mutations so that they are $\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{C})$ distributed and aligned with the eigenvectors.

$$\mathbf{s}_i = \mathbf{B}\mathbf{D}\mathbf{z}_i, \quad 1 \leq i \leq N$$

The procedure works if $\lambda \leq N$. If the population is larger than dimensionality, which is the common case, the rest $\lambda - N$ offspring are generated using the standard sampling method as dictated in Algorithm 3.1. Because there are N mutations whose directions are completely determined by the eigenvector of $\mathbf{C}$, we named such idea as *completely deradomized direction* sampling. The pseudo code of this idea is shown in Algorithm 4.1 below.

The benefits of such idea is that there are N pairs of the mutation vectors orthogonal to each other due to the orthogonality of the eigenvectors. The measure M for any mutations generated in this manner is really low because all there are $N(N-1)/2$ summands in Equation 4.1 equal to 0. The exploration of the space is very diverse.

The disadvantage of this idea seems dominating its performance. After a naive implementation into CMA-ES, the completely derandomized direction is compared to the standard CMA-ES on several fitness functions. The new method requires more evaluations than the standard CMA-ES and therefore decreases the convergence velocity. The reason may be that we are derandomizing too much from the sampling procedure and the corresponding mutations are too "neutral".

Algorithm 4.1 COMPLETE-DERANDOMIZED-DIRECTION-SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$

given: $\mathbf{z}_i, \mathbf{x}_i \in \mathbb{R}^d, s_i \in \mathbb{R}$
$\mathbf{B}, \mathbf{D} \leftarrow$ EIGEN-DECOMPOSITION$(\mathbf{C})$
$[\mathbf{s}_1, \ldots, \mathbf{s}_d] \leftarrow \mathbf{I}_{d \times d}$
for $i = 1 \rightarrow \lambda$ **do**
  if $i \leq d$ **then**
    $a_i \leftarrow \mathcal{U}, \eta_i \leftarrow \chi(d)$
    $\mathbf{z}_i \leftarrow a_i \eta_i \mathbf{s}_i$
  else
    $\mathbf{z}_i \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$
  end if
  $\mathbf{x}_i \leftarrow \mathbf{x} + \sigma \mathbf{B}\mathbf{D}\mathbf{z}_i$
end for

4.1.3 Orthogonal Sampling

The completely derandomized direction sampling method seems removing too much randomnesses and its performance is getting worse consequently. The N mutation vectors are aligned with N eigenvectors of the covariance matrix $\mathbf{C}$. Such mutation generation procedure renders the directions of N mutation vectors deterministic and prescribed by the covariance $\mathbf{C}$ uniquely. This would imply that we fully trust the direction information encoded in $\mathbf{C}$ for mutation. There are mainly two reasons why such scheme does not work:

1. Due to the noisy adaptation process (CMA, MSC), the covariance matrix $\mathbf{C}$ can not approximate the local landscape (by the convex quadratic form $\frac{1}{2}\mathbf{x}^T\mathbf{C}\mathbf{x}$) as accurate as its maximal ability. Thus, the directions indicated by the covariance matrix are in no way reliable.

2. There is no effective exploration of directions even if the covariance matrix were accurate. No new information on good mutation directions can be to the covariance matrix, which aggravates the covariance adaptation mechanism. The direction randomness of mutation vector is useful and should not be removed completely.

Although the completely derandomized direction method is not a successful attempt to direction derandomization, its intuition, generating *orthogonal mutation vectors*, could be promising for our further development. The problem of completely derandomized direction approach is that the orthogonal mutations generated are totally deterministic in direction. Therefore, the solution is to add some randomness in direction to all the mutations uniformly, which leads to a random *orthogonal mutation operator*.

The first method to add the exploration effects back is simply rotating all the orthogonal mutations under a randomly generated rotation matrix. The random rotation matrix $\mathbf{R}$ is obtained by the NOISE-GENERATION procedure which is formalized in NOISY-MIRRORED sampling technique. The steps of the new method are roughly the same as that of completely derandomized direction sampling, except that after generating an $N \times N$ identity matrix $\mathbf{I}$, we multiply it by the random rotation matrix $\mathbf{R}$,

$$\mathbf{S} = \mathbf{R}\mathbf{I}_{N\times N} = [\mathbf{R}\mathbf{e}_1, ..., \mathbf{R}\mathbf{e}_N] = [\mathbf{s}_1, \ldots, \mathbf{s}_N].$$

Note the $\mathbf{e}_1, \ldots, \mathbf{e}_N$ are N canonical basis of the N-dimensional space. The random rotation matrix $\mathbf{R}$ actually randomizes the directions of the orthogonal mutations uniformly. Hereon, the direction derandomization between mutation vectors and the directional randomness of all the mutations are achieved uniformly. Such method is named as *orthogonal sampling*.

The method above only works if the population size $\lambda = N$. If $\lambda \geq N$, the rest $\lambda - N$ mutations are sampled directly from the Gaussian distribution. If $\lambda \leq N$, we pick λ columns (rotated bases) from $\mathbf{S}$ uniformly as our samples. We can achieve this by generating a discrete uniform random number $a \sim \mathcal{U}(1, N)$[1] on the set $\{1, 2, \ldots, N\}$ at the beginning. Then the ath column of $\mathbf{S}$ is selected as our first sample and removed from $\mathbf{S}$. The rest $N-1$ column vectors are reassigned with index $\{1, 2, \ldots, N-1\}$. After that a is generated again from $\mathcal{U}(0, N-1)$ and used to select and remove the ath column from $\mathbf{S}$. We just need to repeat this cycle until λ mutations are obtained. The details of this approach is listed in Algorithm 4.2.

[1]Here the notation $\mathcal{U}(a, b)$ is used to represent the discrete uniform distribution on set $\{a, a+1, \ldots, b\}$. It may conflict with the notation of continuous uniform distribution in other literatures.

Algorithm 4.2 ORTHOGONAL-SAMPLING1($\mathbf{x}, \sigma, \mathbf{C}, \lambda$)

1 **given**: $\mathbf{z}_i, \mathbf{x}_i \in \mathbb{R}^d, s_i \in \mathbb{R}$
2 $\mathbf{B}, \mathbf{D} \leftarrow$ EIGEN-DECOMPOSITION($\mathbf{C}$)
3 $\mathbf{R} \leftarrow$ NOISE-GENERATION(d)
4 $\mathbf{S} \leftarrow [\mathbf{s}_1, \ldots, \mathbf{s}_d] \leftarrow \mathbf{R}\mathbf{I}_{d\times d}$
5 **for** $i = 1 \rightarrow \lambda$ **do**
6 **if** $i \leq d$ **then**
7 $a_i \leftarrow \mathcal{U}, \eta_i \leftarrow \chi(d), u \leftarrow \mathcal{U}(1, d-i+1)$
8 $\mathbf{z}_i \leftarrow a_i \eta_i \mathbf{s}_u$
9 Remove $\mathbf{S}_u$ from $\mathbf{S}$ and reassign the index $\{1, 2, \ldots, d-i\}$ to the rest $d-i$vectors.
10 **else**
11 $\mathbf{z}_i \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$
12 **end if**
13 $\mathbf{x}_i \leftarrow \mathbf{x} + \sigma \mathbf{B}\mathbf{D}\mathbf{z}_i$
14 **end for**

Note that $\mathcal{U}$ still denotes the discrete uniform distribution on set $\{-1, 1\}$ as in the previous section. When $\lambda > N$, pure random mutation samples are taken as dictated in Line 10 above.The drawbacks of this algorithm is that the parameters η, a of NOISE-GENERATION procedure (see Section 3.4) is yet to be determined. After a linear search based on sphere function, we found that $a = 1, \eta = \frac{\pi\sqrt{8N}}{30}$ could be a reasonable parameter setting, which will be an standard setting in the later testing.

Our second attempt to generate the random orthogonal mutations is to exploit the Gram-Schmidt process [14]. The Gram-Schmidt process is a method for orthonormalising a set of vectors in an inner product space, most commonly the Euclidean space $\mathbb{R}^N$. The Gram-Schmidt process takes a finite, linearly independent set $\mathbf{S} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ for $k \leq N$ and generates an orthogonal set $\mathbf{S}' = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ that spans the same k-dimensional subspace of $\mathbb{R}^N$ as $\mathbf{S}$. The pseudo code is listed below.

Algorithm 4.3 GRAM-SCHMIDT($\mathbf{v}_1, \ldots, \mathbf{v}_k$)

1 **for** $i = 2 \rightarrow \lambda$ **do**
2 **for** $i = 1 \rightarrow i - 1$ **do**
3 $\mathbf{v}_i \leftarrow \mathbf{v}_i - \mathbf{v}_j \cdot \langle \mathbf{v}_i, \mathbf{v}_j \rangle / ||\mathbf{v}_j||$ ($\langle ., . \rangle$ denotes the vector inner product)
4 **end for**
5 **end for**
6 **for** $i = 1 \rightarrow \lambda$ **do**
7 $\mathbf{v}_i \leftarrow \mathbf{v}_i / ||\mathbf{v}_i||$
8 **end for**

Based on the functionality of Gram-Schmidt process, if we first sample λ mutation vectors,

$$\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_\lambda], \mathbf{s}_i \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}),$$

then, processing S by Gram-Schmidt process would give us a collection of random orthonormal vectors,

$$[\mathbf{s}'_1, \ldots, \mathbf{s}'_\lambda] = \text{GRAM-SCHMIDT}(\mathbf{S}).$$

Note that each vector of $\mathbf{s}'_1, \ldots, \mathbf{s}'_\lambda$ is of unit length and orthogonal to the rest. Finally, rescaling the length of $\mathbf{s}_i$ by χ random number renders them normally distributed again, as we we have done in completely direction derandomization. The detailed algorithm is shown in Algorithm 4.4.

Algorithm 4.4 ORTHOGONAL-SAMPLING2($\mathbf{x}, \sigma, \mathbf{C}, \lambda$)

given: $\mathbf{z}_i, \mathbf{x}_i \in \mathbb{R}^d, s_i \in \mathbb{R}$
$\mathbf{B}, \mathbf{D} \leftarrow$ EIGEN-DECOMPOSITION($\mathbf{C}$)
for $i = 1 \rightarrow \lambda$ **do**
  $\mathbf{s}_i \leftarrow \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$
end for
$[\mathbf{s}'_1, \ldots, \mathbf{s}'_\lambda] =$ GRAM-SCHMIDT($\mathbf{s}_1, \ldots, \mathbf{s}_\lambda$).
for $i = 1 \rightarrow \lambda$ **do**
  $a_i \leftarrow \mathcal{U}, \eta_i \leftarrow \chi(d)$
  $\mathbf{z}_i \leftarrow a_i \eta_i \mathbf{s}'_i$
  $\mathbf{x}_i \leftarrow \mathbf{x} + \sigma \mathbf{B}\mathbf{D}\mathbf{z}_i$
end for

This alternative orthogonal sampling method, Algorithm 4.4 is theoretically the same with Algorithm 4.2. However, it has same advantages. First, it exploits the Gram-Schmidt process which is much faster than the noise-generation procedure used in Algorithm 4.2 and therefore reduce much time complexity. Second, There is no additional parameters needed in this method. We will denote the CMA-ES with ORTHOGONAL-SAMPLING1 (ORTHOGONAL-SAMPLING2) as Orthogonal1-$(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES (Orthogonal2-$(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES).The empirical results of both of the orthogonal sampling methods can be found in Chapter 6.

4.2 Step-size derandomization

4.2.1 Motivation

The opposite face of the our coin would be: derandomization of the step-size of mutation vectors. The goal of performing step-size derandomization is to render the length of mutation vector deterministic. The motivation actually comes from an observation of the *covariance matrix adaptation* technique. In the standard $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES, a decreasing weights is used to given more "trust" to the offspring having better fitness. It is not only used in recombination operator but also in covariance matrix estimation. Adding the weight effect into Equation 2.7, the covariance matrix estimation read [25],

$$\mathbf{C}_{\mu}^{(g+1)} = \sum_{i=1}^{\mu} w_i \left(\mathbf{x}_{i:\lambda}^{(g)} - \mathbf{x}^{(g)}\right)\left(\mathbf{x}_{i:\lambda}^{(g)} - \mathbf{x}^{(g)}\right)^T,$$

where the weight vector is suggested as [24],

$$w_i = \frac{w'_i}{\sum_{j=1}^{\mu} w'_j}, \quad w'_i = \ln\left(\frac{\lambda+1}{2}\right) - \ln i \quad \text{for } i = 1, \ldots, \mu.$$

Such weights decreases very fast as the rank of offspring increases. Now let's rearrange the estimation equation above. By denoting the mutation vector as $\mathbf{y}_i^{(g)} = \mathbf{x}_i^{(g)} - \mathbf{x}^{(g)}$, we have

$$\mathbf{C}_{\mu}^{(g+1)} = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}^{(g)} \mathbf{y}_{i:\lambda}^{(g)^T}, \tag{4.2}$$

where $\mathbf{y}_{i:\lambda}^{(g)} \sim \sigma^{(g)} \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{C}^{(g)})$ under the random selection. Note that each summand in the formula above defines a the covariance matrix of a *singular* distribution $\mathbf{s}_i$ having the form

$$\mathbf{s}_i \sim w_i \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{y}_{i:\lambda}^{(g)} \mathbf{y}_{i:\lambda}^{(g)^T})$$

Its covariance matrix has rank one, only one eigenvector $\mathbf{y}_{i:\lambda}^{(g)}$ and eigenvalue $\left(w_i||\mathbf{y}_{i:\lambda}^{(g)}||\right)^2$. Therefore, as the sum of all the singular distributions, $\mathbf{C}_{\mu}^{(g+1)}$ has rank μ. Its eigenvectors are $\mathbf{y}_{1:\lambda}^{(g)}, \ldots, \mathbf{y}_{\mu:\lambda}^{(g)}$ and the square root of corresponding eigenvalues are $w_1||\mathbf{y}_{1:\lambda}^{(g)}||, \ldots, w_\mu||\mathbf{y}_{\mu:\lambda}^{(g)}||$.

Therefore, accumulating $\mathbf{C}_{\mu}^{(g+1)}$ to the current matrix $\mathbf{C}^{(g)}$ is tuning its the eigenvectors and eigenvalues towards that of $\mathbf{C}_{\mu}^{(g+1)}$. Recall that the suppose of covariance matrix adaptation is to tune its eigenvectors and eigenvalues so that the longest principal axis of its contours roughly points to the optimum. Then, a critical problem arise here: There is no guarantee that *the better fitness a mutation vector has, the better direction it is heading*. The same fitted point could be reached by combination of a worst direction and a long vector length. Recall again that the length of multivariate Gaussian vector, has expectation and variance as

$$\mathrm{E}[||\boldsymbol{\mathcal{N}}(\mathbf{0}, \sigma^2\mathbf{I})||] \simeq \sigma\sqrt{N}, \quad \mathrm{Var}[||\boldsymbol{\mathcal{N}}(\mathbf{0}, \sigma^2\mathbf{I})||] \simeq \frac{1}{2}\sigma^2,$$

The variance is not vanishing with the increasing dimensionality such that a shorter mutation having a better direction may not outperforms a longer mutation having a worse direction. The effect is somehow undesirable because the better fitted mutation vector is assigned a much bigger weight during the covariance estimation and therefore affects the covariance adaptation more. This observation simply suggests that the covariance matrix estimation may be over fitted and some information is wasted.

4.2.2 The algorithm

As the simplest solution to the problem, we could transform the mutation vector so that they are distributed on the hyper-ellipsoid for sure. In other words, we are trying to eliminate the disturbing variance of the normal vector. This specific method can be formalized as followed. First, λ standard normal vector are sampled

$$\mathbf{s}_i = \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I}), \quad 1 \leq i \leq \lambda$$

Then we normalize and rescale $\mathbf{s}_i$ so that they are now lying uniformly on the hyper sphere having the radius $\mathrm{E}[\chi]$,

$$\mathbf{s}_i' = \mathrm{E}[\chi] \cdot \frac{\mathbf{s}_i}{||\mathbf{s}_i||}$$

Note that the normalization would be invalid if the length if $||\mathbf{s}_i|| = 0$. However, the probability of sampling 0 for all the entries of $\mathbf{s}_i$ is infinitesimal. There is no need to consider that extreme case. Finally, the mutations are transformed again by the eigenvalues and eigenvectors of $\mathbf{C}$,

$$\mathbf{z}_i = \sigma\mathbf{B}\mathbf{D}\mathbf{s}_i'$$

The pseudo-code the derandomized step-size sampling is listed in Algorithm 4.5. The derandomized step-size sampling method above distributed all the mutations uniformly on the hyper-ellipsoid representing the covariance matrix. Because the full randomness of direction is still kept, the direction exploration is not restricted. At this time, a mutation roughly heading to the direction of the longest principal axis of the covariance matrix is guaranteed to be longer than the mutations heading to other directions. Consequently, if the direction of the longest principal axis is not pointing to the optimum, the mutation vectors aligned with

it could perform much worse so that the ill-tuned eigenvectors could be discovered by selection and adjusted by the adaptation mechanism quickly. In total, the potential ambiguity in the selection of mutation directions has been avoided as much as possible.

Algorithm 4.5 DERANDOMIZED-STEPSIZE-SAMPLING$(\mathbf{x}, \sigma, \mathbf{C}, \lambda)$

given: $\mathbf{s}_i, \mathbf{z}_i, \mathbf{y}_i, \mathbf{x}_i \in \mathbb{R}^d$
$\mathbf{B}, \mathbf{D} \leftarrow$ EIGEN-DECOMPOSITION$(\mathbf{C})$
$\beta \leftarrow \mathrm{E}[\chi]$
for $i = 1 \rightarrow \lambda$ **do**
  $\mathbf{s}_i \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_i \leftarrow \beta \cdot \mathbf{s}_i / \|\mathbf{s}_i\|$
  $\mathbf{x}_i \leftarrow \mathbf{x} + \sigma \mathbf{B}\mathbf{D}\mathbf{z}_i$
end for

4.2.3 Limitation

It would be really nice if derandomized step-size sampling could be inserted into advanced ES algorithms (like CMA-ES) to save evaluations. However, the author realizes that implementing this sampling method in CMA-ES is not possible. Furthermore it does not work with any ES algorithm using CSA (cumulative step-size adaptation) as their step-size control mechanism. The reason is deeply rooted in the foundation of CSA technique.

Recall that the key part of CSA is the evolution path cumulation. The evolution path is acting as a indicator of the local landscape encountered currently and historically. By comparing this indicator to the "neutral case", the path length under random selection, the decision on tunning the step-size can be calculated. The mathematical foundation of this procedure is that the sum of normal vector is again a normal vector so that the evolution path cumulation is well-defined.

In our case, due to the normalization of the normal vectors, the result is not normally distributed any more. Instead, they follow the *multiuniform* distribution (see Section 4.3.1), which does not preserve the nice property of normal distribution any more. Thus, the preliminary condition of CSA is not met. Currently, this new sampling method only works with the MSC adaptation mechanism.

4.2.4 Application

Due to the limitations in the last section, the application of derandomized step-size sampling is restricted. The original motivation, which is trying to avoid the undesired cases in covariance adaptation, could not be tested if it we do not apply it into the CMA-ES. Thanks to a variant of CMA-ES, (μ, λ)-CMA-σSA-ES (or CMSA-ES for short) introduced by Beyer in [13], we could verify and test our new sampling method.

The (μ, λ)-CMSA-ES algorithm replace the CSA mechanism in CMA-ES by the mutative-self-adaptation of step-size, just like in the (μ, λ)-MSC-ES algorithm. This algorithm is very suitable for our case because it keeps the covariance matrix adaptation as covariance control while using mutative-self-adaptation of step-size. The experiment on derandomized step-size sampling is in Chapter 5 actually performed on the (μ, λ)-CMSA-ES. For the self-consistency of this thesis, the CMSA-ES is briefly introduced here.

The mutation operator is a mixture of the CMA-ES and the MSC-ES. The offspring $\mathbf{x}_i$ and their step-size σ_i are created from the parent $\bar{\mathbf{x}}$, $\bar{\sigma}$, the matrices $\mathbf{B}$ and $\mathbf{D}$ (from the eigen decomposition of the covariance matrix $\mathbf{C}$) as follows:

$$\begin{aligned}\sigma_i &= \bar{\sigma} \cdot \exp(\tau \mathcal{N}(0,1)) \\ \mathbf{s}_i &= \mathbf{BD}\boldsymbol{\mathcal{N}}(\mathbf{0},\mathbf{I}) \\ \mathbf{x}_i &= \bar{\mathbf{x}} + \sigma_i \mathbf{s}_i\end{aligned}$$

Recombination operator is based on equal weights as in the MSC-ES, applied to selected solution vectors and step-sizes. The covariance matrix adaptation rule looks roughly the same as Equation 2.7,

$$\mathbf{C}' = \left(1 - \frac{1}{\tau_C}\right)\mathbf{C} + \frac{1}{\mu\tau_C}\sum_{i=1}^{\mu} \mathbf{s}_{i:\lambda}\mathbf{s}_{i:\lambda}{}^T$$

where the suggested setting of extraneous parameters are $\tau = \frac{1}{\sqrt{2N}}$ and $\tau_C = 1 + \frac{N(N+1)}{2\mu}$ according to [13]. The pseudo-code is presented in Algorithm 4.6.

Algorithm 4.6 (μ, λ)-CMSA-ES

```
1  Initialize x̄
2  Initialize σ̄
3  C ← I
4  repeat
5     B, D ← EIGEN-DECOMPOSITION(C)
6     for i = 1 → λ do
7        σ_i ← σ̄ · exp(τ N(0,1))
8        s_i ← BD N(0, I)
9        x_i ← x̄ + σ_i s_i
10       f_i ← f(x_i)
11    end for
12    Section the μ best of (x_i σ_i)
13    x̄ ← (1/μ) Σ_{i=1}^{μ} x_{i:λ}
14    σ̄ ← (1/μ) Σ_{i=1}^{μ} σ_{i:λ}
15    C' ← (1 − 1/τ_C) C + 1/(τ_C μ) Σ_{i=1}^{μ} s_{i:λ} s_{i:λ}^T
16 until
```

4.3 Theoretical results on step-size derandomization

In this section, some analytical results on step-size derandomization sampling are gradually established. In order to make a mathematical analysis possible and focus on the comparison of mutation operator, we restrict our investigation to very simple cases, namely simple evolution strategy on simple fitness models, due to the difficulty on manipulating the new distribution. The analysis approach is the same as Section 3.2.

4.3.1 Prerequisite on probability distribution

To facilitate the theoretical analysis of this new method, we must figure out the distribution of the new random variable obtained by normalizing a Gaussian vector. Fortunately, there exists a nice theorem describing the relation between multivariate normal distribution and the new artificial one in [18, 21, 38]. Intuitively, given a multivariate Gaussian vector, $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the resulting vector after the normalization:

$$\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

is uniformly distributed on a unit hyper sphere. This is due to the rotation-invariant property of multivariate Gaussian distribution (Section 3.2.1), which means $\mathbf{x}$ and $\mathbf{Ox}$ should have the same distribution provided $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{O}$ is an orthogonal matrix. Then the multivariate Gaussian samples are "uniformly" distributed in each direction.

We follow the notation in [18], let $\mathbf{u}$ denote a random vector distributed uniformly on the unit sphere surface in $\mathbb{R}^N$ and $\mathcal{O}(N)$ denote the set of $N \times N$ orthogonal matrices. Then it is obvious that $\mathbf{u}$ and $\mathbf{\Gamma u}$ have the same distribution for every $\mathbf{\Gamma} \in \mathcal{O}(n)$, namely rotation-invariant. It has been termed as *multiuniform distribution*.

Then the next theorem (rendered from [18, Theorem 2.2]) gives the relation between the multivariate normal distribution and the multiuniform distribution.

Theorem 1. *Given a $N \times 1$ random vector $\mathbf{x}$ distributed as $\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$, then $\mathbf{x}$ has a stochastic representation*

$$\mathbf{x} = r\mathbf{u}$$

where r is independent of $\mathbf{u}$, and $r \sim \chi_N$ and $\mathbf{u}$ follows multiuniform distribution.

r is the length of the Gaussian vector as discussed before. The density function of it reads,

$$p(r) = \frac{2r^{N-1}e^{-\frac{r^2}{2}}}{2^{N/2}\Gamma(\frac{N}{2})}$$

The normalization in derandomized direction sampling is exactly $\mathbf{x}/r$ and therefore the resulting random variable is multiuniformly distributed. If we denote such random vector as $\mathbf{u} = [u_1, \ldots, u_2]^T$, where $\sum_{i=1}^n u_i^2 = 1$ (distributed on unit sphere), then the joint probability density function of $u_1, u_2, \ldots, u_k, 1 \leq k \leq N$, is given by [18]

$$p(u_1, \ldots, u_k) = \frac{\Gamma(\frac{N}{2})}{\pi^{k/2}\Gamma\left(\frac{N-k}{2}\right)} \left(1 - \sum_{i=1}^{k} u_i^2\right)^{\frac{N-k}{2}-1}$$

where Γ stands for the Gamma function. In addition, marginal density function for any elements of $\mathbf{u}$, u_i, is given as

$$p(u_i) = \frac{\Gamma(\frac{N}{2})}{\sqrt{\pi}\,\Gamma\left(\frac{N-1}{2}\right)} \left(1 - u_i^2\right)^{\frac{N-3}{2}} \tag{4.3}$$

4.3.2 $(1+1)$-ES on Linear Model

Due to the difficulty of manipulating the multiuniform distribution function. The following discussions are based on simple $(1+1)$-ES to investigate whether the new distribution makes a difference to the normal distribution. The first analysis is performed on linear model on which the progress can only be made in one direction. Since the most important condition of the analytical approach in Section 3.2.2, the rotation-invariant property of mutation vector has already been established for multiuniform random variables in the last section, we will apply such analytical approach here.

We first rotate the coordinate system so that the first canonical basis $\mathbf{e}_1$ is heading toward the optimum. Then the progress of the algorithm is uniquely determined by the first component U_1 of the mutation vector $\mathbf{U}$. Consider the rescaling factor $\mathrm{E}[\chi]$, the relation between the mutation vector and the standard multiuniform vector $\mathbf{u}$ reads,

$$\mathbf{U} = \mathrm{E}[\chi]\mathbf{u}, \quad U_1 = \mathrm{E}[\chi]u_1$$

The marginal density of u_1 is given in Equation 4.3. Let's denote $\mathrm{E}[\chi]$ as β. Then the convergence velocity on linear model reads,

$$\begin{aligned}
\varphi_{1+1} &= \mathrm{E}(U_1) \\
&= \sigma\beta\,\mathrm{E}(u_1) \\
&= \sigma\beta \int_0^1 u_1 p(u_1)\,\mathrm{d}u_1 \\
&= \sigma\beta \int_0^1 u_1 \frac{\Gamma(\frac{N}{2})}{\sqrt{\pi}\,\Gamma\left(\frac{N-1}{2}\right)} \left(1-u_1^2\right)^{\frac{N-3}{2}}\,\mathrm{d}u_1 \\
&= -\sigma\beta \frac{\Gamma(\frac{N}{2})}{2\sqrt{\pi}\,\Gamma\left(\frac{N-1}{2}\right)} \int_0^1 \left(1-u_1^2\right)^{\frac{N-3}{2}}\,\mathrm{d}(1-u_1^2) \\
&= -\sigma\beta \frac{\Gamma(\frac{N}{2})}{\sqrt{\pi}(N-1)\Gamma\left(\frac{N-1}{2}\right)} \left(1-u_1^2\right)^{\frac{N-1}{2}} \Big|_0^1 \\
&= \sigma\beta \frac{\Gamma(\frac{N}{2})}{\sqrt{\pi}(N-1)\Gamma\left(\frac{N-1}{2}\right)}.
\end{aligned}$$

Substituting $\beta = \sqrt{2}\Gamma(\frac{N+1}{2})/\Gamma(\frac{N}{2})$ to the equation above, the result can be simplified as,

$$\begin{aligned}
\varphi_{1+1} &= \sqrt{\frac{2}{\pi}}\sigma \frac{\Gamma(\frac{N+1}{2})\Gamma(\frac{N}{2})}{(N-1)\Gamma\left(\frac{N-1}{2}\right)\Gamma(\frac{N}{2})} \\
&= \sqrt{\frac{2}{\pi}}\sigma \frac{\frac{N-1}{2}\Gamma(\frac{N-1}{2})}{(N-1)\Gamma\left(\frac{N-1}{2}\right)} \\
&= \frac{\sigma}{\sqrt{2\pi}}
\end{aligned} \tag{4.4}$$

Note that the range of u_1 is $[-1, 1]$ due to its distribution definition and therefore the upper limit of the integral above is 1 instead of infinity. The lower limit is 0 because we are dealing with "+" strategy. Such simplified result can be compared to the classic derivation

of Gaussian mutation on linear model,

$$\begin{aligned}\varphi'_{1+1} &= \mathrm{E}(Z_1) \\ &= \sigma\,\mathrm{E}(z_1) \\ &= \sigma\int_0^\infty z_1\phi(z_1)\,\mathrm{d}z_1 \\ &= \frac{\sigma}{\sqrt{2\pi}},\end{aligned} \tag{4.5}$$

where $Z_1 \sim \mathcal{N}(0,\sigma^2)$. The Equation 4.4 and 4.5 are exactly the same. Therefore, on linear function, the multiuniform mutation vector does not make any differences of theoretical performance for $(1+1)$-ES.

4.3.3 $(1+1)$-ES on Sphere Model

The next fitness is the sphere model. The arguments here is basically the same as Section 3.2.2 except that $(1+1)$-ES is the subject. As the first step, we rotate the mutation vector $\mathbf{U}$ (or coordinate system equivalently) such that the first component U_1 is aligned with the direction to the optimum, w. l. o. g. Given the distance from the parent to optimum is R, the distance from the offspring to optimum reads,

$$r^2 = (R-U_1)^2 + (\sigma\beta)^2 - U_1^2 = R^2 + (\sigma\beta)^2 - 2RU_1$$

Using this relation, the convergence velocity of derandomized step-size on sphere model reads,

$$\begin{aligned}\varphi &= \mathrm{E}(R^2 - r^2) = \mathrm{E}(2RU_1 - l^2) \\ &= \mathrm{E}(2R\sigma\beta u_1 - \sigma^2\beta^2) \\ &= 2R\sigma\beta\int_{u_{\min}}^1 u_1 p(u_1)\,\mathrm{d}u_1 - \sigma^2\beta^2\int_{u_{\min}}^1 p(u_1)\,\mathrm{d}u_1 \\ &= 2R\sigma\beta\int_{u_{\min}}^1 u_1\frac{\Gamma(\frac{N}{2})}{\sqrt{\pi}\,\Gamma\left(\frac{N-1}{2}\right)}\left(1-u_1^2\right)^{\frac{N-3}{2}}\,\mathrm{d}u_1 - \sigma^2\beta^2\int_{u_{\min}}^1 \frac{\Gamma(\frac{N}{2})}{\sqrt{\pi}\,\Gamma\left(\frac{N-1}{2}\right)}\left(1-u_1^2\right)^{\frac{N-3}{2}}\,\mathrm{d}u_1 \\ &= -2R\sigma\beta\frac{\Gamma(\frac{N}{2})}{\sqrt{\pi}(N-1)\Gamma\left(\frac{N-1}{2}\right)}\left(1-u_1^2\right)^{\frac{N-1}{2}}\Big|_1^{u_{\min}} - \sigma^2\beta^2\frac{\Gamma(\frac{N}{2})}{\sqrt{\pi}\,\Gamma\left(\frac{N-1}{2}\right)}\int_{\sin^{-1}u_{\min}}^{\frac{\pi}{2}}\cos^{N-2}\alpha\,\mathrm{d}\alpha \\ &= \frac{2R\sigma}{\sqrt{2\pi}}\left(1-\frac{\sigma^2\beta^2}{4R^2}\right)^{\frac{N-1}{2}} - \frac{\sigma^2\beta(N-1)}{\sqrt{2\pi}}\int_{\sin^{-1}u_{\min}}^{\frac{\pi}{2}}\cos^{N-2}\alpha\,\mathrm{d}\alpha \\ &\simeq \frac{2R\sigma}{\sqrt{2\pi}}\left(1-\frac{\sigma^2 N}{4R^2}\right)^{\frac{N-1}{2}} - \frac{\sigma^2\sqrt{N}(N-1)}{\sqrt{2\pi}}\int_{\sin^{-1}u_{\min}}^{\frac{\pi}{2}}\cos^{N-2}\alpha\,\mathrm{d}\alpha\end{aligned} \tag{4.6}$$

where the lower limit the integral $u_{\min} = \sigma\beta/2R$ which is obtained from boundary condition $r^2 \le R^2$ of $(1+1)$-ES. Compared to the convergence velocity on sphere Model for standard $(1+1)$-ES,

$$\tilde{\varphi} = \frac{2R\sigma}{\sqrt{2\pi}}\exp\left(-\frac{\sigma^2 N^2}{8R^2}\right) - \sigma^2 n\left(1-\Phi\left(\frac{\sigma N}{2R}\right)\right), \tag{4.7}$$

There is no possible analytical work furthermore. Instead we compare Equation 4.6 and 4.7 numerically. The following figure shows the comparison when $\sigma = 1$. The difference between these two convergence velocities is roughly 0.73 and remain unchanged as the dimension increases, as shown in Figure 4.2.

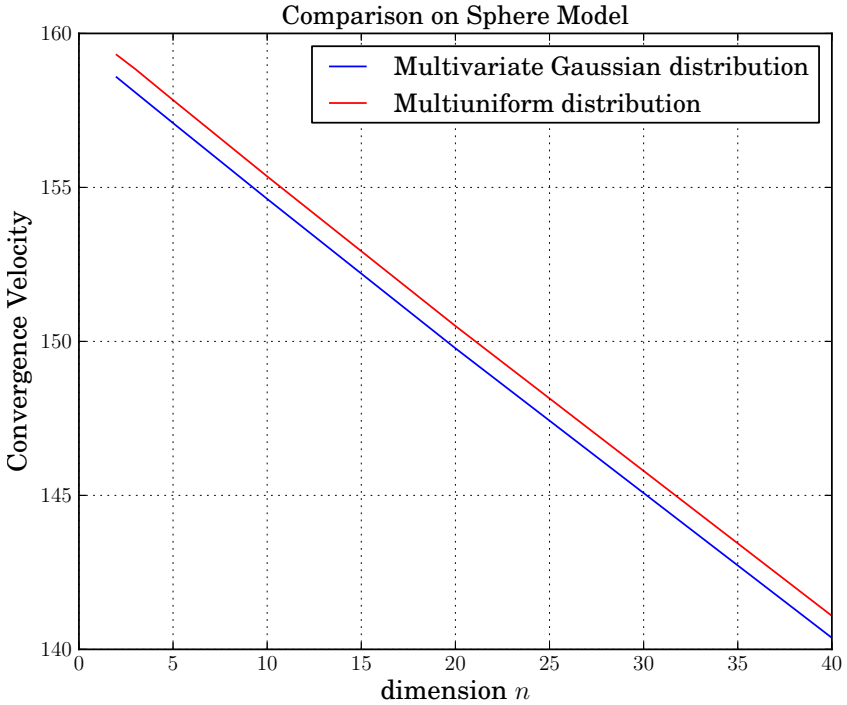

Figure 4.2: Convergence velocity comparison of $(1 + 1)$-ES on the sphere model.

Chapter 5

Empirical results

The goal of this chapter is to test the proposed ES variants systematically. In order to conduct a reliable and comparable test, we adopt the BBOB (Black-Box Optimization Benchmarking) softeware as our testbed and benchmark. The BBOB is a benchmark for systematic and sound comparisons of real parameter global optimizers. The benchmark currently contains 24 test functions which enable to change optimal point and optimal value under some constraints from trial to trial. It also provides pretty handy tools for post-processing and visualization of the experiment data.

This chapter is organized as follows. In Section 5.1, the reasonable quantitative measure of the ES performance, *expected running time* (ERT), is introduced. Section 5.2 explains some detailed aspects of BBOB benchmark and the overall setup of the our experiment. Section 5.3 gives some instructions on reading the BBOB figures and tables. Finally, Section 5.4 contains the specific results obtained by BBOB post-processing procedures, both for single ES variants and the comparisons among them.

5.1 Performance measure

The performance of any ES algorithm could be conceived by its speed of approaching the global optimum, which means using as few function evaluations as possible. A quantitative measure is needed to make how fast a ES variant is, which is always critical. It should reflect most of the aspects of the algorithm performance. In BBOB, *expected running time* (ERT) is used as the most prominent performance measure. It is introduced in [34] as *the average of the expected number of function evaluations per success and the expected number of function evaluations per unsuccessful trial*. Due to the numerical precision, the success of one trial is defined as reaching the optimal value f^* under a given precision Δf, namely the current fitness $f \leq f^* + \Delta f$. After multiple trials, the success rate p_s can be obtained. Let RT_{S} and RT_{US} denote the average number of function evaluations among the successful and unsuccessful trials, respectively. If $p_s \neq 0$, then ERT is defined as

$$\begin{aligned} \mathrm{ERT}(f_{\mathrm{target}}) &= RT_{\mathrm{S}} + \frac{1-p_s}{p_s} RT_{\mathrm{US}} \\ &= \frac{\langle FEs \rangle}{p_s} \end{aligned}$$

where $\langle FEs \rangle$ is the mean of all the function evaluations performed in the experiment. It is quite obvious that the ERT measure is positively correlated to the number of evaluations consumed while negatively correlated to the success rate. Given the same success rate, the more function evaluation consumed, the worse the algorithm performs. Given the same function evaluation, the more success trial algorithm makes, the better the algorithms. The practical reason for defining the ERT measure is that the global optimums of some test functions (highly multimodal) can only be reached by enormous number of function evaluations. However, the evaluation budget prescribed in the real experiment is relatively small in order to reduce the computational overheads. Therefore, many trials in an experiment are unsuccessful and the ERT measure is a reasonable quantity which is able to handle the unsuccessful trials.

5.2 Experiment details

5.2.1 BBOB features

The experiment is conducted on BBOB-2012[1] software, in which a collection of selected test functions and the interfaces to programming languages C/C++, Java, Python, Matlab/Octave are provided. In addition, real-time experiment data and configurations, like the number of function evaluations consumed or the current evaluation budget are governed and recorded by the software automatically. These disturbing details of the experiment are hidden from the user and the method to construct a experiment is quite standardized[2]. Using BBOB for a large benchmark work could save us a lot of efforts. In the test, the real-parameter search algorithm is run on a testbed of benchmark functions to be minimized. On each function and for each dimensionality Ntrial trials are carried out. In each trial, a function instance is created by performing some operations to the function base. The details about the operations involved in the test function generation can be found in [20].

5.2.2 Experiment objects

All the ES algorithm covered in this thesis and their first occurrence are listed in Table 5.1

ES name	tested?	First occurrence	Section of results
(μ, λ)-MSC-ES	No	Section 2.2.2	None
Simple-(μ, λ)-MSC-ES	Yes	Section 2.2.5	None
Cu-Simple-(μ, λ)-MSC-ES	Yes	Section 2.2.5	Section 5.4.1
$(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES	Yes	Section 3.1	None
Noisy$-(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES	Yes	Section 3.4	Section 5.4.2
Orthogonal1 $-$ $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES	Yes	Section 4.1.3	Section 5.4.6
Orthogonal2 $-$ $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES	Yes	Section 4.1.3	Section 5.4.7
(μ, λ)-CMSA-ES	Yes	Section 4.2.4	None
Derandomized-stepsize-(μ, λ)-CMSA-ES	Yes	Section 4.2.4	Section 5.4.3

Table 5.1: All the ES variants covered in this thesis.

[1] The exact version is v11.06.

[2] There is an example in the BBOB software showing how to construct you own experiment.

Note that we do not given the results of some tested ES algorithms. The reason is that these algorithms have already been tested extensively and their results can be found easily from web. In addition, the goal of this chapter is not to benchmark and rank all well-known ES variants. We focus on validating whether our new ES variants actually improve the performance by comparing them to the standard ES algorithms. Thus, some standard ES algorithms are still tested here and their results are not listed in this chapter. In this manner, three empirical comparisons, Cu-Simple-(μ, λ)-MSC-ES against Simple-(μ, λ)-MSC-ES, Noisy-$(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES against $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES, Orthogonal$1/2-(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES against $(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES and Derandomized-stepsize-CMSA-ES against (μ, λ)-CMSA-ES, are shown from Section 5.3.4 to 5.3.8.

5.2.3 Test functions

The following table provides a summary of all 24 test functions with their commonly used names and some of their features. The detailed description of the test functions can be found on BBOB web page[3].

Symbol	Name	Characteristic
f_1	sphere	unimodal, high symmetric
f_2	ellipsoid	unimodal, separable, condition $> 10^6$
f_3	Rastrigin	multimodal
f_4	Buche-Rastrigin	multimodal
f_5	linear slope	unimodal
f_6	attractive sector	unimodal, high asymmetric
f_7	step ellipsoid	unimodal with many plateaus
f_8	Rosenbrock	multimodal
f_9	rotated Rosenbrock	multimodal
f_{10}	ellipsoid	unimodal, non separable version of f_2
f_{11}	discus	unimodal, condition $> 10^6$
f_{12}	bent cigar	unimodal, condition $> 10^6$
f_{13}	sharp ridge	unimodal
f_{14}	different powers	unimodal
f_{15}	Rastrigin	unimodal, non separable version of f_3
f_{16}	Weierstrass	multimodal
f_{17}	Schaffers $F7$	highly multimodal
f_{18}	ill-conditioned Schaffers $F7$	highly multimodal
f_{19}	composite Griewank-Rosenbrock	highly multimodal
f_{20}	Schwefel function	multimodal
f_{21}	Gallagher's Gaussian 101-me peaks	multimodal with randomly distributed local optima
f_{22}	Gallagher's Gaussian 21-hi peaks	multimodal distributed local optima
f_{23}	Katsuura	highly multimodal
f_{24}	Lunacek bi-Rastrigin Function	highly multimodal

Table 5.2: 24 test functions in BBOB-2012

[3]http://coco.gforge.inria.fr/doku.php?id=downloads

All the test function bases are defined everywhere in $\mathbb{R}^N$ (N is the dimensionality) and have their global optimum in $[-5, 5]^N$. Most functions have their global optimum in $[-4, 4]^N$, which could be used as a reasonable setting for the initial parent. As introduced before, the optimization algorithm reaches the global optimum if and only if the difference between its current fitness f and the optimal fitness f^* is less than $\Delta f = 10^{-8}$, namely $f^* - f < \Delta f$. Such condition value Δf is called the target precision value.

Note that there are two versions of each function in the table above, the noiseless and noisy one are both implemented in the BBOB. In our experiment, only noiseless functions are exploited. As listed in the third column of the table above, these test functions have different properties, such as ill-conditioning, separability and symmetry. Each property stands for a certain type of difficulty that ES algorithm would encounter. We introduce those properties here briefly.

1. **Ill-Conditioning**. Ill-conditioning is a typical challenge in real-parameter optimization. If the function contour lines around the global (or local) optimum can be approximated by a convex quadratic function, $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$, then the conditioning of a function can be rigorously formalized as the condition number of the Hessian matrix $\mathbf{H}$, namely the ratio between the largest eigenvalue and the smallest eigenvalue. Because contour lines associated to a convex quadratic function are ellipsoids, the condition number corresponds to the square of the ratio between the length of the largest axis of the ellipsoid and the shortest axis. The BBOB testbed contains ill-conditioned functions with a typical conditioning of 10^6.

2. **Separability**. A high dimensional separable functions can be reformulated into a product or sum of univariate functions which takes one dimension component as its input. This implies that the separable functions pose an essentially different search problem to solve, because the search process can be reduced to N one-dimensional search procedures. Therefore, non-separable functions must be considered much more difficult and most benchmark functions are non-separable.

3. **Symmetry**. Most of the ES algorithms are built on the Gaussian mutation operator, which is symmetric in high dimensional space (or rotation-invariant as discussed before). Consequently, the symmetric benchmark functions could be in favor of these operators. In order to test the real performance of a ES, some asymmetric functions are needed to show how the ES performs in its less favorable case. Thus, a good benchmark should include both symmetric functions and asymmetric functions.

5.2.4 Parameter Setting and Implementation Issue

The parameter setting of the experiment is the same for all the tested ES variants. The initial global step-size σ is set to 1. The maximal function evaluations is set to $1e6 \times N$ (N is the dimensionality). The initial solution vector (initial parent) is a uniformly distributed random vector restricted in hyper box $[-4, 4]^N$. The dimensionality used in the experiment are $2, 3, 5, 10, 20, 40$. The maximal function evaluations is somehow different. For the most of the algorithms tested, the evaluation budget is set to $10^5 \times N$ (N is the dimensionality) except for Cu-Simple-(μ, λ)-MSC-ES and Simple-(μ, λ)-MSC-ES in which maximal function evaluations is set to $5 \times 10^4 \times N$.

All the tested ES algorithms are coded in Python and thus the Python version of BBOB-2012 is used in the benchmarking. In the algorithm code, the *Numpy* [30], the numeric Python package is extensively used to improve the efficiency. The time-consuming sub-procedure, NOISE-GENERATION is implemented in C and then imported and invoked by the other algorithms. The example testing code in BBOB-2012 is just a sequential execution from the first test function to the last one, which would take too long if we need a very big test (large function evaluation budget). Thus, the author changes it into the parallel execution mode in which 24 processes are created for 24 test functions. In such way, all the test function can perform parallel on multi-core computers, clusters or grid. Our experiment is basically conducted on the DAS-4[4] cluster located in LIACS[5] (Leiden Institute of Advanced Computer Science). The experiment usually occupies two DAS-4 nodes, each of which contains an Intel® Xeon® 2.67GHz hexadeca-core CPU X5650. For the detailed code for the algorithm, the author is likely to provide you. Please contact: wangronin@gmail.com if you need.

5.3 Explanation on the BBOB figures

The experiment results presented in the following are generated by the post-processing procedure in BBOB software. They are quite scientific and need some explanations. Each type of the figures in the results is explained in this section to make themselves readable.

1. **Running time figure.** The running time figures resemble the example below. The

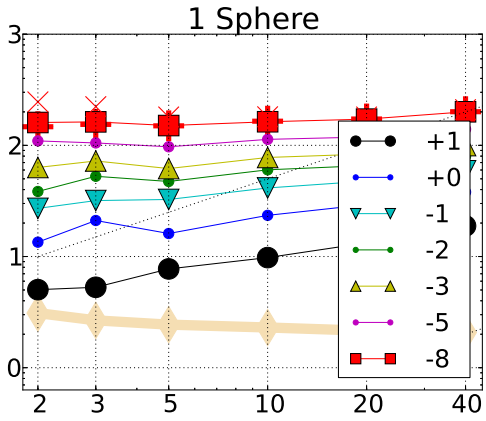

Figure 5.1: An Example of running time figures.

Expected number of f-evaluations (ERT measure) to reach $f_{\text{opt}}+\Delta f$, is plotted against the dimensionality in such type. Different numerical precisions ($\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$) are also shown. The title of the figure gives the function on which the raw are obtained.

[4] http://www.cs.vu.nl/das4/clusters.shtml
[5] http://www.liacs.nl/home-en/

2. **Empirical cumulative distribution function.** This type of the figure does not directly show the ERT data of the experiment. Instead, the probability that a ES algorithm make a successful trial on a certain test function is investigated. Considering the number of the function evaluations on one test function as a random variable, then its cumulative probability distribution could be very usefully since it characterizes the probability that certain amount of function evaluations are enough to reach the global optimum. The function evaluation consumed in one trial running can be considered as one sample of such distribution. Thus the approximate distribution function can be calculated though the multiple trials on one function, and is called the empirical cumulative distribution function (ECDF). The more trials conducted, the more accurate the ECDFs would be. However, due to the computational complexity, 15 trials are conducted for each test function and are sufficient to make relevant performance differences statistically significant. A typical ECDF figure is shown below.

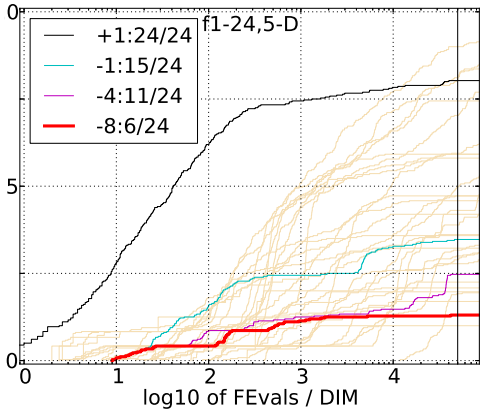

Figure 5.2: An Example of Empirical cumulative distribution function figures.

The real random variable in practice is the logarithm of the number of the function evaluations normalized by the dimensionality as can be seen in Figure 5.2. Four empirical cumulative distribution functions for four different Δf are shown in each figure. In addition, such statistical analysis is performed for dimension 5 and 20, and for certain group of the test functions (e.g. unimodal functions, separable functions).
The implication of ECDFs on ES performance is quite straightforward. The less function evaluations a ES algorithm likely to consume when reaching the optimum, the better this ES performs. Consequently, if an ECDF curves increasing faster than others, it is considered better. Or, in another viewpoint, for a given number of function evaluations, the bigger probability an ECDF gives, the better its corresponding ES performs. ECDFs play a vital role in the ES comparison scenario.

3. **ERT ratio figure.** The comparison of two ES algorithms is mainly done in the ERT ratio figure type. A typical one looks like,

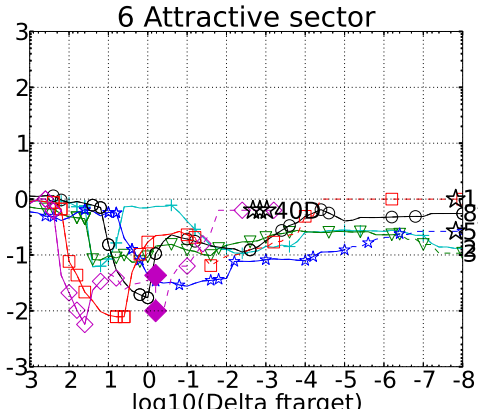

Figure 5.3: An Example of ERT ratio figures.

Given the ERT measures for two ES .algorithm, ERT1 and ERT2 for each dimension and each Δf. The $\log \frac{\text{ERT1}}{\text{ERT2}}$ are plotted against $\log \Delta f$ in Figure 5.3. Multiple curves shows the comparisons for different dimensions. From this type of figure, it is obvious to verify whether an ES algorithm is better than another on one test function. In Figure 5.3, all the curves are below 0 for every Δf. This means $\log \frac{\text{ERT1}}{\text{ERT2}}$ is always negative and therefore ERT1 is always smaller. The maximum or minimum of one curve illustrates the condition under which one ES beats its adversary the most and the amount of its advantages in performance.

5.4 Results on BBOB-2012 benchmark

5.4.1 Cu-Simple-(μ, λ)-MSC-ES

The characteristic of Cu-Simple-(μ, λ)-MSC-ES is shown in Figure 5.4, 5.5 and in Table 5.3, 5.4. The expected number of function evaluations (ERT) for all 24 test functions are shown in Figure 5.4. For one test function, the ERT numbers are illustrated for each dimension and for each target precision. 7 target precisions $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$are provided to characterize the ability of the optimizer. The specific ERT value for precision 10^{-8} is shown in Table 5.4.

The empirical cumulative distribution functions of function evaluations divided by search space dimension are shown in the left part of Figure 5.5 while the empirical distributions of target precisions are shown in the right. The ERT loss ratio is compared to that of BBOB-2009 best in Table 5.3.

Table 5.3: ERT loss ratio compared to the respective best result from BBOB-2009 for budgets given in the first column. The last row $\mathrm{RL_{US}}$/D gives the number of function evaluations in unsuccessful runs divided by dimension. Shown are the smallest, 10%-ile, 25%-ile, 50%-ile, 75%-ile and 90%-ile value (smaller values are better). The ERT Loss ratio equals to one for the respective best algorithm from BBOB-2009. Typical median values are between ten and hundred.

	f_1–f_{24} in 5-D, maxFE/D=49999					
#FEs/D	best	10%	25%	**med**	75%	90%
2	2.1	3.0	3.8	6.5	10	10
10	1.6	1.8	2.7	3.3	5.1	18
100	1.2	2.5	5.6	9.7	20	34
1e3	6.7	8.1	28	62	98	3.0e2
1e4	8.2	57	1.4e2	3.9e2	8.1e2	1.3e3
1e5	8.2	57	3.3e2	1.2e3	2.5e3	6.1e3
$\mathrm{RL_{US}}$/D	5e4	5e4	5e4	5e4	5e4	5e4
	f_1–f_{24} in 20-D, maxFE/D=49999					
#FEs/D	best	10%	25%	**med**	75%	90%
2	1.0	3.9	8.9	31	40	40
10	0.94	1.7	2.9	4.6	6.6	28
100	1.7	1.9	3.7	6.3	24	74
1e3	6.9	11	22	39	88	4.7e2
1e4	14	54	83	1.9e2	6.3e2	2.7e3
1e5	14	1.1e2	3.1e2	7.9e2	4.8e3	1.4e4
1e6	14	2.1e2	8.6e2	4.4e3	1.1e4	5.7e4
$\mathrm{RL_{US}}$/D	5e4	5e4	5e4	5e4	5e4	5e4

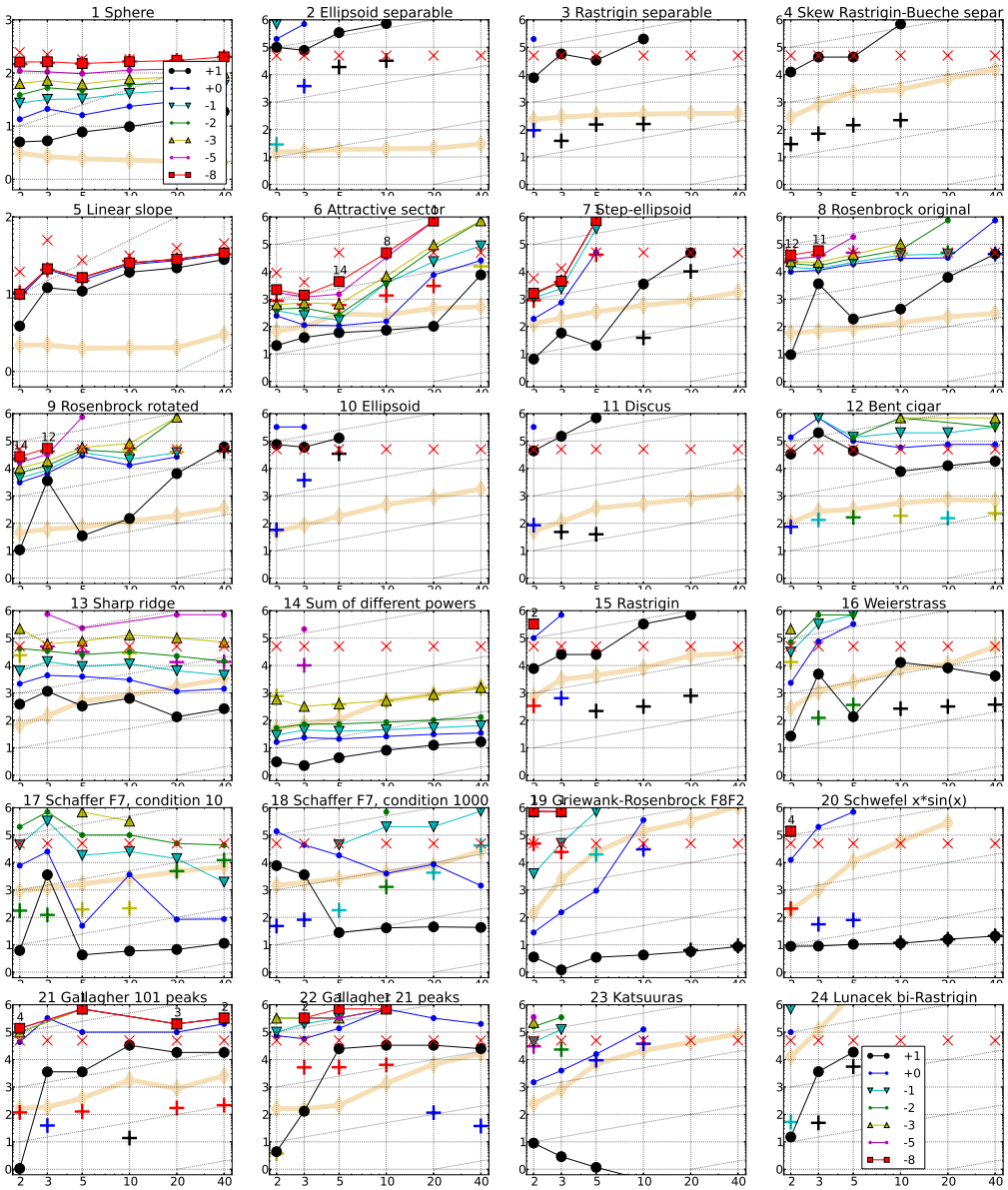

Figure 5.4: Expected number of f-evaluations (ERT, with lines, see legend) to reach $f_{\mathrm{opt}} + \Delta f$, median number of f-evaluations to reach the most difficult target that was reached at least once ($+$) and maximum number of f-evaluations in any trial ($\times$), all divided by dimension and plotted as $\log_{10}$ values versus dimension. Shown are $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$. Numbers above ERT-symbols indicate the number of successful trials. The light thick line with diamonds indicates the respective best result from BBOB-2009 for $\Delta f = 10^{-8}$. Horizontal lines mean linear scaling, slanted grid lines depict quadratic scaling.

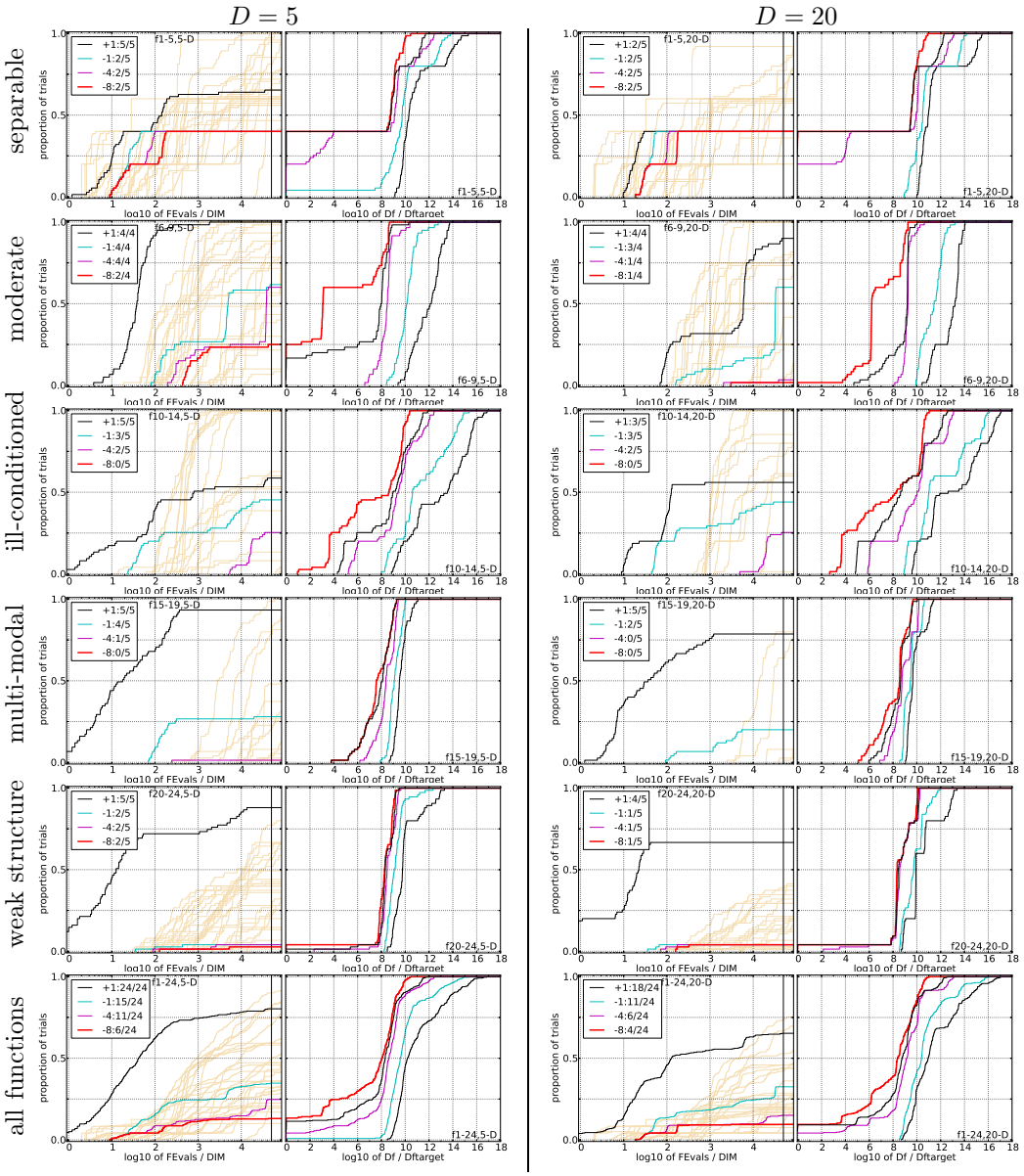

Figure 5.5: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials with an outcome not larger than the respective value on the x-axis. Left subplots: ECDF of number of function evaluations (FEvals) divided by search space dimension D, to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D, \ldots$ function evaluations (from right to left cycling black-cyan-magenta). The thick red line represents the most difficult target value $f_{\text{opt}} + 10^{-8}$. Legends indicate the number of functions that were solved in at least one trial. Light brown lines in the background show ECDFs for $\Delta f = 10^{-8}$ of all algorithms benchmarked during BBOB-2009.

5-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_1}$	11	12	12	12	12	12	15/15
	3.5(3)	6.6(5)	13(5)	25(6)	40(9)	55(10)	15/15
$\mathbf{f_2}$	83	87	88	90	92	94	15/15
	20624(22001)	∞	∞	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_3}$	716	1622	1637	1646	1650	1654	15/15
	234(349)	∞	∞	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_4}$	809	1633	1688	1817	1886	1903	15/15
	271(463)	∞	∞	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_5}$	10	10	10	10	10	10	15/15
	5.5(3)	7.6(3)	8.1(4)	8.2(4)	8.2(4)	8.2(4)	15/15
$\mathbf{f_6}$	114	214	281	580	1038	1332	15/15
	2.6(1)	2.5(1)	3.0(2)	5.7(3)	7.3(3)	16(3)	14/15
$\mathbf{f_7}$	24	324	1171	1572	1572	1597	15/15
	4.4(4)	770(973)	1568(1814)	2360(2465)	2360(2624)	2323(2505)	1/15
$\mathbf{f_8}$	73	273	336	391	410	422	15/15
	13(8)	348(468)	334(380)	540(335)	2247(2438)	∞ *2.5e5*	0/15
$\mathbf{f_9}$	35	127	214	300	335	369	15/15
	5.0(2)	1158(1968)	889(1175)	970(838)	11171(11926)	∞ *2.5e5*	0/15
$\mathbf{f_{10}}$	349	500	574	626	829	880	15/15
	1848(1800)	∞	∞	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_{11}}$	143	202	763	1177	1467	1673	15/15
	24545(29803)	∞	∞	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_{12}}$	108	268	371	461	1303	1494	15/15
	2026(2311)	1869(2334)	1857(2361)	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_{13}}$	132	195	250	1310	1752	2255	15/15
	13(14)	102(90)	184(160)	293(326)	658(714)	1600(1830)	0/15
$\mathbf{f_{14}}$	10	41	58	139	251	476	15/15
	2.2(3)	2.6(1)	3.4(1)	14(9)	∞	∞ *2.5e5*	0/15
$\mathbf{f_{15}}$	511	9310	19369	20073	20769	21359	14/15
	247(490)	∞	∞	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_{16}}$	120	612	2662	10449	11644	12095	15/15
	5.6(5)	2659(3063)	1315(1549)	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_{17}}$	5.2	215	899	3669	6351	7934	15/15
	4.1(5)	1.2(0.9)	102(139)	954(1090)	∞	∞ *2.5e5*	0/15
$\mathbf{f_{18}}$	103	378	3968	9280	10905	12469	15/15
	1.4(1)	242(331)	55(79)	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_{19}}$	1	1	242	1.2e5	1.2e5	1.2e5	15/15
	18(20)	4694(5056)	14858(16515)	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_{20}}$	16	851	38111	54470	54861	55313	14/15
	3.3(3)	4115(4702)	∞	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_{21}}$	41	1157	1674	1705	1729	1757	14/15
	437(3)	432(540)	2090(2314)	2053(2493)	2025(2386)	1992(2277)	1/15
$\mathbf{f_{22}}$	71	386	938	1008	1040	1068	14/15
	1762(3521)	1780(2264)	1734(2133)	1619(1921)	1577(1802)	1545(1755)	1/15
$\mathbf{f_{23}}$	3.0	518	14249	31654	33030	34256	15/15
	2.0(3)	153(199)	∞	∞	∞	∞ *2.5e5*	0/15
$\mathbf{f_{24}}$	1622	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
	58(87)	∞	∞	∞	∞	∞ *2.5e5*	0/15

20-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_1}$	43	43	43	43	43	43	15/15
	6.2(2)	14(2)	22(2)	39(2)	55(3)	72(3)	15/15
$\mathbf{f_2}$	385	386	387	390	391	393	15/15
	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_3}$	5066	7626	7635	7643	7646	7651	15/15
	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_4}$	4722	7628	7666	7700	7758	1.4e5	9/15
	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_5}$	41	41	41	41	41	41	15/15
	11(3)	13(3)	14(3)	14(3)	14(3)	14(3)	15/15
$\mathbf{f_6}$	1296	2343	3413	5220	6728	8409	15/15
	1.6(0.5)	65(106)	136(164)	367(454)	2088(2527)	1672(1843)	1/15
$\mathbf{f_7}$	1351	4274	9503	16524	16524	16969	15/15
	725(859)	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_8}$	2039	3871	4040	4219	4371	4484	15/15
	61(17)	167(134)	220(127)	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_9}$	1716	3102	3277	3455	3594	3727	15/15
	75(32)	171(165)	230(155)	4285(4342)	∞	∞ *1.0e6*	0/15
$\mathbf{f_{10}}$	7413	8661	10735	14920	17073	17476	15/15
	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{11}}$	1002	2228	6278	9762	12285	14831	15/15
	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{12}}$	1042	1938	2740	4140	12407	13827	15/15
	242(480)	776(1032)	1461(1825)	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{13}}$	652	2021	2751	18749	24455	30201	15/15
	4.1(0.7)	11(14)	47(54)	109(116)	583(654)	∞ *1.0e6*	0/15
$\mathbf{f_{14}}$	75	239	304	932	1648	15661	15/15
	3.3(0.8)	2.6(0.3)	3.5(0.5)	19(6)	∞	∞ *1.0e6*	0/15
$\mathbf{f_{15}}$	30378	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
	461(576)	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{16}}$	1384	27265	77015	1.9e5	2.0e5	2.2e5	15/15
	118(362)	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{17}}$	63	1030	4005	30677	56288	80472	15/15
	2.1(1)	1.6(0.8)	69(128)	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{18}}$	621	3972	19561	67569	1.3e5	1.5e5	15/15
	1.5(1)	44(126)	209(231)	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{19}}$	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
	114(48)	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{20}}$	82	46150	3.1e6	5.5e6	5.6e6	5.6e6	14/15
	3.9(1)	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{21}}$	561	6541	14103	14643	15567	17589	15/15
	648(891)	306(382)	284(355)	273(341)	257(321)	228(284)	3/15
$\mathbf{f_{22}}$	467	5580	23491	24948	26847	1.3e5	12/15
	1429(2142)	1165(1389)	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{23}}$	3.2	1614	67457	4.9e5	8.1e5	8.4e5	15/15
	2.5(3)	∞	∞	∞	∞	∞ *1.0e6*	0/15
$\mathbf{f_{24}}$	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
	∞	∞	∞	∞	∞	∞ *1.0e6*	0/15

Table 5.4: Expected running time (ERT in number of function evaluations) divided by the best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values for functions f_1–f_{24}. The median number of conducted function evaluations is additionally given in *italics*, if $\mathrm{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\mathrm{opt}} + 10^{-8}$.

5.4.2 Noisy$-(\mu/\mu_{\mathbf{w}}, \lambda_{\mathbf{m}})$-CMA-ES

The characteristic of Noisy$-(\mu/\mu_{\mathrm{w}}, \lambda_{\mathrm{m}})$-CMA-ES is shown in Figure 5.6, 5.7 and in Table 5.5, 5.6. The expected number of function evaluations (ERT) for all 24 test functions are shown in Figure 5.6. For one test function, the ERT numbers are illustrated for each dimension and for each target precision. 7 target precisions $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$are provided to characterize the ability of the optimizer. The specific ERT value for precision 10^{-8} is shown in Table 5.6.

The empirical cumulative distribution functions of function evaluations divided by search space dimension are shown in the left part of Figure 5.7 while the empirical distributions of target precisions are shown in the right. The ERT loss ratio is compared to that of BBOB-2009 best in Table 5.5.

Table 5.5: ERT loss ratio compared to the respective best result from BBOB-2009 for budgets given in the first column. The last row $\mathrm{RL_{US}}$/D gives the number of function evaluations in unsuccessful runs divided by dimension. Shown are the smallest, 10%-ile, 25%-ile, 50%-ile, 75%-ile and 90%-ile value (smaller values are better). The ERT Loss ratio equals to one for the respective best algorithm from BBOB-2009. Typical median values are between ten and hundred.

	f_1–f_{24} in 5-D, maxFE/D=99998					
#FEs/D	best	10%	25%	**med**	75%	90%
2	1.1	2.2	3.7	6.1	8.3	10
10	1.9	2.2	2.8	3.5	5.3	37
100	0.68	1.7	3.8	7.9	23	98
1e3	1.1	2.9	6.6	29	75	1.1e2
1e4	1.1	2.9	7.1	37	82	4.5e2
1e5	1.1	2.9	7.1	37	1.4e2	4.6e2
$\mathrm{RL_{US}}$/D	1e5	1e5	1e5	1e5	1e5	1e5
	f_1–f_{24} in 20-D, maxFE/D=99999					
#FEs/D	best	10%	25%	**med**	75%	90%
2	1.0	1.6	7.8	31	40	40
10	0.79	1.8	2.6	3.7	6.0	2.0e2
100	0.13	0.52	2.4	3.0	11	2.5e2
1e3	0.21	1.5	4.4	15	51	2.3e2
1e4	1.00	2.0	6.1	54	2.1e2	6.0e2
1e5	1.00	2.0	6.1	72	4.5e2	5.9e3
1e6	1.00	2.0	6.1	96	9.8e2	7.9e3
$\mathrm{RL_{US}}$/D	1e5	1e5	1e5	1e5	1e5	1e5

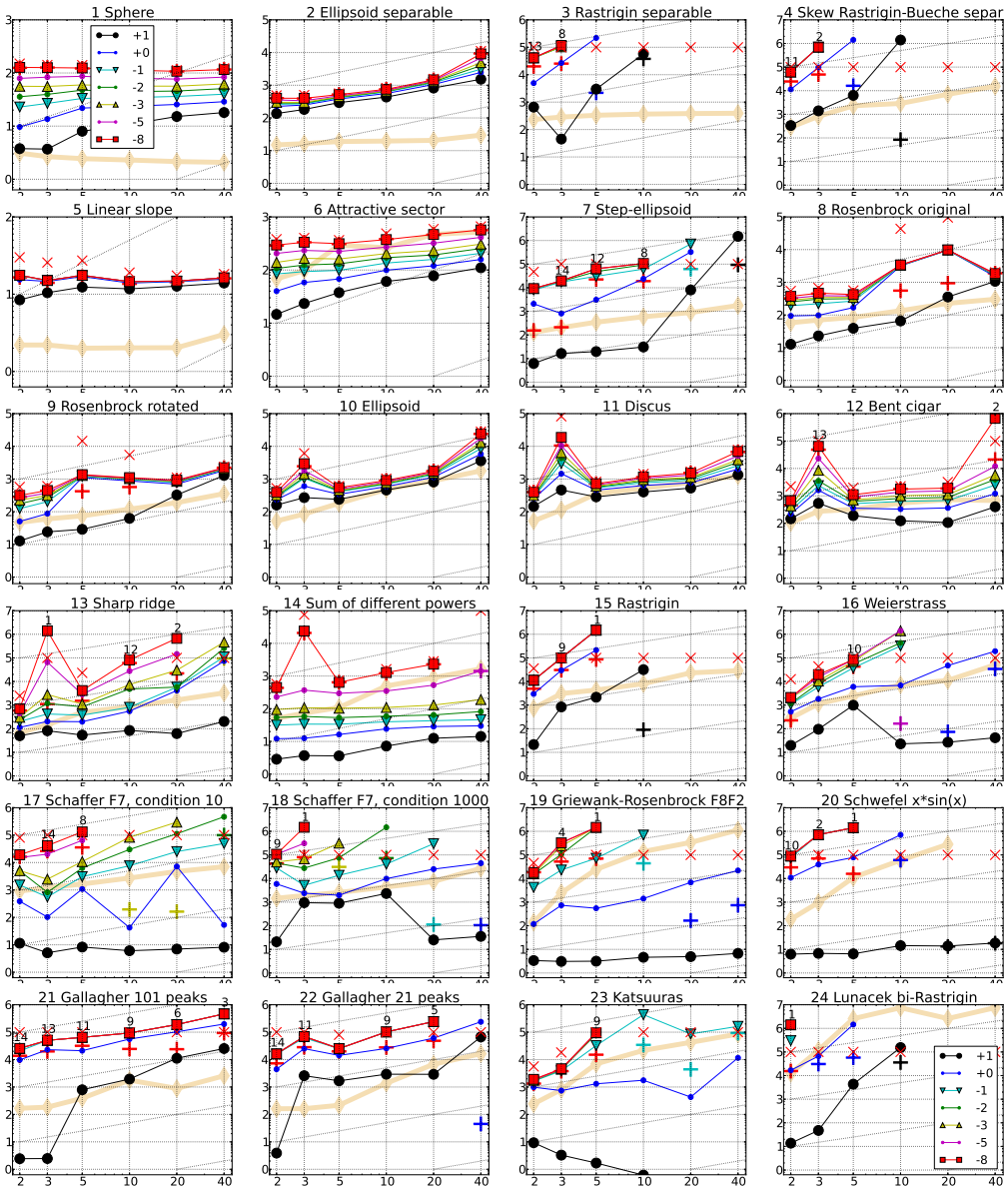

Figure 5.6: Expected number of f-evaluations (ERT, with lines, see legend) to reach $f_{\text{opt}} + \Delta f$, median number of f-evaluations to reach the most difficult target that was reached at least once ($+$) and maximum number of f-evaluations in any trial ($\times$), all divided by dimension and plotted as $\log_{10}$ values versus dimension. Shown are $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$. Numbers above ERT-symbols indicate the number of successful trials. The light thick line with diamonds indicates the respective best result from BBOB-2009 for $\Delta f = 10^{-8}$. Horizontal lines mean linear scaling, slanted grid lines depict quadratic scaling.

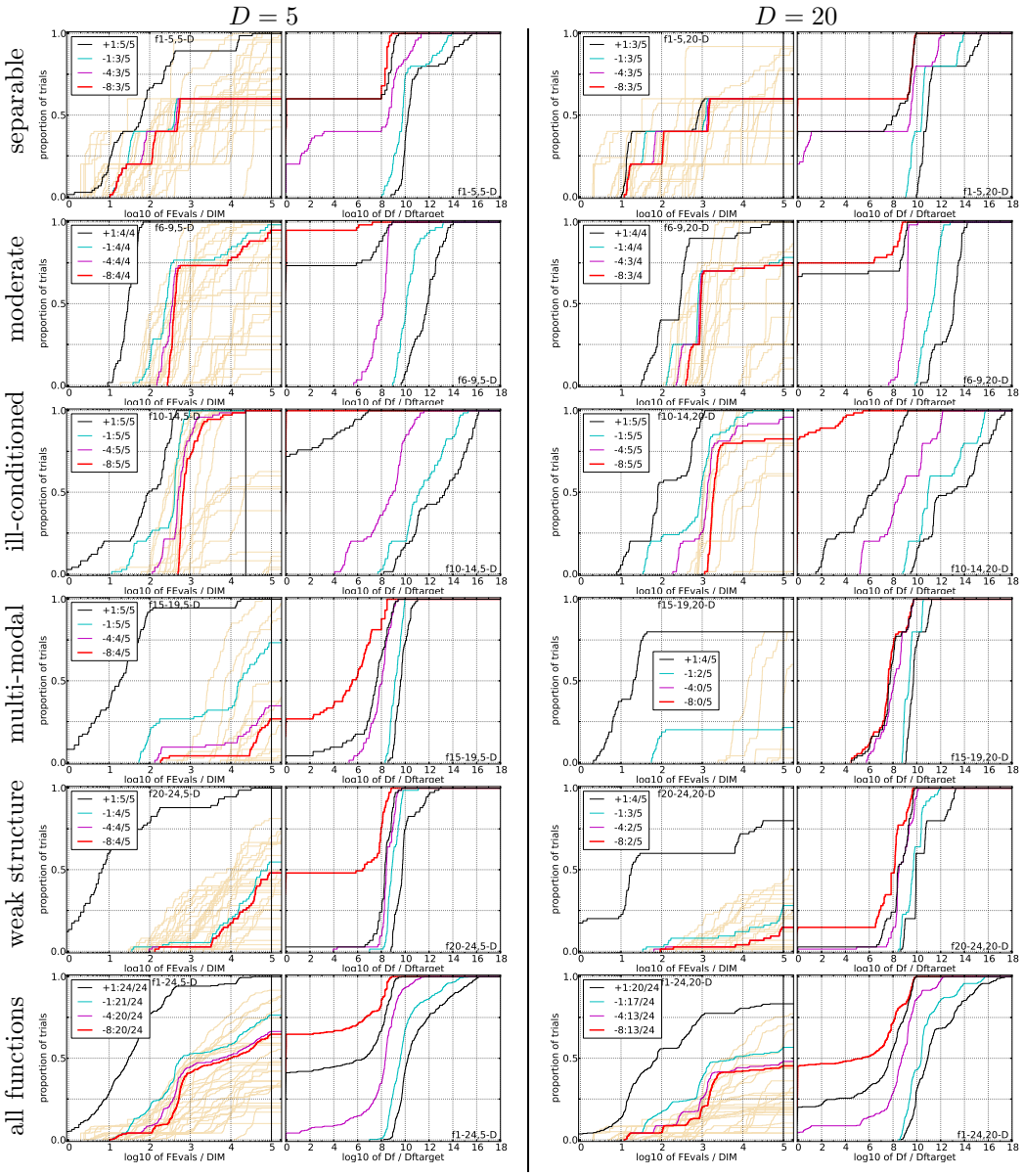

Figure 5.7: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials with an outcome not larger than the respective value on the x-axis. Left subplots: ECDF of number of function evaluations (FEvals) divided by search space dimension D, to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D, \ldots$ function evaluations (from right to left cycling black-cyan-magenta). The thick red line represents the most difficult target value $f_{\text{opt}} + 10^{-8}$. Legends indicate the number of functions that were solved in at least one trial. Light brown lines in the background show ECDFs for $\Delta f = 10^{-8}$ of all algorithms benchmarked during BBOB-2009.

	5-D								20-D							
Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ	Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_1}$	11	12	12	12	12	12	15/15	$\mathbf{f_1}$	43	43	43	43	43	43	15/15
	3.6(3)	9.0(2)	14(2)	24(4)	35(5)	46(6)	15/15		7.0(1)	12(1)	17(1)	26(2)	36(2)	45(2)	15/15
$\mathbf{f_2}$	83	87	88	90	92	94	15/15	$\mathbf{f_2}$	385	386	387	390	391	393	15/15
	18(6)	22(3)	23(3)	25(2)	26(2)	27(2)	15/15		43(8)	55(9)	62(7)	68(4)	71(4)	73(4)	15/15
$\mathbf{f_3}$	716	1622	1637	1646	1650	1654	15/15	$\mathbf{f_3}$	5066	7626	7635	7643	7646	7651	15/15
	21(50)	674(774)	∞	∞	∞	∞ *5.0e5*	0/15		∞	∞	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_4}$	809	1633	1688	1817	1886	1903	15/15	$\mathbf{f_4}$	4722	7628	7666	7700	7758	1.4e5	9/15
	39(50)	4335(5204)	∞	∞	∞	∞ *5.0e5*	0/15		∞	∞	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_5}$	10	10	10	10	10	10	15/15	$\mathbf{f_5}$	41	41	41	41	41	41	15/15
	6.2(3)	8.5(4)	8.7(4)	8.7(4)	8.7(4)	8.7(4)	15/15		6.2(0.9)	7.1(0.9)	7.2(0.7)	7.3(0.7)	7.3(0.7)	7.3(0.7)	15/15
$\mathbf{f_6}$	114	214	281	580	1038	1332	15/15	$\mathbf{f_6}$	1296	2343	3413	5220	6728	8409	15/15
	1.7(0.7)	1.6(0.4)	1.8(0.4)	1.4(0.4)	1.1(0.2)	1.1(0.1)	15/15		1.2(0.2)	1.0(0.2)	0.93(0.1)	0.89(0.1)	0.95(0.1)	0.99(0.1)	15/15
$\mathbf{f_7}$	24	324	1171	1572	1572	1597	15/15	$\mathbf{f_7}$	1351	4274	9503	16524	16524	16969	15/15
	4.2(2)	48(80)	130(172)	202(258)	202(186)	199(185)	12/15		119(214)	1528(1482)	1498(1544)	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_8}$	73	273	336	391	410	422	15/15	$\mathbf{f_8}$	2039	3871	4040	4219	4371	4484	15/15
	2.7(1)	3.1(1)	4.0(0.9)	4.4(0.8)	4.6(0.7)	4.9(0.7)	15/15		3.5(0.9)	49(85)	48(82)	46(78)	45(76)	44(74)	15/15
$\mathbf{f_9}$	35	127	214	300	335	369	15/15	$\mathbf{f_9}$	1716	3102	3277	3455	3594	3727	15/15
	4.2(1)	44(2)	28(1)	21(1)	20(1)	18(1)	15/15		3.8(0.7)	4.5(0.5)	4.8(0.4)	4.9(0.4)	4.9(0.4)	4.9(0.3)	15/15
$\mathbf{f_{10}}$	349	500	574	626	829	880	15/15	$\mathbf{f_{10}}$	7413	8661	10735	14920	17073	17476	15/15
	3.4(1)	3.4(1.0)	3.6(0.3)	3.6(0.3)	2.9(0.2)	3.0(0.2)	15/15		2.2(0.8)	2.6(0.8)	2.5(0.4)	2.1(0.2)	1.9(0.2)	1.9(0.2)	15/15
$\mathbf{f_{11}}$	143	202	763	1177	1467	1673	15/15	$\mathbf{f_{11}}$	1002	2228	6278	9762	12285	14831	15/15
	10(4)	11(3)	3.5(0.7)	2.6(0.4)	2.2(0.4)	2.1(0.3)	15/15		11(2)	6.9(1)	2.9(0.4)	2.3(0.2)	2.1(0.2)	2.0(0.2)	15/15
$\mathbf{f_{12}}$	108	268	371	461	1303	1494	15/15	$\mathbf{f_{12}}$	1042	1938	2740	4140	12407	13827	15/15
	8.8(7)	6.6(5)	7.0(6)	7.7(4)	3.5(2)	3.5(2)	15/15		2.0(2)	3.8(4)	5.0(4)	5.4(2)	2.5(0.8)	2.6(0.8)	15/15
$\mathbf{f_{13}}$	132	195	250	1310	1752	2255	15/15	$\mathbf{f_{13}}$	652	2021	2751	18749	24455	30201	15/15
	2.0(0.6)	5.1(5)	7.5(6)	4.2(6)	8.1(7)	8.3(11)	15/15		1.9(0.4)	41(60)	42(43)	32(50)	119(153)	141(175)	2/15
$\mathbf{f_{14}}$	10	41	58	139	251	476	15/15	$\mathbf{f_{14}}$	75	239	304	932	1648	15661	15/15
	1.8(2)	2.0(1)	2.9(0.9)	3.7(1)	5.9(0.9)	5.5(0.7)	15/15		3.3(1.0)	2.4(0.4)	2.8(0.5)	2.8(0.3)	6.4(1)	2.1(0.3)	15/15
$\mathbf{f_{15}}$	511	9310	19369	20073	20769	21359	14/15	$\mathbf{f_{15}}$	30378	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
	21(68)	115(107)	384(426)	370(386)	358(397)	348(375)	1/15		∞	∞	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{16}}$	120	612	2662	10449	11644	12095	15/15	$\mathbf{f_{16}}$	1384	27265	77015	1.9e5	2.0e5	2.2e5	15/15
	41(2)	49(75)	70(81)	34(36)	36(43)	35(32)	10/15		0.39(0.1)	35(37)	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{17}}$	5.2	215	899	3669	6351	7934	15/15	$\mathbf{f_{17}}$	63	1030	4005	30677	56288	80472	15/15
	8.0(8)	25(26)	17(40)	15(22)	51(57)	83(82)	8/15		2.2(1)	139(0.2)	125(250)	194(228)	∞	∞ *2.0e6*	0/15
$\mathbf{f_{18}}$	103	378	3968	9280	10905	12469	15/15	$\mathbf{f_{18}}$	621	3972	19561	67569	1.3e5	1.5e5	15/15
	44(0.9)	26(87)	17(27)	170(189)	∞	∞ *5.0e5*	0/15		0.81(0.2)	126(252)	306(358)	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{19}}$	1	1	242	1.2e5	1.2e5	1.2e5	15/15	$\mathbf{f_{19}}$	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
	16(14)	2758(562)	1486(2064)	61(62)	61(68)	60(68)	1/15		99(48)	1.4e5(4e5)	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{20}}$	16	851	38111	54470	54861	55313	14/15	$\mathbf{f_{20}}$	82	46150	3.1e6	5.5e6	5.6e6	5.6e6	14/15
	2.0(1)	462(449)	186(203)	130(147)	129(137)	128(156)	1/15		3.4(0.9)	∞	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{21}}$	41	1157	1674	1705	1729	1757	14/15	$\mathbf{f_{21}}$	561	6541	14103	14643	15567	17589	15/15
	97(302)	91(128)	193(222)	189(202)	187(215)	184(196)	11/15		398(544)	315(333)	269(334)	259(273)	244(302)	216(238)	6/15
$\mathbf{f_{22}}$	71	386	938	1008	1040	1068	14/15	$\mathbf{f_{22}}$	467	5580	23491	24948	26847	1.3e5	12/15
	118(176)	184(279)	134(146)	124(136)	121(132)	118(128)	15/15		125(167)	217(321)	207(245)	195(230)	181(213)	36(43)	5/15
$\mathbf{f_{23}}$	3.0	518	14249	31654	33030	34256	15/15	$\mathbf{f_{23}}$	3.2	1614	67457	4.9e5	8.1e5	8.4e5	15/15
	2.8(3)	13(19)	11(14)	15(17)	14(17)	14(15)	9/15		2.7(3)	5.3(0.4)	26(30)	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{24}}$	1622	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15	$\mathbf{f_{24}}$	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
	13(23)	34(39)	∞	∞	∞	∞ *5.0e5*	0/15		∞	∞	∞	∞	∞	∞ *2.0e6*	0/15

Table 5.6: Expected running time (ERT in number of function evaluations) divided by the best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values for functions f_1–f_{24}. The median number of conducted function evaluations is additionally given in *italics*, if $\mathrm{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\mathrm{opt}} + 10^{-8}$.

5.4.3 Derandomized-stepsize-(μ, λ)-CMSA-ES

The characteristic of Derandomized-stepsize-(μ, λ)-CMSA-ES is shown in Figure 5.8, 5.9 and in Table 5.7, 5.8. The expected number of function evaluations (ERT) for all 24 test functions are shown in Figure 5.8. For one test function, the ERT numbers are illustrated for each dimension and for each target precision. 7 target precisions $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$are provided to characterize the ability of the optimizer. The specific ERT value for precision 10^{-8} is shown in Table 5.8.

The empirical cumulative distribution functions of function evaluations divided by search space dimension are shown in the left part of Figure 5.9 while the empirical distributions of target precisions are shown in the right. The ERT loss ratio is compared to that of BBOB-2009 best in Table 5.7.

Table 5.7: ERT loss ratio compared to the respective best result from BBOB-2009 for budgets given in the first column. The last row $\mathrm{RL_{US}}$/D gives the number of function evaluations in unsuccessful runs divided by dimension. Shown are the smallest, 10%-ile, 25%-ile, 50%-ile, 75%-ile and 90%-ile value (smaller values are better). The ERT Loss ratio equals to one for the respective best algorithm from BBOB-2009. Typical median values are between ten and hundred.

	f_1–f_{24} in 5-D, maxFE/D=99998					
#FEs/D	best	10%	25%	**med**	75%	90%
2	1.1	3.3	5.3	8.3	10	10
10	1.6	3.1	3.6	5.4	8.1	18
100	1.2	2.5	4.3	6.8	13	50
1e3	0.84	1.2	3.4	9.8	26	1.0e2
1e4	1.2	1.8	3.6	11	41	7.2e2
1e5	1.2	1.8	3.6	10	3.1e2	1.6e3
$\mathrm{RL_{US}}$/D	1e5	1e5	1e5	1e5	1e5	1e5
	f_1–f_{24} in 20-D, maxFE/D=99999					
#FEs/D	best	10%	25%	**med**	75%	90%
2	1.0	3.9	11	31	40	40
10	1.5	2.7	4.3	6.0	8.4	2.0e2
100	1.3	2.3	3.9	5.9	24	71
1e3	2.2	4.7	8.9	19	59	2.8e2
1e4	2.0	5.7	13	44	4.0e2	2.4e3
1e5	2.0	5.7	26	1.1e2	6.5e2	6.7e3
1e6	2.0	5.7	26	6.0e2	4.3e3	1.1e4
$\mathrm{RL_{US}}$/D	1e5	1e5	1e5	1e5	1e5	1e5

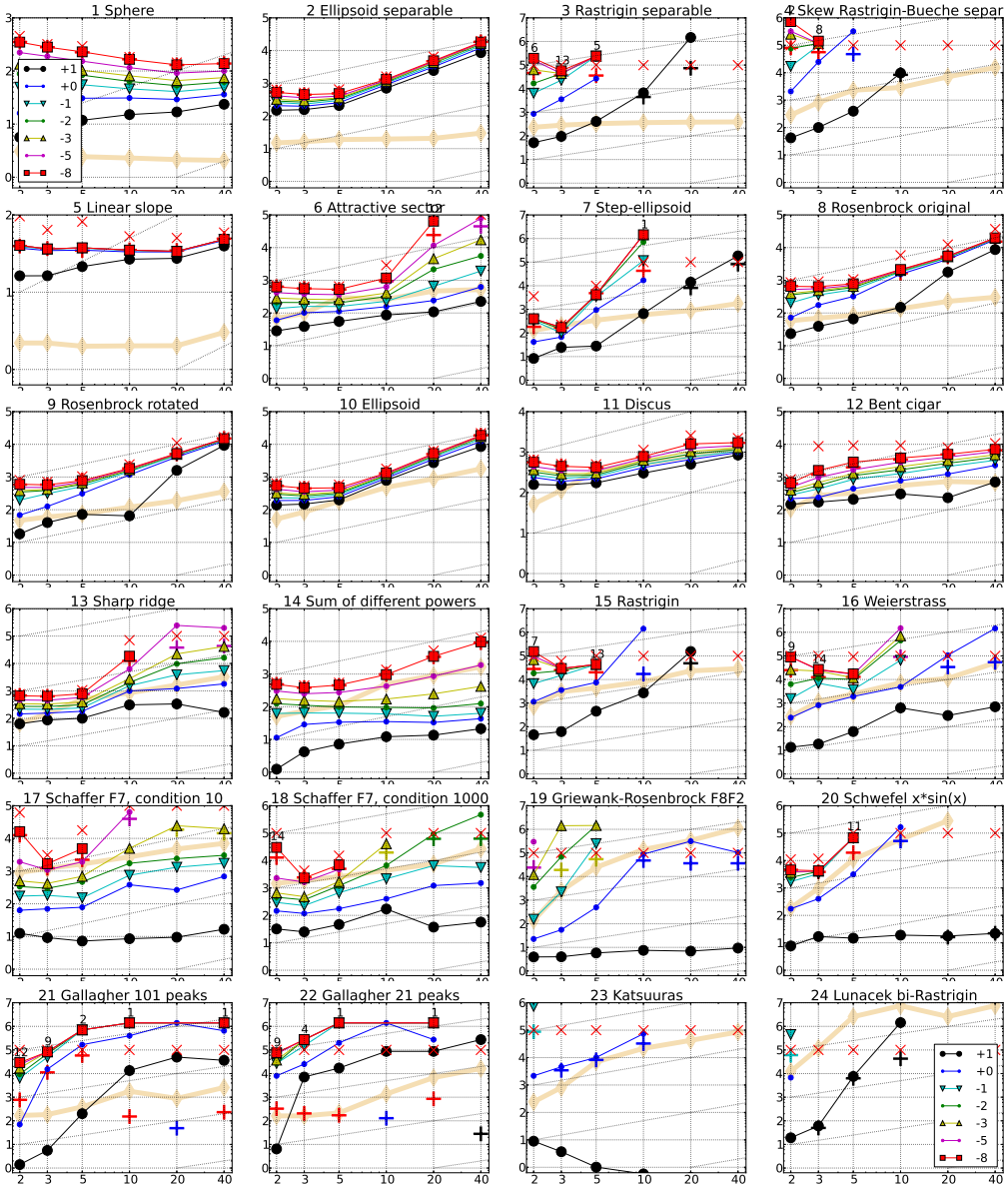

Figure 5.8: Expected number of f-evaluations (ERT, with lines, see legend) to reach $f_{\text{opt}} + \Delta f$, median number of f-evaluations to reach the most difficult target that was reached at least once ($+$) and maximum number of f-evaluations in any trial ($\times$), all divided by dimension and plotted as $\log_{10}$ values versus dimension. Shown are $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$. Numbers above ERT-symbols indicate the number of successful trials. The light thick line with diamonds indicates the respective best result from BBOB-2009 for $\Delta f = 10^{-8}$. Horizontal lines mean linear scaling, slanted grid lines depict quadratic scaling.

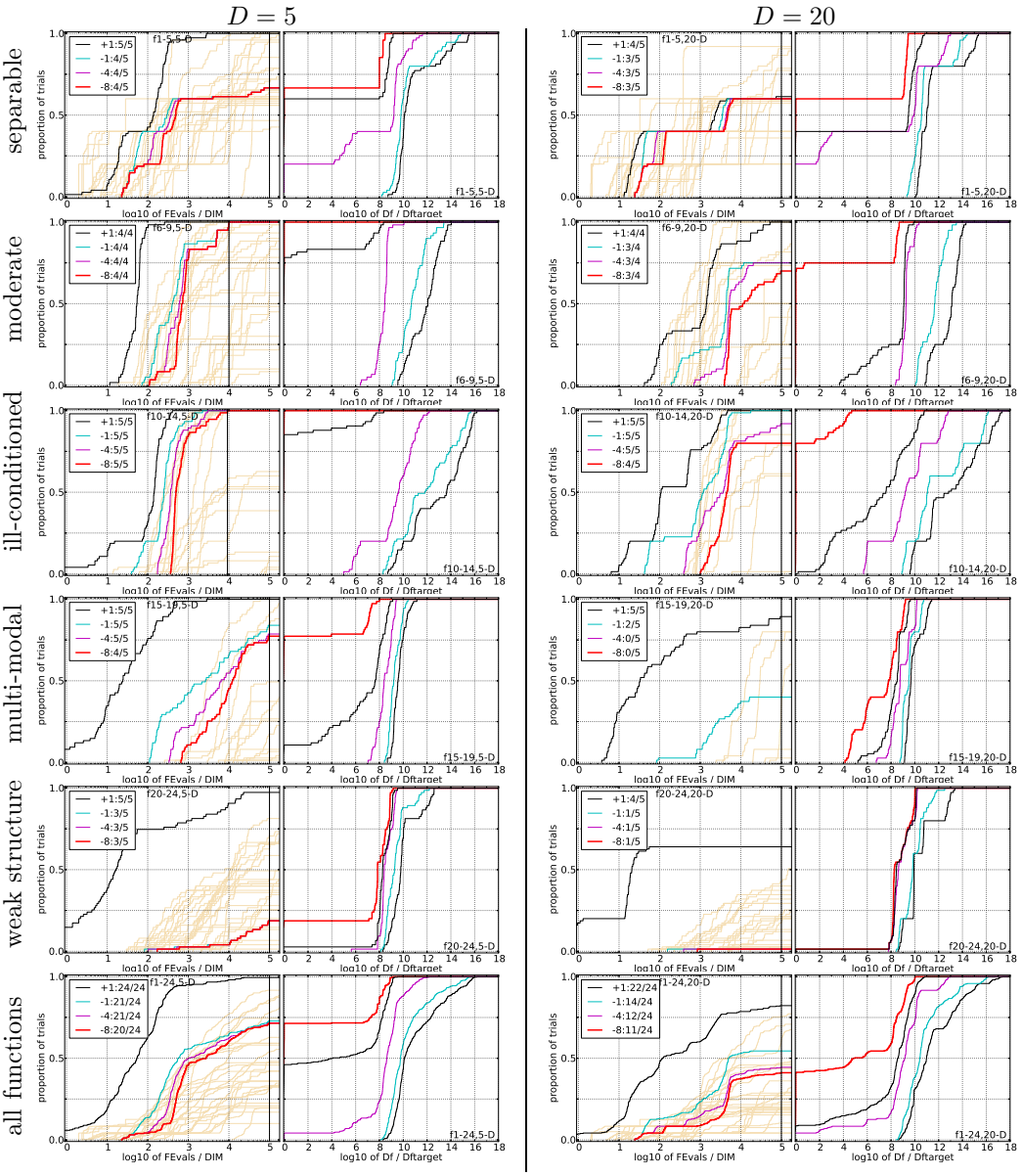

Figure 5.9: Empirical cumulative distribution functions (ECDFs), plotting the fraction of trials with an outcome not larger than the respective value on the x-axis. Left subplots: ECDF of number of function evaluations (FEvals) divided by search space dimension D, to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. Right subplots: ECDF of the best achieved Δf divided by 10^{-8} for running times of $D, 10\,D, 100\,D, \ldots$ function evaluations (from right to left cycling black-cyan-magenta). The thick red line represents the most difficult target value $f_{\text{opt}} + 10^{-8}$. Legends indicate the number of functions that were solved in at least one trial. Light brown lines in the background show ECDFs for $\Delta f = 10^{-8}$ of all algorithms benchmarked during BBOB-2009.

5-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_1	11	12	12	12	12	12	15/15
	5.4(4)	13(4)	23(4)	42(5)	63(8)	85(10)	15/15
f_2	83	87	88	90	92	94	15/15
	13(5)	15(6)	17(5)	20(5)	22(5)	24(5)	15/15
f_3	716	1622	1637	1646	1650	1654	15/15
	2.9(3)	83(95)	735(792)	732(826)	730(805)	728(822)	5/15
f_4	809	1633	1688	1817	1886	1903	15/15
	2.4(0.6)	990(1092)	∞	∞	∞	∞ *5.0e5*	0/15
f_5	10	10	10	10	10	10	15/15
	11(3)	17(7)	19(8)	19(8)	19(8)	19(8)	15/15
f_6	114	214	281	580	1038	1332	15/15
	2.4(0.8)	2.6(0.5)	2.8(0.6)	2.2(0.2)	1.7(0.2)	1.8(0.2)	15/15
f_7	24	324	1171	1572	1572	1597	15/15
	5.9(2)	15(35)	15(21)	13(15)	13(15)	13(15)	15/15
f_8	73	273	336	391	410	422	15/15
	4.5(1)	5.9(3)	7.7(3)	8.3(2)	8.6(2)	8.9(2)	15/15
f_9	35	127	214	300	335	369	15/15
	10(3)	12(8)	12(6)	11(4)	10(3)	10(3)	15/15
f_{10}	349	500	574	626	829	880	15/15
	2.9(1)	2.5(0.6)	2.5(0.6)	2.8(0.6)	2.4(0.4)	2.5(0.4)	15/15
f_{11}	143	202	763	1177	1467	1673	15/15
	6.1(2)	5.4(2)	1.6(0.4)	1.3(0.2)	1.2(0.2)	1.2(0.2)	15/15
f_{12}	108	268	371	461	1303	1494	15/15
	10(5)	8.2(7)	11(7)	14(10)	6.4(5)	7.6(8)	15/15
f_{13}	132	195	250	1310	1752	2255	15/15
	3.8(1)	4.7(2)	4.8(2)	1.5(0.5)	1.7(0.5)	1.6(0.4)	15/15
f_{14}	10	41	58	139	251	476	15/15
	3.7(3)	4.1(1)	5.5(2)	5.2(0.8)	5.4(1)	4.3(0.6)	15/15
f_{15}	511	9310	19369	20073	20769	21359	14/15
	4.5(2)	4.0(4)	11(13)	11(13)	10(13)	10(12)	13/15
f_{16}	120	612	2662	10449	11644	12095	15/15
	2.6(3)	16(15)	7.0(7)	5.7(4)	7.2(5)	7.2(5)	15/15
f_{17}	5.2	215	899	3669	6351	7934	15/15
	7.0(9)	1.8(1.0)	0.84(0.3)	0.94(1)	1.5(1)	3.0(4)	15/15
f_{18}	103	378	3968	9280	10905	12469	15/15
	2.3(2)	2.3(2)	0.84(0.8)	0.93(0.8)	2.2(3)	2.8(3)	15/15
f_{19}	1	1	242	1.2e5	1.2e5	1.2e5	15/15
	29(25)	2471(3448)	5114(5405)	60(66)	∞	∞ *5.0e5*	0/15
f_{20}	16	851	38111	54470	54861	55313	14/15
	4.6(3)	18(24)	8.9(12)	6.2(7)	6.2(7)	6.1(7)	11/15
f_{21}	41	1157	1674	1705	1729	1757	14/15
	24(3)	723(879)	2115(2293)	2077(2226)	2049(2169)	2016(2186)	2/15
f_{22}	71	386	938	1008	1040	1068	14/15
	1209(3521)	2589(3235)	7465(9064)	6944(7439)	6729(6969)	6558(7026)	1/15
f_{23}	3.0	518	14249	31654	33030	34256	15/15
	1.7(2)	97(104)	∞	∞	∞	∞ *5.0e5*	0/15
f_{24}	1622	2.2e5	6.4e6	9.6e6	1.3e7	1.3e7	3/15
	23(30)	∞	∞	∞	∞	∞ *5.0e5*	0/15

20-D

Δf	1e+1	1e+0	1e-1	1e-3	1e-5	1e-7	#succ
f_1	43	43	43	43	43	43	15/15
	7.9(1)	14(2)	19(2)	31(2)	42(3)	55(4)	15/15
f_2	385	386	387	390	391	393	15/15
	128(31)	164(41)	185(38)	221(34)	235(46)	244(52)	15/15
f_3	5066	7626	7635	7643	7646	7651	15/15
	5819(6711)	∞	∞	∞	∞	∞ *2.0e6*	0/15
f_4	4722	7628	7666	7700	7758	1.4e5	9/15
	∞	∞	∞	∞	∞	∞ *2.0e6*	0/15
f_5	41	41	41	41	41	41	15/15
	13(4)	16(3)	17(4)	17(4)	17(4)	17(4)	15/15
f_6	1296	2343	3413	5220	6728	8409	15/15
	1.7(0.6)	2.1(2)	3.9(4)	17(26)	34(24)	63(80)	12/15
f_7	1351	4274	9503	16524	16524	16969	15/15
	211(262)	∞	∞	∞	∞	∞ *2.0e6*	0/15
f_8	2039	3871	4040	4219	4371	4484	15/15
	17(7)	23(5)	24(5)	25(5)	25(4)	25(4)	15/15
f_9	1716	3102	3277	3455	3594	3727	15/15
	19(5)	26(4)	28(4)	29(4)	28(4)	28(4)	15/15
f_{10}	7413	8661	10735	14920	17073	17476	15/15
	7.4(4)	8.2(2)	7.7(2)	6.2(0.7)	5.7(0.7)	5.8(0.7)	15/15
f_{11}	1002	2228	6278	9762	12285	14831	15/15
	10(0.9)	5.9(1)	2.4(0.5)	2.1(0.4)	2.0(0.6)	2.0(0.8)	15/15
f_{12}	1042	1938	2740	4140	12407	13827	15/15
	4.5(0.2)	13(11)	15(11)	15(9)	6.6(3)	6.8(3)	15/15
f_{13}	652	2021	2751	18749	24455	30201	15/15
	10(16)	12(12)	28(17)	24(26)	200(202)	∞ *2.0e6*	0/15
f_{14}	75	239	304	932	1648	15661	15/15
	3.6(1.0)	2.8(0.5)	3.3(0.4)	5.3(0.6)	10(2)	2.8(0.8)	15/15
f_{15}	30378	1.5e5	3.1e5	3.2e5	4.5e5	4.6e5	15/15
	101(127)	∞	∞	∞	∞	∞ *2.0e6*	0/15
f_{16}	1384	27265	77015	1.9e5	2.0e5	2.2e5	15/15
	4.3(2)	78(86)	∞	∞	∞	∞ *2.0e6*	0/15
f_{17}	63	1030	4005	30677	56288	80472	15/15
	3.0(2)	5.2(11)	6.5(5)	16(16)	∞	∞ *2.0e6*	0/15
f_{18}	621	3972	19561	67569	1.3e5	1.5e5	15/15
	1.2(0.4)	6.2(9)	6.7(8)	∞	∞	∞ *2.0e6*	0/15
f_{19}	1	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
	139(36)	6.3e6(7e6)	∞	∞	∞	∞ *2.0e6*	0/15
f_{20}	82	46150	3.1e6	5.5e6	5.6e6	5.6e6	14/15
	4.3(1)	∞	∞	∞	∞	∞ *2.0e6*	0/15
f_{21}	561	6541	14103	14643	15567	17589	15/15
	1782(3563)	4281(4740)	∞	∞	∞	∞ *2.0e6*	0/15
f_{22}	467	5580	23491	24948	26847	1.3e5	12/15
	3750(6427)	986(1434)	1192(1362)	1123(1303)	1043(1192)	208(230)	1/15
f_{23}	3.2	1614	67457	4.9e5	8.1e5	8.4e5	15/15
	2.7(3)	∞	∞	∞	∞	∞ *2.0e6*	0/15
f_{24}	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
	∞	∞	∞	∞	∞	∞ *2.0e6*	0/15

Table 5.8: Expected running time (ERT in number of function evaluations) divided by the best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values for functions f_1–f_{24}. The median number of conducted function evaluations is additionally given in *italics*, if $\mathrm{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\mathrm{opt}} + 10^{-8}$.

5.4.4 Cu-Simple-(μ, λ)-MSC-ES vs Simple-(μ, λ)-MSC-ES

The goal of the comparison is to verify if the cumulative parameter updating rule actually is better. The straight forward comparison is shown in the following three four graphs. Figure 5.10 plots the expected function evaluation against dimension for both the competitive ESs. However, this figure is useless because both of the ES algorithms are not going to find the global optimum within the evaluation budget. There is no effective comparisons in most of the subfigures.

In Figure 5.11, more information on ERT comparison is given. The logarithm of the ratio between two competitive ESs are plotted against decreasing target precision. Most of the curves in each subfigures are quite noisy. If most of the curves in one subfigure are roughly below 0, then we could consider that the newly purposed algorithm is performing better than the original algorithm in that subfigure. From figure 5.11, the cumulative variant actually dominates the test functions $f_5, f_6, f_7, f_8, f_{14}, f_{17}$, and gets worse in $f_1, f_2, f_{15}, f_{19}, f_{20}, f_{22}, f_{23}$. There is no decision for the rest functions. Figure 5.12 changes to another manner to compare the ERTs and basically tells the same story as Figure 5.11.

Then the most important characteristic is shown in Figure 5.13. This the comparison of empirical cumulative distributions of function evaluations. Given a fixed function evaluation number, the bigger size of the area under the distribution curve, the better performance a optimizer would have. For target precision $10^{\{1,-1,-4\}}$, the distribution curve for the new ES variant is increasing faster than the standard MSC-ES in every dimension and every group of th functions. Finally, the specific ERT numbers under precision 10^{-8} are compared in Table 5.9.

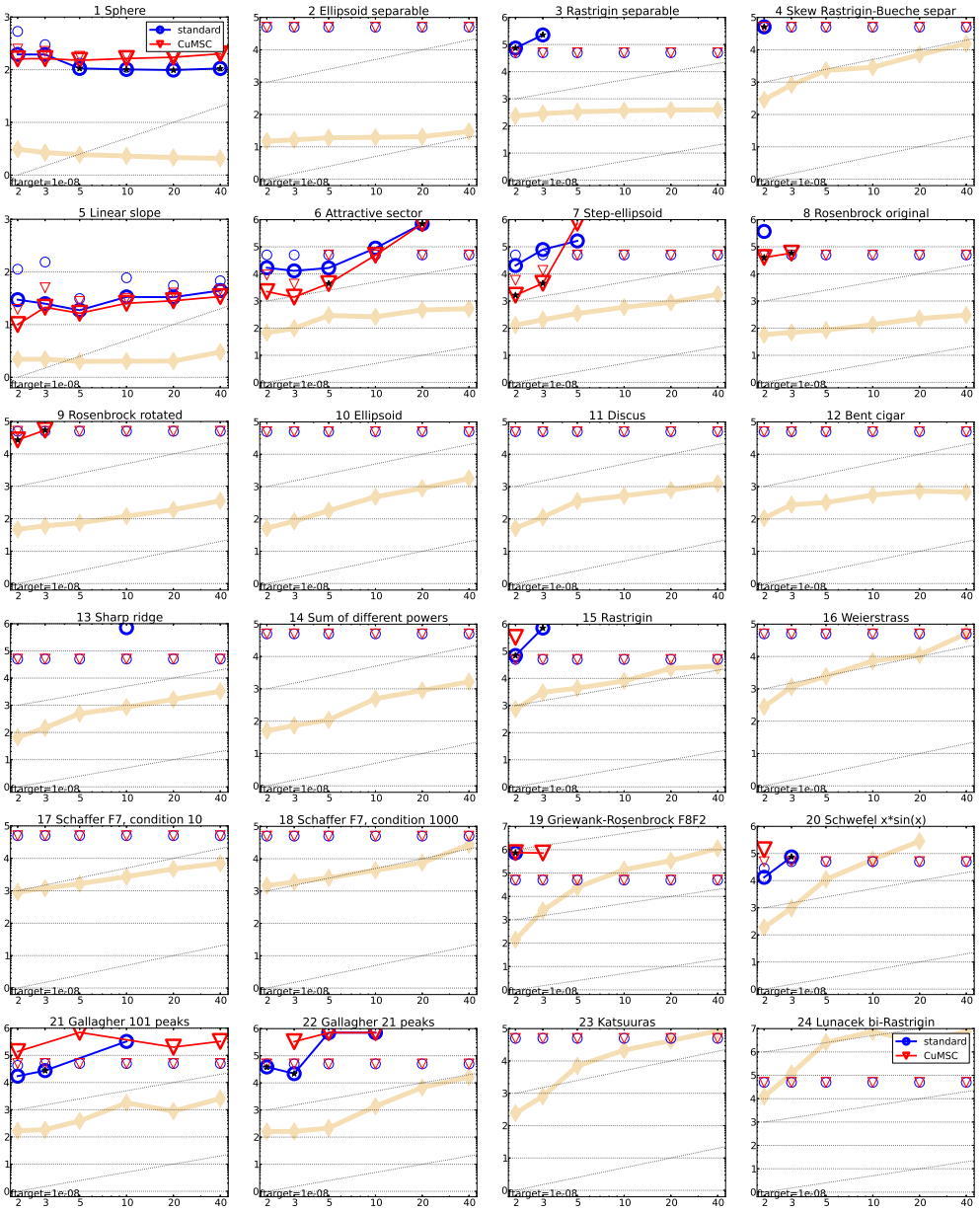

Figure 5.10: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function 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: $\circ$:Simple-(μ, λ)-MSC-ES, $\triangledown$:Cu-Simple-(μ, λ)-MSC-ES

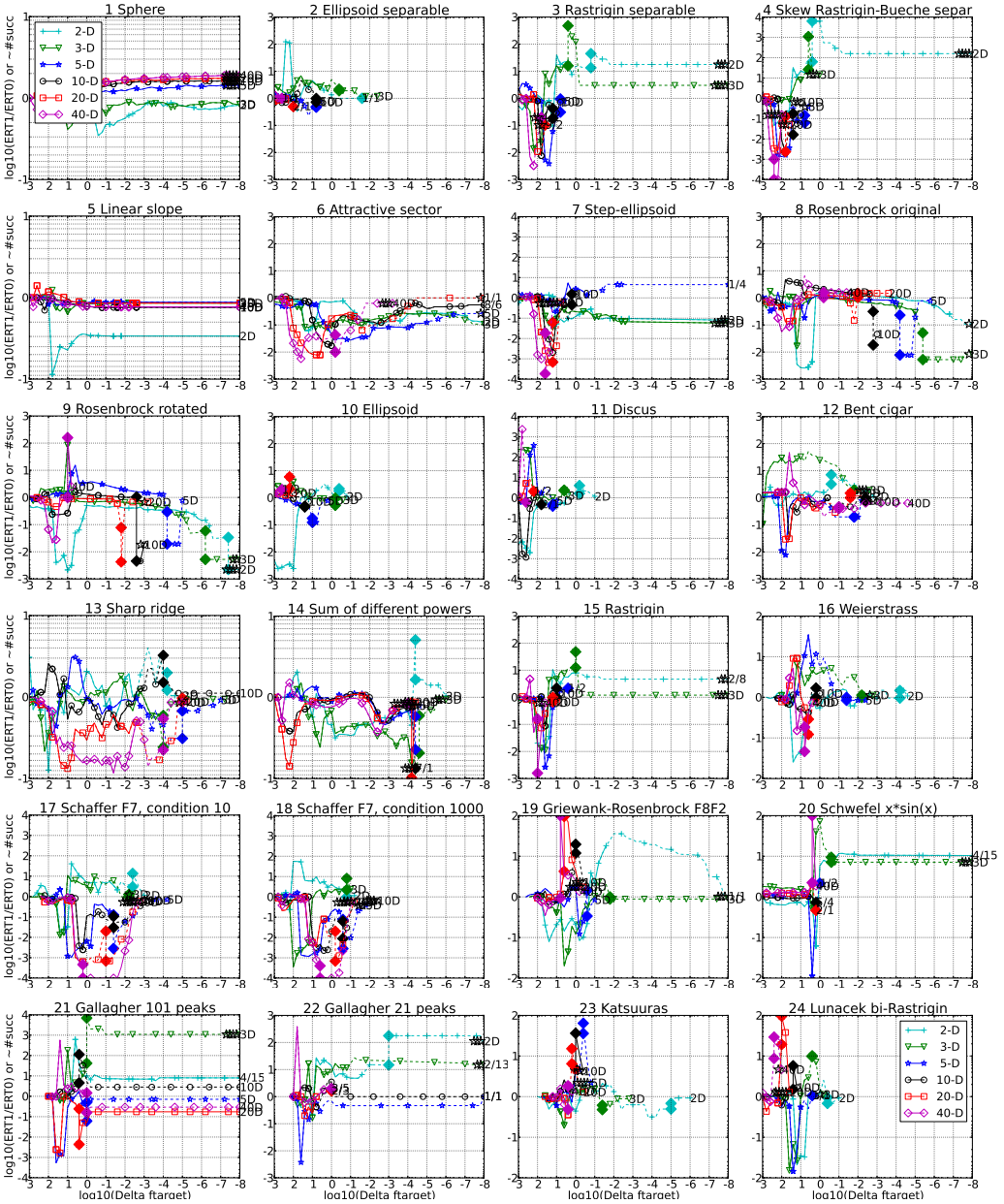

Figure 5.11: Ratio of ERT for Cu-Simple-(μ, λ)-MSC-ES over ERT for Simple-(μ, λ)-MSC-ES versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇. Ratios $< 10^0$ indicate an advantage of Cu-Simple-(μ, λ)-MSC-ES, smaller values are always better. The line becomes dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall number of f-evaluations for the same algorithm on this function. Filled symbols indicate the best achieved Δf-value of one algorithm (ERT is undefined to the right). The dashed line continues as the fraction of successful trials of the other algorithm, where 0 means 0% and the y-axis limits mean 100%, values below zero for Cu-Simple-(μ, λ)-MSC-ES. The line ends when no algorithm reaches Δf anymore. The number of successful trials is given, only if it was in $\{1 \ldots 9\}$ for Cu-Simple-(μ, λ)-MSC-ES (1^{st} number) and non-zero for Simple-(μ, λ)-MSC-ES (2^{nd} number). Results are significant with $p = 0.05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni correction within each figure.

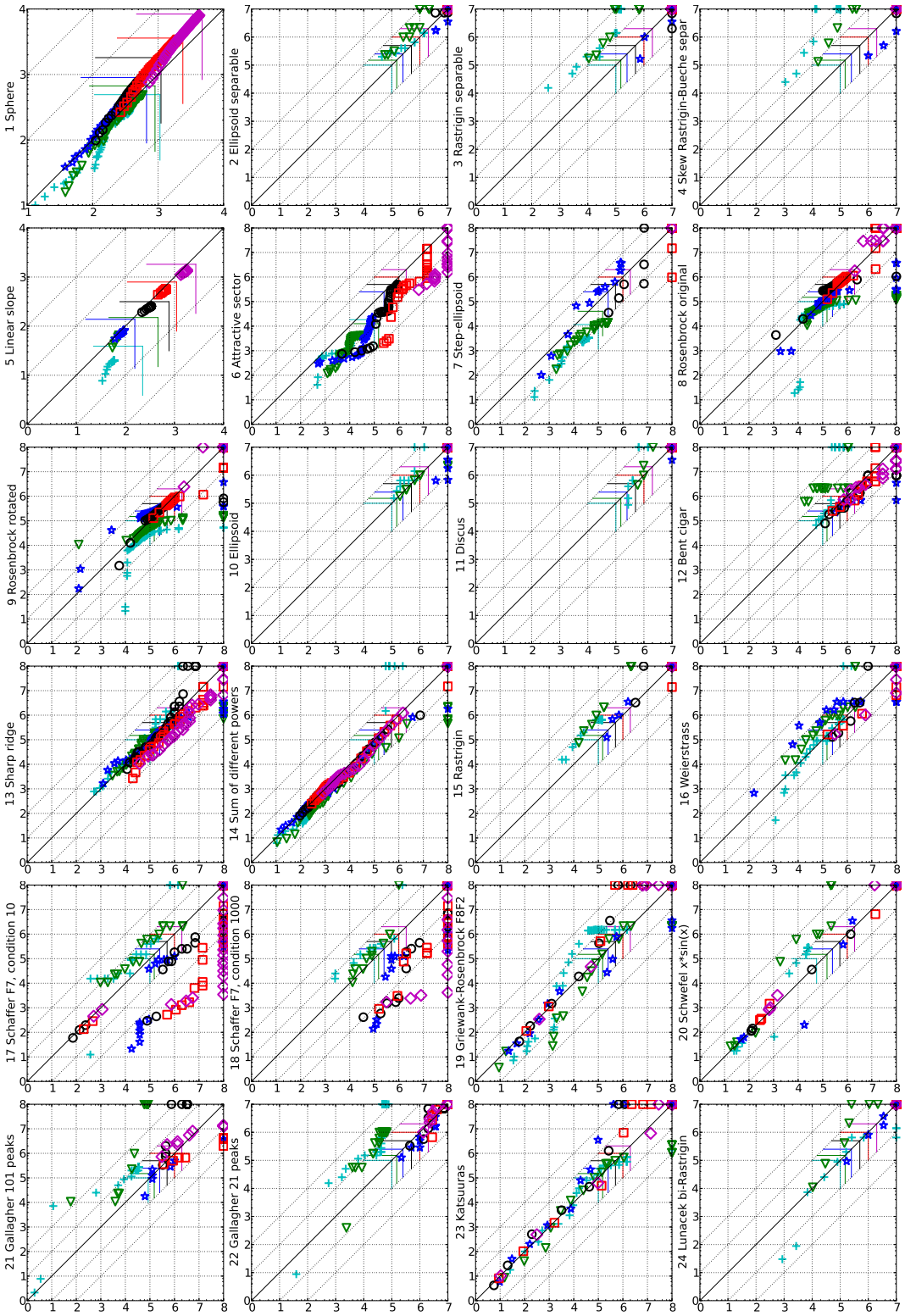

Figure 5.12: Expected running time (ERT in $\log_{10}$ of number of function evaluations) of Simple-(μ, λ)-MSC-ES (x-axis) versus Cu-Simple-(μ, λ)-MSC-ES (y-axis) for 46 target values $\Delta f \in [10^{-8}, 10]$ in each dimension on functions f_1–f_{24}. Markers on the upper or right edge indicate that the target value was never reached. Markers represent dimension: 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40:◇.

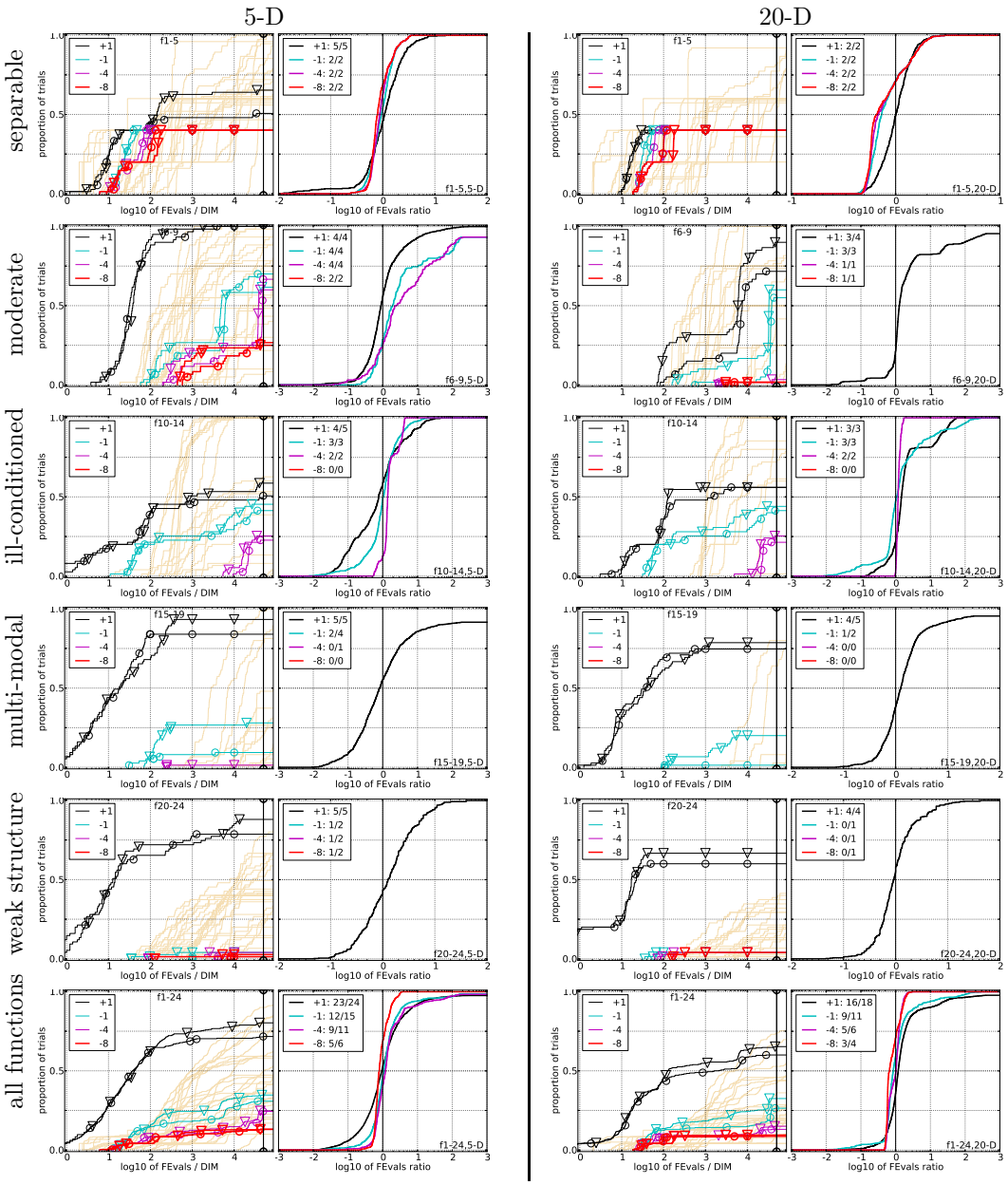

Figure 5.13: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for Simple-(μ, λ)-MSC-ES ($\circ$) and Cu-Simple-(μ, λ)-MSC-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 Simple-(μ, λ)-MSC-ES divided by Cu-Simple-(μ, λ)-MSC-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 (Simple-(μ, λ)-MSC-ES 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: simple	3.5(2)	11(4)	20(4)	**29**(5)$^{\star 2}$	**39**(7)$^{\star 3}$	15/15
2: Cu	3.5(3)	13(5)	25(6)	40(9)	55(10)	15/15
f_2	83	88	90	92	94	15/15
1: simple	43620(47961)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	20624(23019)	∞	∞	∞	∞*2.5e5*	0/15
f_3	716	1637	1646	1650	1654	15/15
1: simple	1020(1223)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	234(350)	∞	∞	∞	∞*2.5e5*	0/15
f_4	809	1688	1817	1886	1903	15/15
1: simple	1237(1390)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	271(463)	∞	∞	∞	∞*2.5e5*	0/15
f_5	10	10	10	10	10	15/15
1: simple	6.1(3)	9.2(4)	9.4(4)	9.4(4)	9.4(4)	15/15
2: Cu	5.5(3)	8.1(4)	8.2(4)	8.2(4)	8.2(4)	15/15
f_6	114	281	580	1038	1332	15/15
1: simple	4.5(2)	110(148)	70(117)	59(102)	61(101)	12/15
2: Cu	2.6(1)	3.0(2)	5.7(3)	7.3(3)	16(3)	14/15
f_7	24	1171	1572	1572	1597	15/15
1: simple	20(9)	672(766)	525(545)	525(578)	517(542)	4/15
2: Cu	4.4(4)	1568(1584)	2360(2504)	2360(2703)	2323(2505)	1/15
f_8	73	336	391	410	422	15/15
1: simple	26(22)	357(386)	644(363)	∞	∞*2.5e5*	0/15
2: Cu	13(8)	334(380)	540(335)	2247(2133)	∞*2.5e5*	0/15
f_9	35	214	300	335	369	15/15
1: simple	3.6(1)	316(587)	652(421)	∞	∞*2.5e5*	0/15
2: Cu	5.0(2)	889(1172)	970(837)	11171(10994)	∞*2.5e5*	0/15
f_{10}	349	574	626	829	880	15/15
1: simple	10579(11806)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	1848(1800)	∞	∞	∞	∞*2.5e5*	0/15
f_{11}	143	763	1177	1467	1673	15/15
1: simple	∞	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	24545(27173)	∞	∞	∞	∞*2.5e5*	0/15
f_{12}	108	371	461	1303	1494	15/15
1: simple	2968(3466)	9445(11130)	∞	∞	∞*2.5e5*	0/15
2: Cu	2026(3466)	1857(2361)	∞	∞	∞*2.5e5*	0/15
f_{13}	132	250	1310	1752	2255	15/15
1: simple	9.5(16)	253(241)	391(385)	2119(2212)	∞*2.5e5*	0/15
2: Cu	13(14)	184(160)	293(318)	658(700)	1600(1802)	0/15
f_{14}	10	58	139	251	476	15/15
1: simple	1.6(2)	3.3(1)	30(19)	∞	∞*2.5e5*	0/15
2: Cu	2.2(3)	3.4(1)	14(9)	∞	∞*2.5e5*	0/15
f_{15}	511	19369	20073	20769	21359	14/15
1: simple	429(490)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	247(489)	∞	∞	∞	∞*2.5e5*	0/15
f_{16}	120	2662	10449	11644	12095	15/15
1: simple	1.3(1)	1321(1502)	∞	∞	∞*2.5e5*	0/15
2: Cu	5.6(5)	1315(1408)	∞	∞	∞*2.5e5*	0/15
f_{17}	5.2	899	3669	6351	7934	15/15
1: simple	3439(16)	417(556)	∞	∞	∞*2.5e5*	0/15
2: Cu	4.1(5)	102(139)	954(1056)	∞	∞*2.5e5*	0/15
f_{18}	103	3968	9280	10905	12469	15/15
1: simple	880(1209)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	1.4(1)	**55**(94)*	∞	∞	∞*2.5e5*	0/15
f_{19}	1	242	1.2e5	1.2e5	1.2e5	15/15
1: simple	22(16)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	18(20)	14858(16515)	∞	∞	∞*2.5e5*	0/15
f_{20}	16	38111	54470	54861	55313	14/15
1: simple	2.9(2)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	3.3(3)	∞	∞	∞	∞*2.5e5*	0/15
f_{21}	41	1674	1705	1729	1757	14/15
1: simple	1526(3050)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	437(3)	2090(2314)	2053(2419)	2025(2242)	1992(2277)	1/15
f_{22}	71	938	1008	1040	1068	14/15
1: simple	3084(5282)	3738(4199)	3488(3844)	3389(3485)	3312(3630)	1/15
2: Cu	1762(3521)	1734(2265)	1619(2045)	1577(1801)	1545(1755)	1/15
f_{23}	3.0	14249	31654	33030	34256	15/15
1: simple	3.2(2)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	2.0(3)	∞	∞	∞	∞*2.5e5*	0/15
f_{24}	1622	6.4e6	9.6e6	1.3e7	1.3e7	3/15
1: simple	105(154)	∞	∞	∞	∞*2.5e5*	0/15
2: Cu	58(88)	∞	∞	∞	∞*2.5e5*	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: simple	6.2(1)	**14**(2)$^{\star 3}$	**23**(2)$^{\star 3}$	**32**(2)$^{\star 3}$	**41**(3)$^{\star 3}$	15/15
2: Cu	6.2(2)	22(2)	39(2)	55(3)	72(3)	15/15
f_2	385	387	390	391	393	15/15
1: simple	∞	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	∞	∞	∞	∞	∞*1.0e6*	0/15
f_3	5066	7635	7643	7646	7651	15/15
1: simple	∞	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	∞	∞	∞	∞	∞*1.0e6*	0/15
f_4	4722	7666	7700	7758	1.4e5	9/15
1: simple	∞	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	∞	∞	∞	∞	∞*1.0e6*	0/15
f_5	41	41	41	41	41	15/15
1: simple	11(5)	16(7)	16(7)	16(7)	16(7)	15/15
2: Cu	11(3)	14(3)	14(3)	14(3)	14(3)	15/15
f_6	1296	3413	5220	6728	8409	15/15
1: simple	189(387)	594(651)	2688(3209)	2088(2527)	1674(1903)	1/15
2: Cu	**1.6**(0.5)$^{\star 2}$	136(164)	367(452)	2088(2378)	1672(1724)	1/15
f_7	1351	9503	16524	16524	16969	15/15
1: simple	∞	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	**725**(837)$^{\star 2}$	∞	∞	∞	∞*1.0e6*	0/15
f_8	2039	4040	4219	4371	4484	15/15
1: simple	75(12)	189(21)	3444(3674)	∞	∞*1.0e6*	0/15
2: Cu	61(17)	220(127)	∞	∞	∞*1.0e6*	0/15
f_9	1716	3277	3455	3594	3727	15/15
1: simple	80(30)	262(154)	∞	∞	∞*1.0e6*	0/15
2: Cu	75(32)	230(155)	4285(4486)	∞	∞*1.0e6*	0/15
f_{10}	7413	10735	14920	17073	17476	15/15
1: simple	∞	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	∞	∞	∞	∞	∞*1.0e6*	0/15
f_{11}	1002	6278	9762	12285	14831	15/15
1: simple	∞	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	∞	∞	∞	∞	∞*1.0e6*	0/15
f_{12}	1042	2740	4140	12407	13827	15/15
1: simple	241(480)	5111(5657)	∞	∞	∞*1.0e6*	0/15
2: Cu	242(480)	1461(1825)	∞	∞	∞*1.0e6*	0/15
f_{13}	652	2751	18749	24455	30201	15/15
1: simple	31(41)	102(104)	390(373)	603(613)	∞*1.0e6*	0/15
2: Cu	**4.1**(0.7)$^{\star 2}$	47(54)	109(107)	583(675)	∞*1.0e6*	0/15
f_{14}	75	304	932	1648	15661	15/15
1: simple	3.7(2)	3.3(1)	31(4)	∞	∞*1.0e6*	0/15
2: Cu	3.3(0.8)	3.5(0.5)	**19**(6)$^{\star 3}$	∞	∞*1.0e6*	0/15
f_{15}	30378	3.1e5	3.2e5	4.5e5	4.6e5	15/15
1: simple	∞	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	461(551)	∞	∞	∞	∞*1.0e6*	0/15
f_{16}	1384	77015	1.9e5	2.0e5	2.2e5	15/15
1: simple	112(361)	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	118(362)	∞	∞	∞	∞*1.0e6*	0/15
f_{17}	63	4005	30677	56288	80472	15/15
1: simple	3.2(2)	3496(4119)	∞	∞	∞*1.0e6*	0/15
2: Cu	2.1(1)	**69**(127)$^{\star 2}$	∞	∞	∞*1.0e6*	0/15
f_{18}	621	19561	67569	1.3e5	1.5e5	15/15
1: simple	251(806)	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	1.5(1)	**209**(231)$^{\star 2}$	∞	∞	∞*1.0e6*	0/15
f_{19}	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
1: simple	117(56)	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	114(48)	∞	∞	∞	∞*1.0e6*	0/15
f_{20}	82	3.1e6	5.5e6	5.6e6	5.6e6	14/15
1: simple	3.4(1)	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	3.9(1)	∞	∞	∞	∞*1.0e6*	0/15
f_{21}	561	14103	14643	15567	17589	15/15
1: simple	648(891)	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	648(891)	284(355)	273(341)	257(321)	228(284)	3/15
f_{22}	467	23491	24948	26847	1.3e5	12/15
1: simple	5892(7498)	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	1429(2142)	∞	∞	∞	∞*1.0e6*	0/15
f_{23}	3.2	67457	4.9e5	8.1e5	8.4e5	15/15
1: simple	2.8(2)	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	2.5(3)	∞	∞	∞	∞*1.0e6*	0/15
f_{24}	1.3e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
1: simple	∞	∞	∞	∞	∞*1.0e6*	0/15
2: Cu	∞	∞	∞	∞	∞*1.0e6*	0/15

Table 5.9: 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 different Δf values. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. 1:simple is Simple-(μ, λ)-MSC-ES and 2:Cu is Cu-Simple-(μ, λ)-MSC-ES. Bold entries are statistically significantly better compared to the other algorithm, with $p = 0.05$ or $p = 10^{-k}$ where $k \in \{2, 3, 4, \ldots\}$ is the number following the $\star$ symbol, with Bonferroni correction of 48. A $\downarrow$ indicates the same tested against the best BBOB-2009.

5.4.5 Noisy$-(\mu/\mu_{\mathbf{w}}, \lambda_{\mathbf{m}})$-CMA-ES vs $(\mu/\mu_{\mathbf{w}}, \lambda_{\mathbf{m}})$-CMA-ES

Results from experiments according to [22] on the benchmark functions given in [19, 23] are presented in Figures 5.14, 5.15, 5.16, 5.17 and Table 5.10. The **expected running time (**ERT**)**, used in the figures and table, depends on a given target function value, $f_{\mathrm{t}} = f_{\mathrm{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_{t}, summed over all trials and divided by the number of trials that actually reached f_{t} [22, 33]. **Statistical significance** is tested with the rank-sum test for a given target Δf_{t} using, for each trial, either the number of needed function evaluations to reach Δf_{t} (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration if available.
Figure 5.14 plots the expected function evaluation against dimension for both the competitive ESs. The "noisy" mirrored sampling method actually is getting worse on function $f_{10}, f_{11}, f_{12}, f_{13}, f_{14}$ under the target precision 10^{-8}.

In Figure 5.15, the observation above is presented in details. The logarithm of the ratio between two competitive ESs are plotted against decreasing target precision. Most of the curves in each subfigures are quite noisy. If most of the curves in one subfigure are roughly below 0, then we could consider that the newly purposed algorithm is performing better than the original algorithm in that subfigure. From figure 5.15, the "noisy" sampling method gets worse on functions $f_{10}, f_{11}, f_{12}, f_{13}, f_{14}$. There is no decision for the rest functions. Figure 5.16 changes to another manner to compare the ERTs and basically tells the same story as Figure 5.15.

Then the most important characteristic is shown in Figure 5.17. The distribution curves for the "noisy" mirrored sampling method are roughly the same as that of mirrored sampling method on most functions, indicating the same performance characteristics of two competitive ESs. Finally, the specific ERT numbers under precision 10^{-8} are compared in Table 5.10.

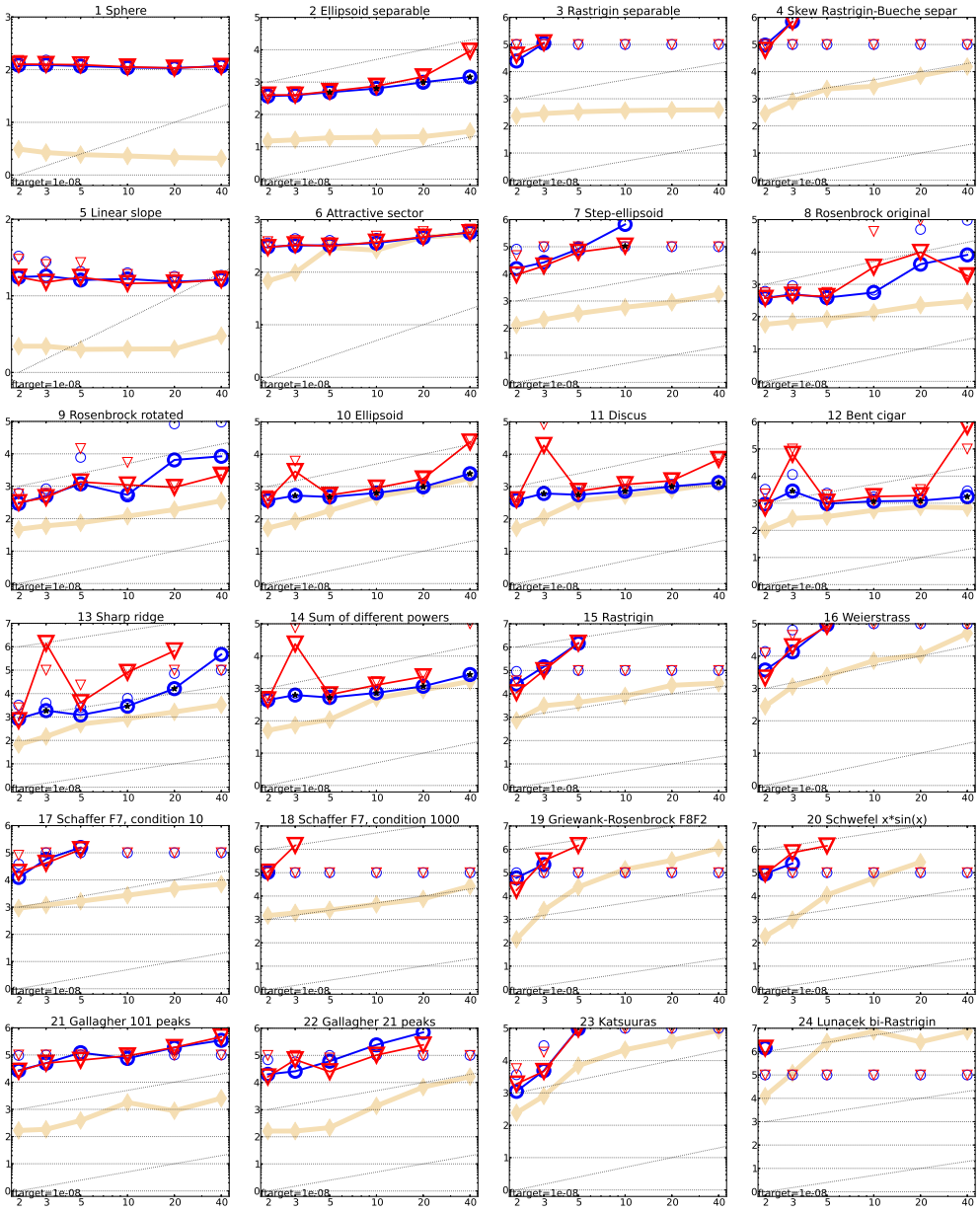

Figure 5.14: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function 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: ○:Mirrored sampling, ▽:"Noisy" mirrored sampling

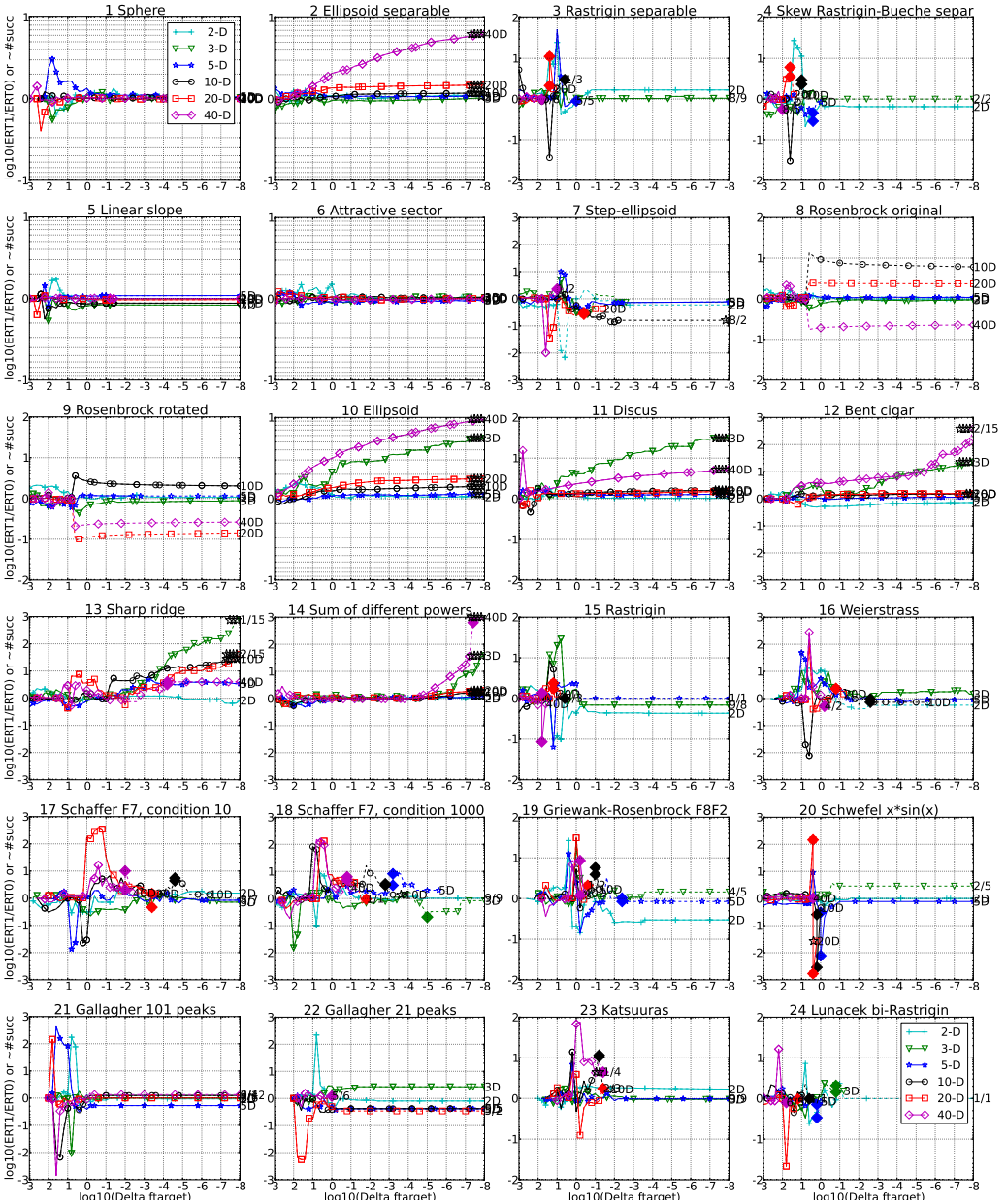

Figure 5.15: Ratio of ERT for "Noisy" mirrored sampling over ERT for Mirrored sampling versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇. Ratios $< 10^0$ indicate an advantage of "Noisy" mirrored sampling, smaller values are always better. The line becomes dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall number of f-evaluations for the same algorithm on this function. Filled symbols indicate the best achieved Δf-value of one algorithm (ERT is undefined to the right). The dashed line continues as the fraction of successful trials of the other algorithm, where 0 means 0% and the y-axis limits mean 100%, values below zero for "Noisy" mirrored sampling. The line ends when no algorithm reaches Δf anymore. The number of successful trials is given, only if it was in $\{1 \ldots 9\}$ for "Noisy" mirrored sampling (1^{st} number) and non-zero for Mirrored sampling (2^{nd} number). Results are significant with $p = 0.05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni correction within each figure.

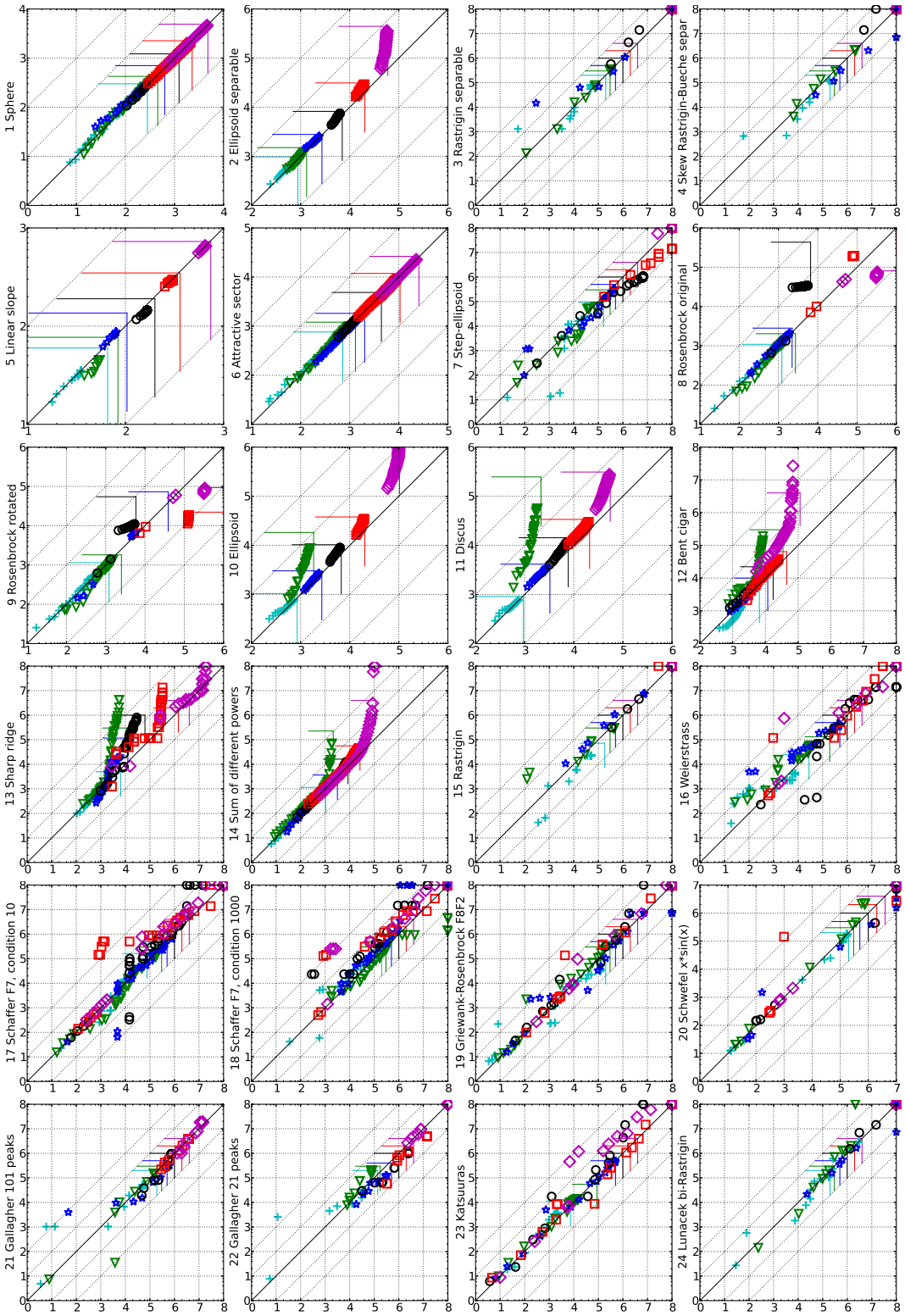

Figure 5.16: Expected running time (ERT in $\log_{10}$ of number of function evaluations) of Mirrored sampling (x-axis) versus "Noisy" mirrored sampling (y-axis) for 46 target values $\Delta f \in [10^{-8}, 10]$ in each dimension on functions f_1–f_{24}. Markers on the upper or right edge indicate that the target value was never reached. Markers represent dimension: 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40:◇.

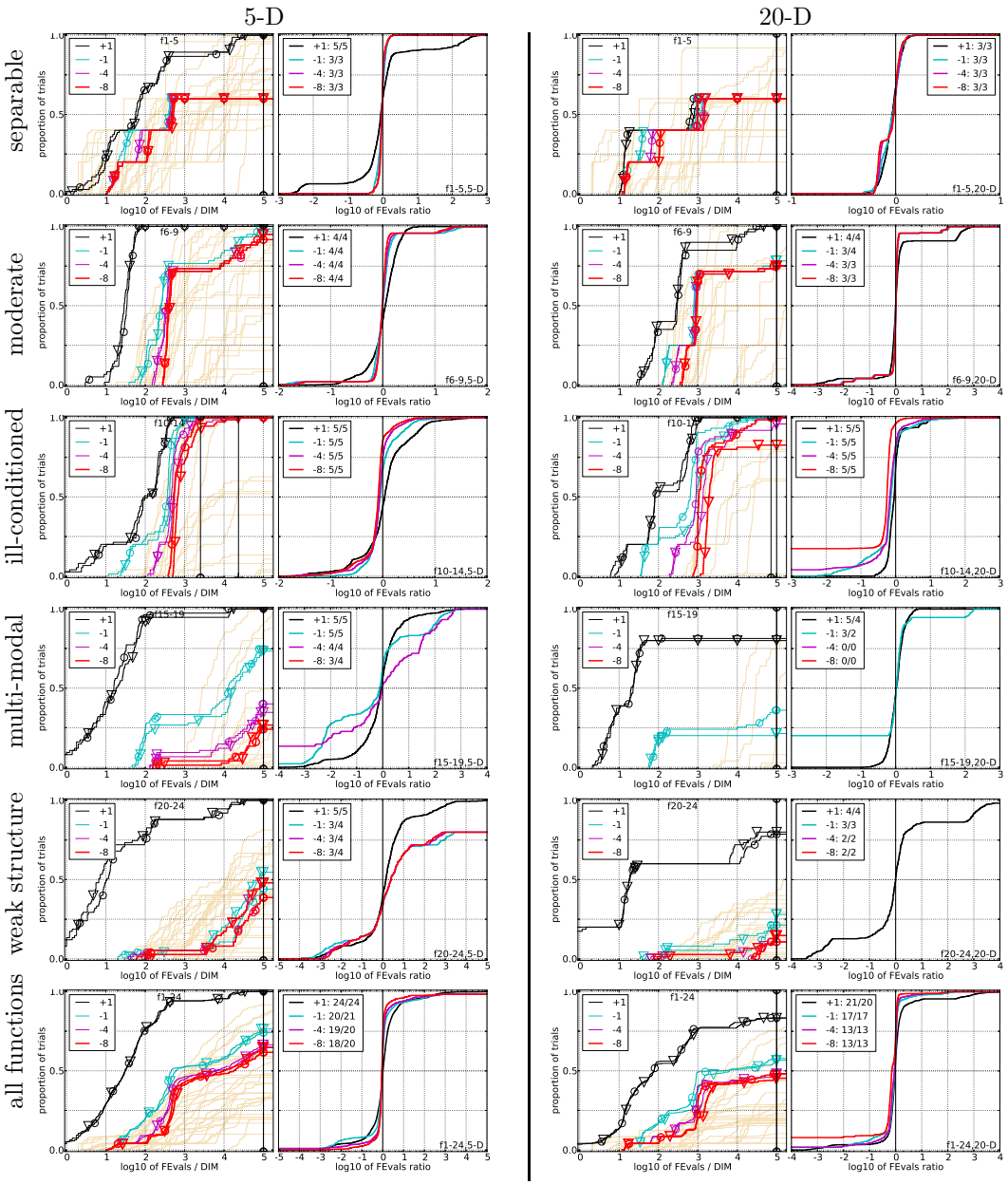

Figure 5.17: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for Mirrored sampling ($\circ$) and "Noisy" mirrored sampling ($\triangledown$). 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 Mirrored sampling divided by "Noisy" mirrored sampling, 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 (Mirrored 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: Mi	2.2(1)	12(3)	22(3)	33(2)	44(3)	15/15
2: NMi	3.6(3)	14(2)	24(4)	35(5)	46(6)	15/15
f_2	83	88	90	92	94	15/15
1: Mi	15(5)	22(2)	23(2)	24(2)	25(2)	15/15
2: NMi	18(6)	23(3)	25(2)	26(2)	27(2)	15/15
f_3	716	1637	1646	1650	1654	15/15
1: Mi	0.40(0.1)	∞	∞	∞	∞ *5.0e5*	0/15
2: NMi	21(50)	∞	∞	∞	∞ *5.0e5*	0/15
f_4	809	1688	1817	1886	1903	15/15
1: Mi	64(79)	∞	∞	∞	∞ *5.0e5*	0/15
2: NMi	39(50)	∞	∞	∞	∞ *5.0e5*	0/15
f_5	10	10	10	10	10	15/15
1: Mi	5.9(2)	8.0(2)	8.0(2)	8.0(2)	8.0(2)	15/15
2: NMi	6.2(3)	8.7(4)	8.7(4)	8.7(4)	8.7(4)	15/15
f_6	114	281	580	1038	1332	15/15
1: Mi	1.8(1.0)	1.9(0.5)	1.5(0.2)	1.2(0.2)	1.1(0.1)	15/15
2: NMi	1.7(0.7)	1.8(0.4)	1.4(0.4)	1.1(0.2)	1.1(0.1)	15/15
f_7	24	1171	1572	1572	1597	15/15
1: Mi	4.0(3)	171(231)	263(320)	263(315)	259(245)	10/15
2: NMi	4.2(2)	130(172)	202(186)	202(186)	199(187)	12/15
f_8	73	336	391	410	422	15/15
1: Mi	2.6(0.9)	3.7(0.6)	4.0(0.7)	4.2(0.6)	4.4(0.5)	15/15
2: NMi	2.7(1)	4.0(0.9)	4.4(0.8)	4.6(0.7)	4.9(0.7)	15/15
f_9	35	214	300	335	369	15/15
1: Mi	5.5(2)	24(53)	18(38)	17(34)	16(31)	15/15
2: NMi	4.2(1)	28(1)	21(1)	20(1)	18(1)	15/15
f_{10}	349	574	626	829	880	15/15
1: Mi	3.3(1)	3.2(0.8)	**3.2**(0.3)*	**2.6**(0.2)$^{\star 2}$	**2.6**(0.2)$^{\star 2}$	15/15
2: NMi	3.4(1)	3.6(0.3)	3.6(0.3)	2.9(0.2)	3.0(0.2)	15/15
f_{11}	143	763	1177	1467	1673	15/15
1: Mi	7.9(5)	**2.8**(0.4)*	**2.1**(0.2)$^{\star 2}$	**1.8**(0.2)$^{\star 2}$	**1.6**(0.2)$^{\star 2}$	15/15
2: NMi	10(4)	3.5(0.7)	2.6(0.4)	2.2(0.4)	2.1(0.3)	15/15
f_{12}	108	371	461	1303	1494	15/15
1: Mi	7.5(8)	7.0(7)	7.2(6)	3.1(3)	3.1(3)	15/15
2: NMi	8.8(7)	7.0(6)	7.7(4)	3.5(2)	3.5(2)	15/15
f_{13}	132	250	1310	1752	2255	15/15
1: Mi	4.9(5)	7.7(3)	1.9(0.8)	2.6(1)	2.3(1)	15/15
2: NMi	**2.0**(0.6)*	7.5(6)	4.2(6)	8.1(7)	8.3(11)	15/15
f_{14}	10	58	139	251	476	15/15
1: Mi	2.9(2)	3.1(1.0)	3.6(0.9)	5.5(1)	4.9(0.7)	15/15
2: NMi	1.8(2)	2.9(0.9)	3.7(1)	5.9(0.9)	5.5(0.7)	15/15
f_{15}	511	19369	20073	20769	21359	14/15
1: Mi	9.1(0.3)	378(445)	364(399)	352(391)	343(339)	1/15
2: NMi	21(68)	384(400)	370(436)	358(421)	348(363)	1/15
f_{16}	120	2662	10449	11644	12095	15/15
1: Mi	0.84(0.6)	27(47)	36(41)	40(40)	38(36)	10/15
2: NMi	41(2)	70(81)	34(38)	36(43)	35(32)	10/15
f_{17}	5.2	899	3669	6351	7934	15/15
1: Mi	8.4(14)	21(31)	18(32)	57(68)	99(114)	7/15
2: NMi	8.0(8)	17(40)	15(22)	51(55)	83(82)	8/15
f_{18}	103	3968	9280	10905	12469	15/15
1: Mi	43(0.5)	11(11)	98(108)	316(344)	∞ *5.0e5*	0/15
2: NMi	44(0.9)	17(27)	170(186)	∞	∞ *5.0e5*	0/15
f_{19}	1	242	1.2e5	1.2e5	1.2e5	15/15
1: Mi	18(20)	1767(2064)	∞	∞	∞ *5.0e5*	0/15
2: NMi	16(14)	1486(2064)	61(62)	61(67)	60(66)	1/15
f_{20}	16	38111	54470	54861	55313	14/15
1: Mi	3.2(2)	∞	∞	∞	∞ *5.0e5*	0/15
2: NMi	2.0(1)	186(203)	130(142)	129(134)	128(136)	1/15
f_{21}	41	1674	1705	1729	1757	14/15
1: Mi	1.2(0.6)	364(483)	357(440)	352(407)	347(458)	8/15
2: NMi	97(302)	193(222)	189(218)	187(199)	184(211)	11/15
f_{22}	71	938	1008	1040	1068	14/15
1: Mi	260(439)	324(377)	301(351)	292(340)	285(331)	12/15
2: NMi	118(176)	134(146)	124(136)	121(132)	118(128)	15/15
f_{23}	3.0	14249	31654	33030	34256	15/15
1: Mi	2.2(2)	13(20)	15(17)	15(17)	14(15)	9/15
2: NMi	2.8(3)	11(14)	15(18)	14(19)	14(16)	9/15
f_{24}	1622	6.4e6	9.6e6	1.3e7	1.3e7	3/15
1: Mi	14(40)	∞	∞	∞	∞ *5.0e5*	0/15
2: NMi	13(23)	∞	∞	∞	∞ *5.0e5*	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: Mi	6.7(0.8)	17(2)	26(3)	36(3)	45(2)	15/15
2: NMi	7.0(1)	17(1)	26(2)	36(2)	45(2)	15/15
f_2	385	387	390	391	393	15/15
1: Mi	**34**(5)*	**44**(3)$^{\star 3}$	**48**(1)$^{\star 3}$	**49**(1)$^{\star 3}$	**50**(1)$^{\star 3}$	15/15
2: NMi	43(8)	62(7)	68(4)	71(4)	73(4)	15/15
f_3	5066	7635	7643	7646	7651	15/15
1: Mi	∞	∞	∞	∞	∞ *2.0e6*	0/15
2: NMi	∞	∞	∞	∞	∞ *2.0e6*	0/15
f_4	4722	7666	7700	7758	1.4e5	9/15
1: Mi	∞	∞	∞	∞	∞ *2.0e6*	0/15
2: NMi	∞	∞	∞	∞	∞ *2.0e6*	0/15
f_5	41	41	41	41	41	15/15
1: Mi	6.2(0.8)	7.5(0.9)	7.5(0.9)	7.5(0.9)	7.5(0.9)	15/15
2: NMi	6.2(0.9)	7.2(0.7)	7.3(0.7)	7.3(0.7)	7.3(0.7)	15/15
f_6	1296	3413	5220	6728	8409	15/15
1: Mi	1.2(0.3)	0.90(0.2)	0.91(0.2)	0.97(0.2)	0.98(0.2)	15/15
2: NMi	1.2(0.2)	0.93(0.1)	0.89(0.1)	0.95(0.1)	0.99(0.1)	15/15
f_7	1351	9503	16524	16524	16969	15/15
1: Mi	127(139)	∞	∞	∞	∞ *2.0e6*	0/15
2: NMi	119(214)	1498(1578)	∞	∞	∞ *2.0e6*	0/15
f_8	2039	4040	4219	4371	4484	15/15
1: Mi	3.2(0.7)	20(0.4)	19(0.4)	19(0.4)	19(0.4)	15/15
2: NMi	3.5(0.9)	48(82)	46(78)	45(76)	44(74)	15/15
f_9	1716	3277	3455	3594	3727	15/15
1: Mi	4.3(1)	39(0.8)	37(0.7)	36(0.7)	35(0.7)	15/15
2: NMi	3.8(0.7)	4.8(0.4)	4.9(0.4)	4.9(0.4)	4.9(0.3)	15/15
f_{10}	7413	10735	14920	17073	17476	15/15
1: Mi	1.8(0.3)	**1.6**(0.1)$^{\star 3}$	**1.2**(0.1)$^{\star 3}$	**1.1**(0.0)$^{\star 3}$	**1.1**(0.0)$^{\star 3}$	15/15
2: NMi	2.2(0.8)	2.5(0.4)	2.1(0.2)	1.9(0.2)	1.9(0.2)	15/15
f_{11}	1002	6278	9762	12285	14831	15/15
1: Mi	**7.8**(2)$^{\star 3}$	**2.1**(0.1)$^{\star 3}$	**1.7**(0.1)$^{\star 3}$	**1.5**(0.1)$^{\star 3}$	**1.3**(0.1)$^{\star 3}$	15/15
2: NMi	11(2)	2.9(0.4)	2.3(0.2)	2.1(0.2)	2.0(0.2)	15/15
f_{12}	1042	2740	4140	12407	13827	15/15
1: Mi	2.7(3)	3.4(3)	3.5(2)	1.6(0.7)	1.7(0.8)	15/15
2: NMi	2.0(2)	5.0(4)	5.4(2)	2.5(0.8)	2.6(0.8)	15/15
f_{13}	652	2751	18749	24455	30201	15/15
1: Mi	4.4(7)	33(62)	13(15)	**12**(12)*	10(10)	15/15
2: NMi	1.9(0.4)	42(43)	32(50)	119(130)	141(199)	2/15
f_{14}	75	304	932	1648	15661	15/15
1: Mi	2.7(1)	2.7(0.3)	2.7(0.2)	5.4(0.6)	**1.2**(0.1)$^{\star 3}$	15/15
2: NMi	3.3(1.0)	2.8(0.5)	2.8(0.3)	6.4(1)	2.1(0.3)	15/15
f_{15}	30378	3.1e5	3.2e5	4.5e5	4.6e5	15/15
1: Mi	922(1020)	∞	∞	∞	∞ *2.0e6*	0/15
2: NMi	∞	∞	∞	∞	∞ *2.0e6*	0/15
f_{16}	1384	77015	1.9e5	2.0e5	2.2e5	15/15
1: Mi	0.44(0.2)	364(416)	∞	∞	∞ *2.0e6*	0/15
2: NMi	0.39(0.1)	∞	∞	∞	∞ *2.0e6*	0/15
f_{17}	63	4005	30677	56288	80472	15/15
1: Mi	1.9(1)	3.7(0.2)	121(130)	∞	∞ *2.0e6*	0/15
2: NMi	2.2(1)	125(250)	194(228)	∞	∞ *2.0e6*	0/15
f_{18}	621	19561	67569	1.3e5	1.5e5	15/15
1: Mi	0.83(0.2)	75(84)	∞	∞	∞ *2.0e6*	0/15
2: NMi	0.81(0.2)	306(407)	∞	∞	∞ *2.0e6*	0/15
f_{19}	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
1: Mi	115(46)	∞	∞	∞	∞ *2.0e6*	0/15
2: NMi	99(44)	∞	∞	∞	∞ *2.0e6*	0/15
f_{20}	82	3.1e6	5.5e6	5.6e6	5.6e6	14/15
1: Mi	3.6(0.7)	∞	∞	∞	∞ *2.0e6*	0/15
2: NMi	3.4(0.9)	∞	∞	∞	∞ *2.0e6*	0/15
f_{21}	561	14103	14643	15567	17589	15/15
1: Mi	513(536)	266(292)	256(279)	241(262)	213(232)	6/15
2: NMi	398(544)	269(284)	259(273)	244(269)	216(228)	6/15
f_{22}	467	23491	24948	26847	1.3e5	12/15
1: Mi	699(863)	601(668)	566(601)	526(559)	105(114)	2/15
2: NMi	125(167)	207(245)	195(230)	181(221)	36(43)	5/15
f_{23}	3.2	67457	4.9e5	8.1e5	8.4e5	15/15
1: Mi	1.5(1)	35(39)	∞	∞	∞ *2.0e6*	0/15
2: NMi	2.7(3)	26(30)	∞	∞	∞ *2.0e6*	0/15
f_{24}	1.3e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
1: Mi	∞	∞	∞	∞	∞ *2.0e6*	0/15
2: NMi	∞	∞	∞	∞	∞ *2.0e6*	0/15

Table 5.10: 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 different Δf values. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. 1:Mi is Mirrored sampling and 2:NMi is "Noisy" mirrored sampling. Bold entries are statistically significantly better compared to the other algorithm, with $p = 0.05$ or $p = 10^{-k}$ where $k \in \{2, 3, 4, \ldots\}$ is the number following the $\star$ symbol, with Bonferroni correction of 48. A $\downarrow$ indicates the same tested against the best BBOB-2009.

5.4.6 Orthogonal$1-(\mu/\mu_{\mathbf{w}},\lambda_{\mathbf{m}})$-CMA-ES vs $(\mu/\mu_{\mathbf{w}},\lambda_{\mathbf{m}})$-CMA-ES

Results from experiments according to [22] on the benchmark functions given in [19, 23] are presented in Figures 5.18, 5.19, 5.20, 5.21 and Table 5.11. The **expected running time (**ERT**)**, used in the figures and table, depends on a given target function value, $f_{\mathrm{t}} = f_{\mathrm{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_{t}, summed over all trials and divided by the number of trials that actually reached f_{t} [22, 33]. **Statistical significance** is tested with the rank-sum test for a given target Δf_{t} using, for each trial, either the number of needed function evaluations to reach Δf_{t} (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration if available.

Figure 5.18 plots the expected function evaluation against dimension for both the competitive ESs. The "noisy" mirrored sampling method actually is getting worse on function $f_7, f_{16}, f_{19}, f_{23}$ under the target precision 10^{-8}.

In Figure 5.19, the observation above is presented in details. The logarithm of the ratio between two competitive ESs are plotted against decreasing target precision. Most of the curves in each subfigures are quite noisy. If most of the curves in one subfigure are roughly below 0, then we could consider that the newly purposed algorithm is performing better than the original algorithm in that subfigure. From figure 5.19, the "noisy" sampling method gets worse on functions $f_2, f_5, f_6, f_{10}, f_{14}$. There is no decision for the rest functions. Figure 5.20 changes to another manner to compare the ERTs and basically tells the same story as Figure 5.19.

Then the most important characteristic is shown in Figure 5.21. The distribution curves for the "noisy" mirrored sampling method are roughly the same as that of mirrored sampling method on most functions, indicating the same performance characteristics of two competitive ESs. Finally, the specific ERT numbers under precision 10^{-8} are compared in Table 5.11.

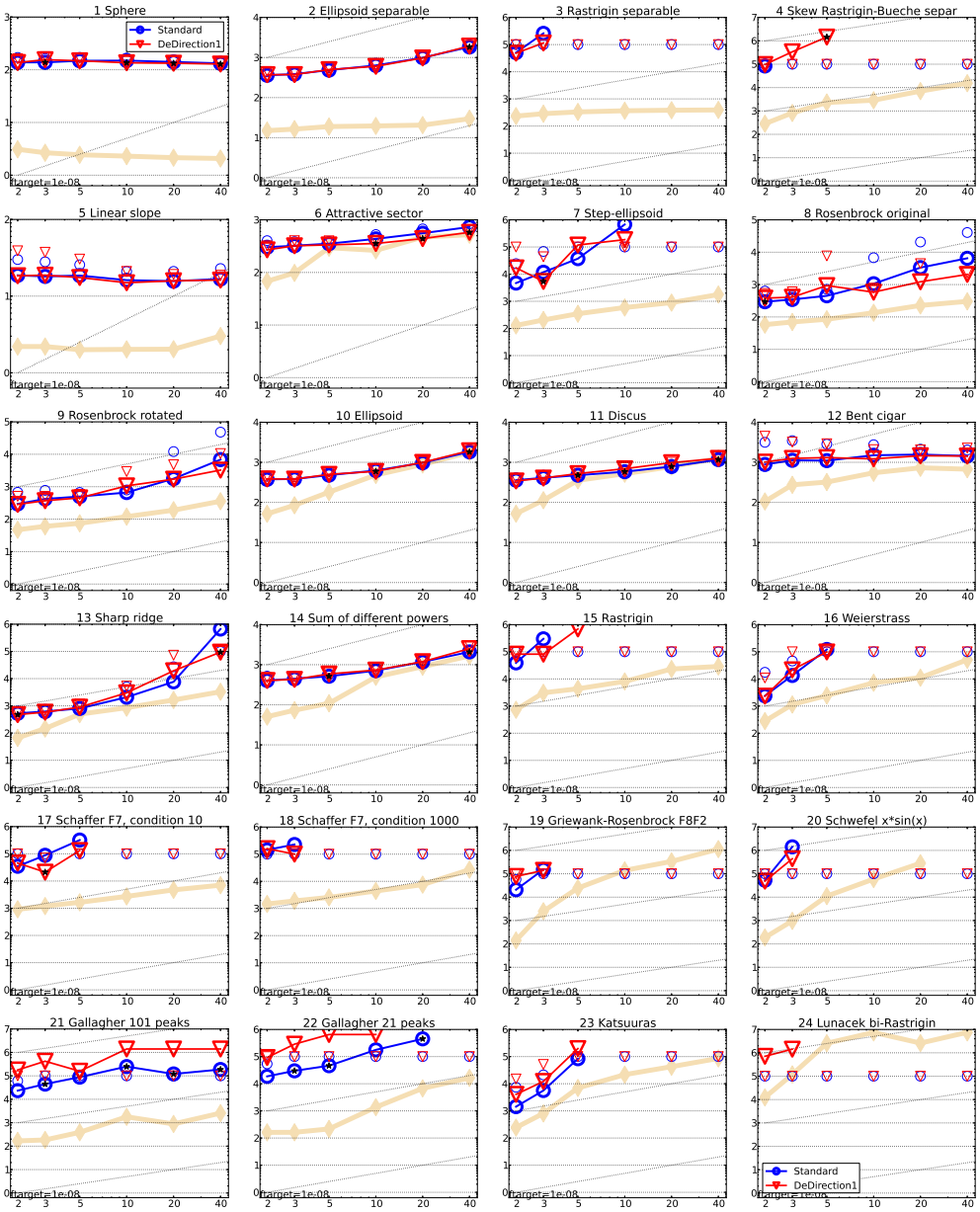

Figure 5.18: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function 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: ○:CMA-ES, ▽:Orthogonal-Sampling1

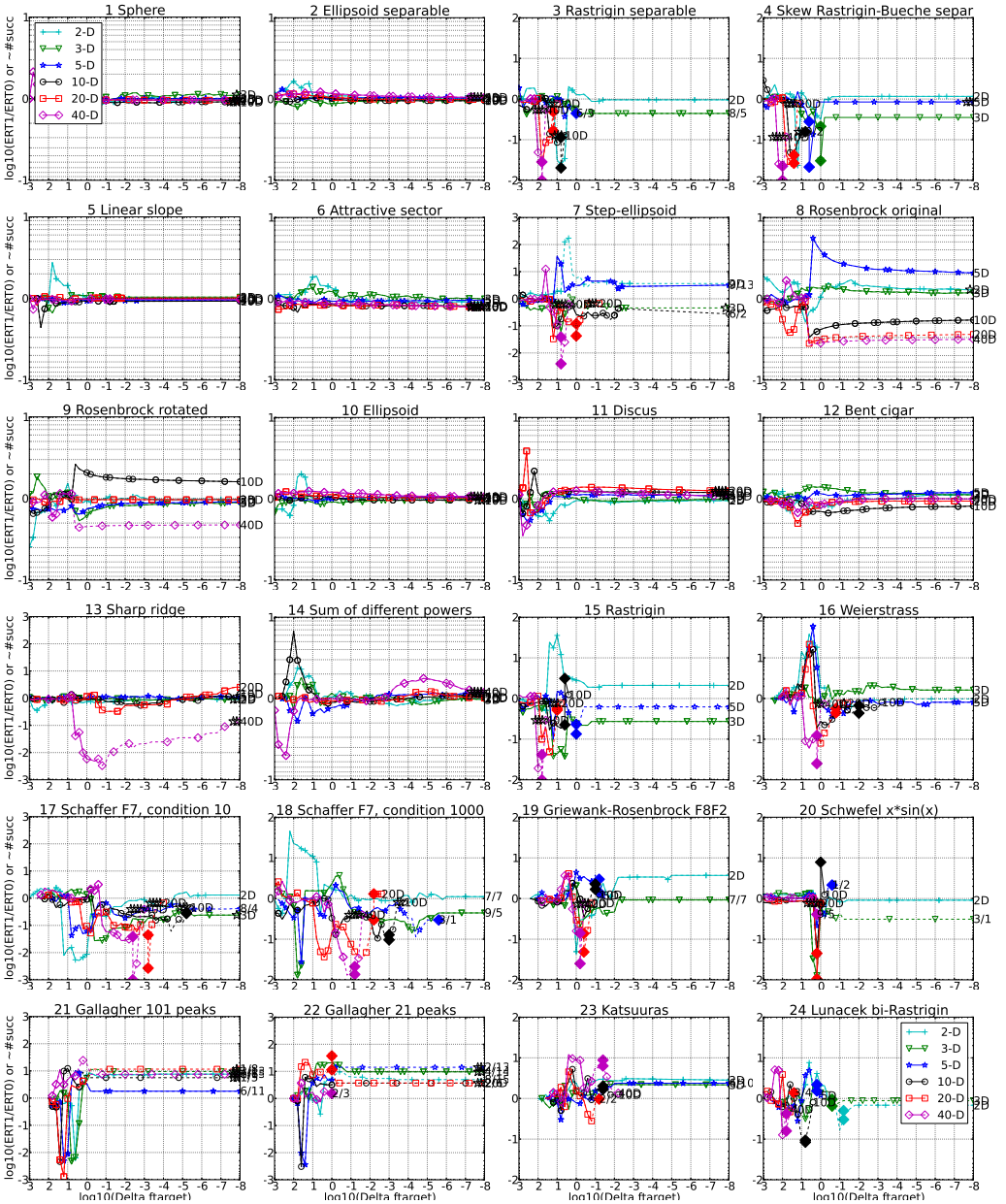

Figure 5.19: Ratio of ERT for Orthogonal-Sampling1 over ERT for CMA-ES versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇. Ratios $< 10^0$ indicate an advantage of Orthogonal-Sampling1, smaller values are always better. The line becomes dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall number of f-evaluations for the same algorithm on this function. Filled symbols indicate the best achieved Δf-value of one algorithm (ERT is undefined to the right). The dashed line continues as the fraction of successful trials of the other algorithm, where 0 means 0% and the y-axis limits mean 100%, values below zero for Orthogonal-Sampling1. The line ends when no algorithm reaches Δf anymore. The number of successful trials is given, only if it was in $\{1 \ldots 9\}$ for Orthogonal-Sampling1 (1^{st} number) and non-zero for CMA-ES (2^{nd} number). Results are significant with $p = 0.05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni correction within each figure.

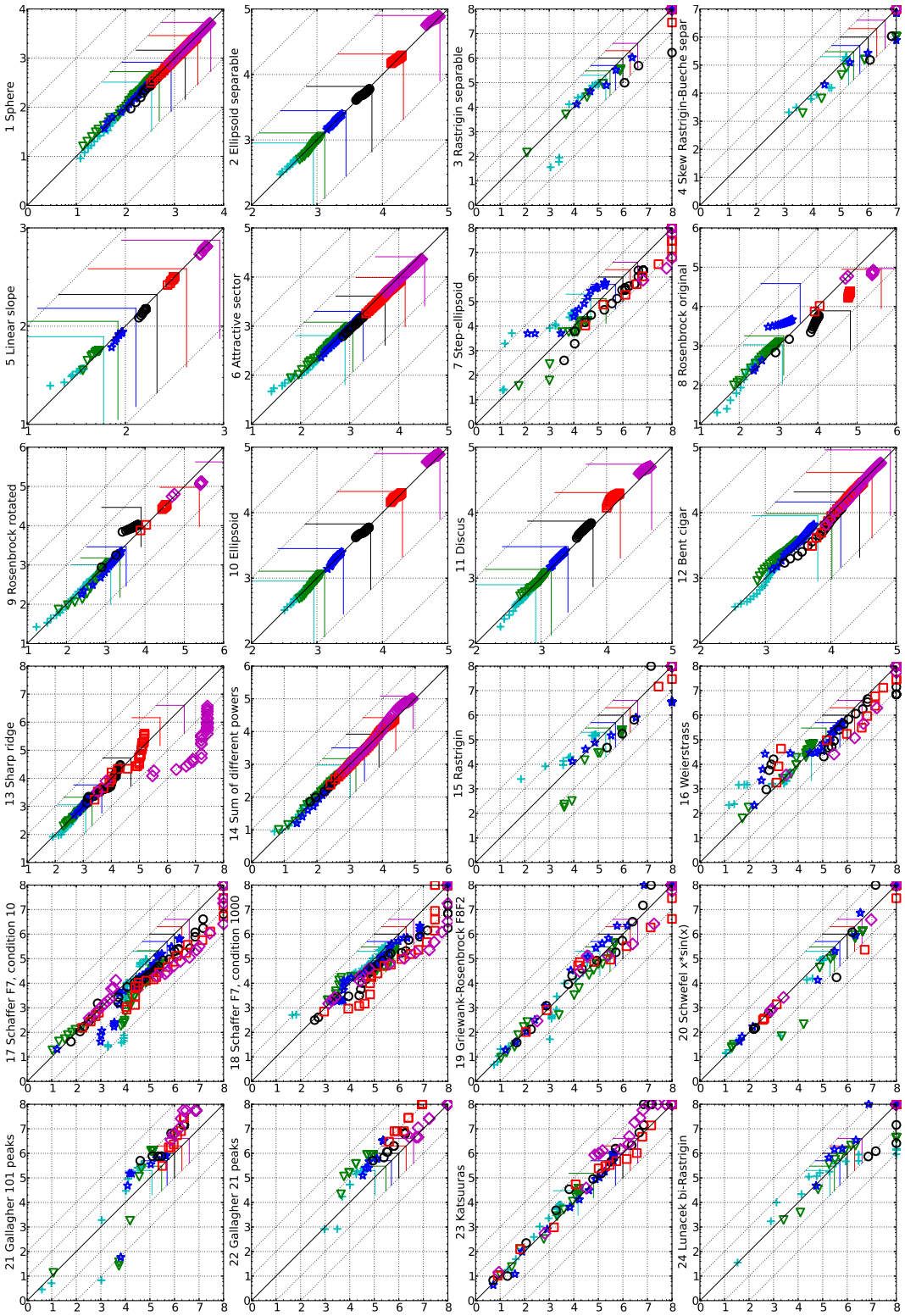

Figure 5.20: Expected running time (ERT in $\log_{10}$ of number of function evaluations) of CMA-ES (x-axis) versus Orthogonal-Sampling1 (y-axis) for 46 target values $\Delta f \in [10^{-8}, 10]$ in each dimension on functions f_1–f_{24}. Markers on the upper or right edge indicate that the target value was never reached. Markers represent dimension: 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40:◇.

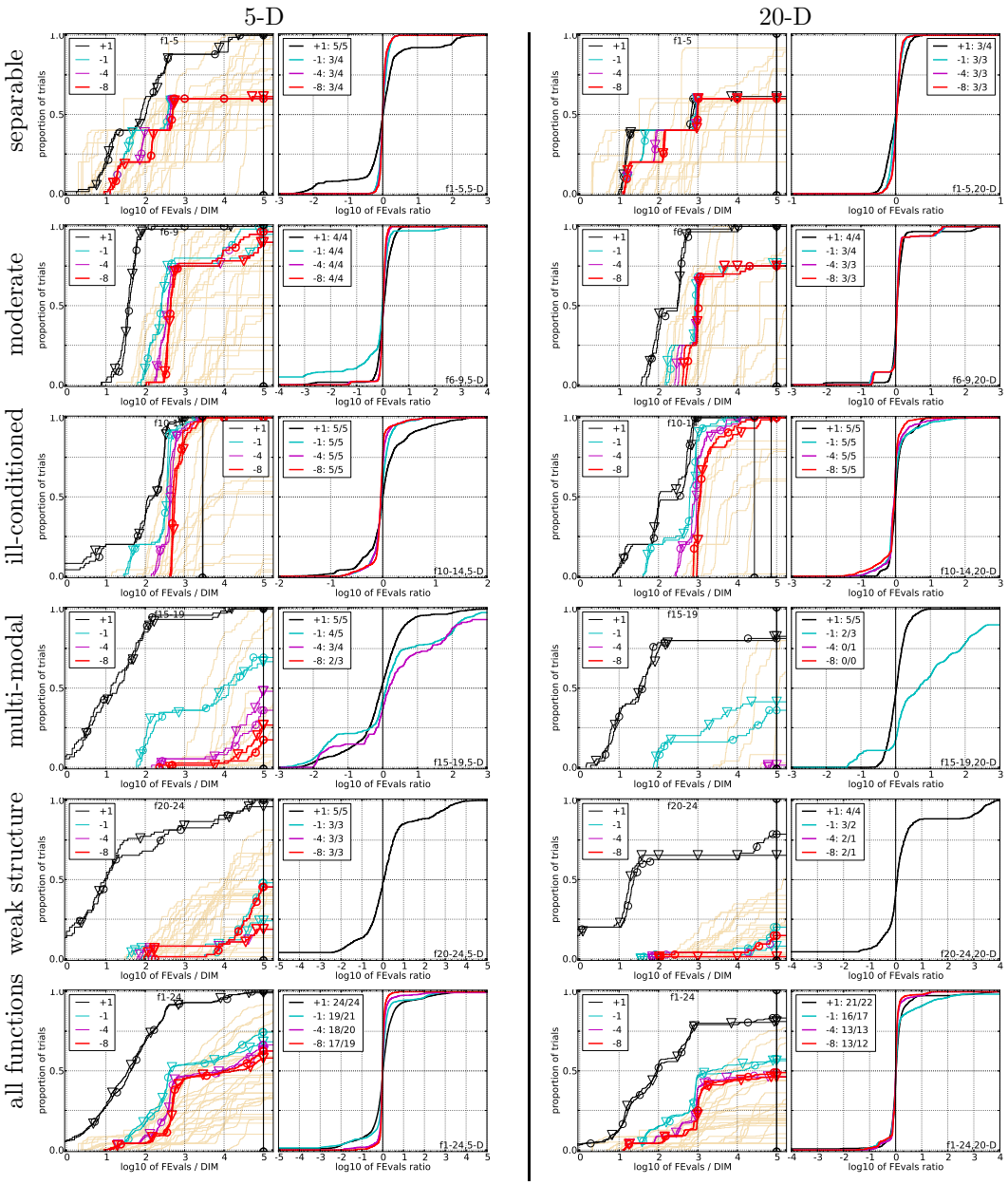

Figure 5.21: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for CMA-ES ($\circ$) and Orthogonal-Sampling1 ($\triangledown$). 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 CMA-ES divided by Orthogonal-Sampling1, 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 (CMA-ES 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: CMA	3.4(2)	15(5)	28(5)	41(4)	54(6)	15/15
2: OS1	3.4(2)	15(4)	28(5)	41(6)	54(6)	15/15
f_2	83	88	90	92	94	15/15
1: CMA	17(6)	21(3)	22(3)	24(3)	25(3)	15/15
2: OS1	17(5)	21(4)	23(2)	25(2)	26(2)	15/15
f_3	716	1637	1646	1650	1654	15/15
1: CMA	18(44)	∞	∞	∞	∞*5.0e5*	0/15
2: OS1	18(26)	∞	∞	∞	∞*5.0e5*	0/15
f_4	809	1688	1817	1886	1903	15/15
1: CMA	34(40)	∞	∞	∞	∞*5.0e5*	0/15
2: OS1	25(43)	4300(4887)	3994(4402)	3849(4176)	3814(3809)	1/15
f_5	10	10	10	10	10	15/15
1: CMA	7.2(3)	9.2(3)	9.3(3)	9.3(3)	9.3(3)	15/15
2: OS1	6.1(2)	8.7(3)	8.7(3)	8.7(3)	8.7(3)	15/15
f_6	114	281	580	1038	1332	15/15
1: CMA	2.3(0.9)	2.2(0.5)	1.6(0.2)	1.2(0.1)	1.2(0.1)	15/15
2: OS1	2.0(0.8)	2.0(0.4)	1.6(0.3)	1.1(0.2)	1.1(0.2)	15/15
f_7	24	1171	1572	1572	1597	15/15
1: CMA	5.6(3)	71(55)	118(162)	118(165)	116(163)	13/15
2: OS1	213(3)	318(352)	376(423)	376(432)	370(415)	9/15
f_8	73	336	391	410	422	15/15
1: CMA	3.2(1)	4.1(0.9)	4.6(1)	4.9(1.0)	5.1(1)	15/15
2: OS1	3.2(1)	11(1)	11(1)	11(1)	11(1)	15/15
f_9	35	214	300	335	369	15/15
1: CMA	7.1(5)	7.6(3)	6.8(2)	6.7(2)	6.5(2)	15/15
2: OS1	5.2(2)	6.3(2)	5.9(1)	5.9(1)	5.9(1)	15/15
f_{10}	349	574	626	829	880	15/15
1: CMA	4.2(0.9)	3.2(0.3)	3.2(0.3)	2.6(0.2)	2.6(0.2)	15/15
2: OS1	3.9(0.9)	3.2(0.6)	3.3(0.4)	2.7(0.3)	2.7(0.3)	15/15
f_{11}	143	763	1177	1467	1673	15/15
1: CMA	10(2)	2.3(0.3)	1.7(0.1)	1.5(0.1)	**1.4**(0.1)*	15/15
2: OS1	10(4)	2.6(0.4)	1.9(0.2)	1.6(0.1)	1.5(0.2)	15/15
f_{12}	108	371	461	1303	1494	15/15
1: CMA	12(15)	9.0(10)	8.8(9)	3.7(4)	3.5(4)	15/15
2: OS1	13(17)	10(9)	10(9)	4.3(3)	4.2(3)	15/15
f_{13}	132	250	1310	1752	2255	15/15
1: CMA	3.6(3)	5.8(1)	1.5(0.5)	1.8(0.3)	1.6(0.2)	15/15
2: OS1	3.7(2)	6.6(3)	1.7(0.5)	1.8(0.4)	1.9(0.3)	15/15
f_{14}	10	58	139	251	476	15/15
1: CMA	2.5(2)	3.8(0.8)	4.3(0.9)	5.8(0.8)	**4.7**(0.5)$^{\star 2}$	15/15
2: OS1	1.6(2)	3.6(0.9)	4.9(1)	6.5(0.7)	5.3(0.4)	15/15
f_{15}	511	19369	20073	20769	21359	14/15
1: CMA	17(58)	∞	∞	∞	∞*5.0e5*	0/15
2: OS1	26(40)	179(194)	173(192)	167(193)	162(187)	2/15
f_{16}	120	2662	10449	11644	12095	15/15
1: CMA	1.4(1)	39(48)	45(48)	52(54)	50(54)	9/15
2: OS1	1.8(1)	27(47)	33(23)	37(34)	41(41)	10/15
f_{17}	5.2	899	3669	6351	7934	15/15
1: CMA	3.2(3)	6.6(18)	13(16)	48(58)	202(239)	4/15
2: OS1	4.0(4)	4.4(9)	5.1(7)	25(30)	81(95)	8/15
f_{18}	103	3968	9280	10905	12469	15/15
1: CMA	18(1)	12(15)	43(54)	668(757)	∞*5.0e5*	0/15
2: OS1	17(1)	6.8(8)	29(21)	198(225)	∞*5.0e5*	0/15
f_{19}	1	242	1.2e5	1.2e5	1.2e5	15/15
1: CMA	17(14)	2405(3097)	∞	∞	∞*5.0e5*	0/15
2: OS1	19(9)	9045(11155)	∞	∞	∞*5.0e5*	0/15
f_{20}	16	38111	54470	54861	55313	14/15
1: CMA	2.5(2)	∞	∞	∞	∞*5.0e5*	0/15
2: OS1	2.6(2)	∞	∞	∞	∞*5.0e5*	0/15
f_{21}	41	1674	1705	1729	1757	14/15
1: CMA	159(433)	263(267)	259(248)	255(259)	251(255)	11/15
2: OS1	1.4(1)	472(659)	464(586)	458(578)	450(580)	6/15
f_{22}	71	938	1008	1040	1068	14/15
1: CMA	478(1096)	245(268)	228(253)	221(251)	216(235)	13/15
2: OS1	1761(3521)	3466(4266)	3224(3720)	3125(3845)	3045(3513)	2/15
f_{23}	3.0	14249	31654	33030	34256	15/15
1: CMA	1.8(1)	19(21)	13(15)	13(14)	12(15)	10/15
2: OS1	1.4(2)	30(36)	31(32)	30(32)	29(31)	6/15
f_{24}	1622	6.4e6	9.6e6	1.3e7	1.3e7	3/15
1: CMA	33(76)	∞	∞	∞	∞*5.0e5*	0/15
2: OS1	30(46)	∞	∞	∞	∞*5.0e5*	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: CMA	7.8(1)	20(2)	33(3)	46(3)	58(3)	15/15
2: OS1	7.3(1)	19(2)	31(2)	44(2)	56(2)	15/15
f_2	385	387	390	391	393	15/15
1: CMA	35(3)	44(4)	47(2)	48(2)	50(2)	15/15
2: OS1	38(7)	45(3)	47(2)	48(2)	49(2)	15/15
f_3	5066	7635	7643	7646	7651	15/15
1: CMA	∞	∞	∞	∞	∞*2.0e6*	0/15
2: OS1	5555(6316)	∞	∞	∞	∞*2.0e6*	0/15
f_4	4722	7666	7700	7758	1.4e5	9/15
1: CMA	∞	∞	∞	∞	∞*2.0e6*	0/15
2: OS1	∞	∞	∞	∞	∞*2.0e6*	0/15
f_5	41	41	41	41	41	15/15
1: CMA	6.6(2)	7.7(1)	7.7(1)	7.7(1)	7.7(1)	15/15
2: OS1	6.5(1.0)	7.8(1.0)	7.8(1.0)	7.8(1.0)	7.8(1.0)	15/15
f_6	1296	3413	5220	6728	8409	15/15
1: CMA	1.7(0.5)	1.2(0.2)	1.1(0.1)	1.2(0.1)	1.2(0.1)	15/15
2: OS1	1.4(0.2)	**1.00**(0.1)$^{\star 2}$	**0.96**(0.1)$^{\star 2}$	**0.97**(0.1)$^{\star 3}$	**0.96**(0.1)$^{\star 3}$	15/15
f_7	1351	9503	16524	16524	16969	15/15
1: CMA	22(78)	∞	∞	∞	∞*2.0e6*	0/15
2: OS1	7.9(0.5)	**3117**(3420)$^{\star 2}$	∞	∞	∞*2.0e6*	0/15
f_8	2039	4040	4219	4371	4484	15/15
1: CMA	4.1(1.0)	16(38)	15(36)	15(35)	15(34)	15/15
2: OS1	3.7(0.9)	5.3(0.5)	5.4(0.4)	5.4(0.4)	5.4(0.4)	15/15
f_9	1716	3277	3455	3594	3727	15/15
1: CMA	4.5(1)	10(0.6)	10(0.5)	10(0.5)	9.4(0.5)	15/15
2: OS1	4.4(1)	9.5(11)	9.4(11)	9.2(10)	9.1(10)	15/15
f_{10}	7413	10735	14920	17073	17476	15/15
1: CMA	1.8(0.3)	1.6(0.1)	1.2(0.0)	1.1(0.0)	1.1(0.0)	15/15
2: OS1	1.9(0.3)	1.7(0.1)	1.2(0.0)	1.1(0.0)	1.1(0.0)	15/15
f_{11}	1002	6278	9762	12285	14831	15/15
1: CMA	10(0.7)	**1.9**(0.1)$^{\star 3}$	**1.4**(0.0)$^{\star 3}$	**1.2**(0.0)$^{\star 3}$	**1.0**(0.0)$^{\star 3}$	15/15
2: OS1	12(4)	2.7(0.2)	1.9(0.1)	1.5(0.0)	1.3(0.0)	15/15
f_{12}	1042	2740	4140	12407	13827	15/15
1: CMA	5.0(4)	5.6(3)	5.0(2)	2.1(0.7)	2.2(0.7)	15/15
2: OS1	3.0(2)	5.1(2)	4.7(2)	2.0(0.6)	2.0(0.6)	15/15
f_{13}	652	2751	18749	24455	30201	15/15
1: CMA	3.9(5)	30(56)	5.3(8)	5.0(6)	4.7(5)	15/15
2: OS1	2.7(0.5)	10(11)	3.0(2)	4.6(3)	8.3(8)	15/15
f_{14}	75	304	932	1648	15661	15/15
1: CMA	3.5(2)	3.6(0.6)	4.1(0.5)	**6.0**(0.6)*	1.2(0.1)	15/15
2: OS1	3.2(0.9)	3.3(0.4)	3.8(0.4)	7.3(1)	1.2(0.2)	15/15
f_{15}	30378	3.1e5	3.2e5	4.5e5	4.6e5	15/15
1: CMA	934(1103)	∞	∞	∞	∞*2.0e6*	0/15
2: OS1	483(494)	∞	∞	∞	∞*2.0e6*	0/15
f_{16}	1384	77015	1.9e5	2.0e5	2.2e5	15/15
1: CMA	0.96(0.2)	∞	∞	∞	∞*2.0e6*	0/15
2: OS1	1.3(0.7)	367(416)	∞	∞	∞*2.0e6*	0/15
f_{17}	63	4005	30677	56288	80472	15/15
1: CMA	2.8(1)	8.4(19)	447(488)	∞	∞*2.0e6*	0/15
2: OS1	2.4(2)	2.6(5)	**19**(22)$^{\star 3}$	∞	∞*2.0e6*	0/15
f_{18}	621	19561	67569	1.3e5	1.5e5	15/15
1: CMA	1.5(0.6)	65(64)	∞	∞	∞*2.0e6*	0/15
2: OS1	1.1(0.5)	**4.9**(7)$^{\star 3}$	∞	∞	∞*2.0e6*	0/15
f_{19}	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
1: CMA	112(50)	∞	∞	∞	∞*2.0e6*	0/15
2: OS1	103(40)	∞	∞	∞	∞*2.0e6*	0/15
f_{20}	82	3.1e6	5.5e6	5.6e6	5.6e6	14/15
1: CMA	4.5(1)	∞	∞	∞	∞*2.0e6*	0/15
2: OS1	4.0(1)	∞	∞	∞	∞*2.0e6*	0/15
f_{21}	561	14103	14643	15567	17589	15/15
1: CMA	568(1107)	171(213)	165(195)	155(199)	137(181)	8/15
2: OS1	549(1781)	1985(2340)	1912(2254)	1799(1991)	1592(1848)	1/15
f_{22}	467	23491	24948	26847	1.3e5	12/15
1: CMA	862(1067)	**384**(397)*	**361**(401)*	**336**(376)*	**67**(76)*	3/15
2: OS1	6428(8569)	∞	∞	∞	∞*2.0e6*	0/15
f_{23}	3.2	67457	4.9e5	8.1e5	8.4e5	15/15
1: CMA	2.5(2)	92(104)	∞	∞	∞*2.0e6*	0/15
2: OS1	3.2(3)	70(79)	∞	∞	∞*2.0e6*	0/15
f_{24}	1.3e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
1: CMA	∞	∞	∞	∞	∞*2.0e6*	0/15
2: OS1	∞	∞	∞	∞	∞*2.0e6*	0/15

Table 5.11: 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 different Δf values. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. 1:CMA is CMA-ES and 2:OS1 is Orthogonal-Sampling1. 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.

5.4.7 Orthogonal$2-(\mu/\mu_{\mathbf{w}}, \lambda_{\mathbf{m}})$-CMA-ES vs $(\mu/\mu_{\mathbf{w}}, \lambda_{\mathbf{m}})$-CMA-ES

Results from experiments according to [22] on the benchmark functions given in [19, 23] are presented in Figures 5.22, 5.23, 5.24, 5.25 and Table 5.12. The **expected running time (**ERT**)**, used in the figures and table, depends on a given target function value, $f_{\mathrm{t}} = f_{\mathrm{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_{t}, summed over all trials and divided by the number of trials that actually reached f_{t} [22, 33]. **Statistical significance** is tested with the rank-sum test for a given target Δf_{t} using, for each trial, either the number of needed function evaluations to reach Δf_{t} (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration if available.

Figure 5.22 plots the expected function evaluation against dimension for both the competitive ESs. The "noisy" mirrored sampling method actually is getting worse on function $f_{13}, f_{21}, f_{22}, f_{23}$ under the target precision 10^{-8}.

In Figure 5.23, the observation above is presented in details. The logarithm of the ratio between two competitive ESs are plotted against decreasing target precision. Most of the curves in each subfigures are quite noisy. If most of the curves in one subfigure are roughly below 0, then we could consider that the newly purposed algorithm is performing better than the original algorithm in that subfigure. From figure 5.23, the "noisy" sampling method gets worse on functions f_{21}, f_{22}, f_{23}. There is no decision for the rest functions. Figure 5.24 changes to another manner to compare the ERTs and basically tells the same story as Figure 5.23.

Then the most important characteristic is shown in Figure 5.25. The distribution curves for the "noisy" mirrored sampling method are roughly the same as that of mirrored sampling method on most functions, indicating the same performance characteristics of two competitive ESs. Finally, the specific ERT numbers under precision 10^{-8} are compared in Table 5.12.

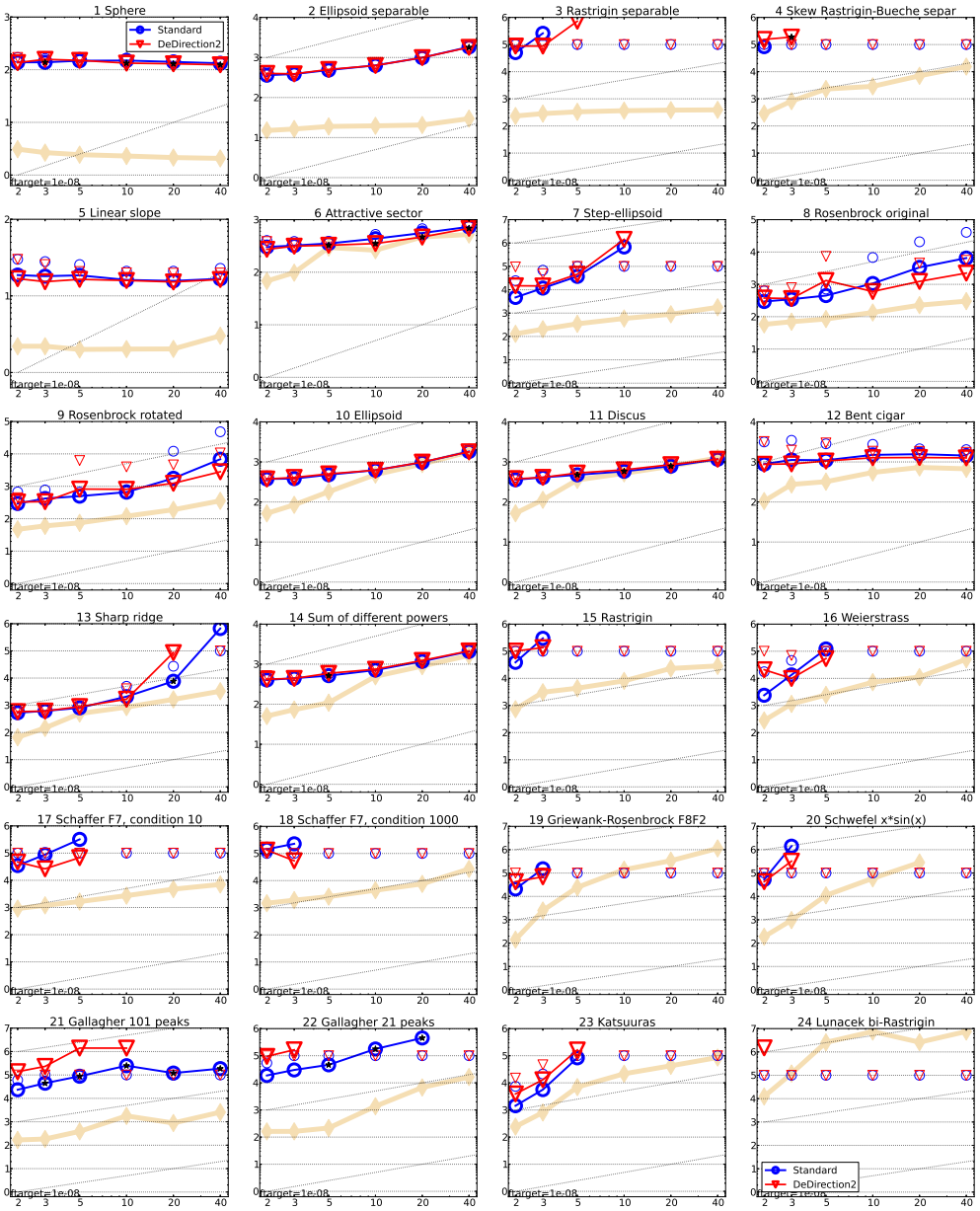

Figure 5.22: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function 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: ○:CMA-ES, ▽:Orthogonal-Sampling2

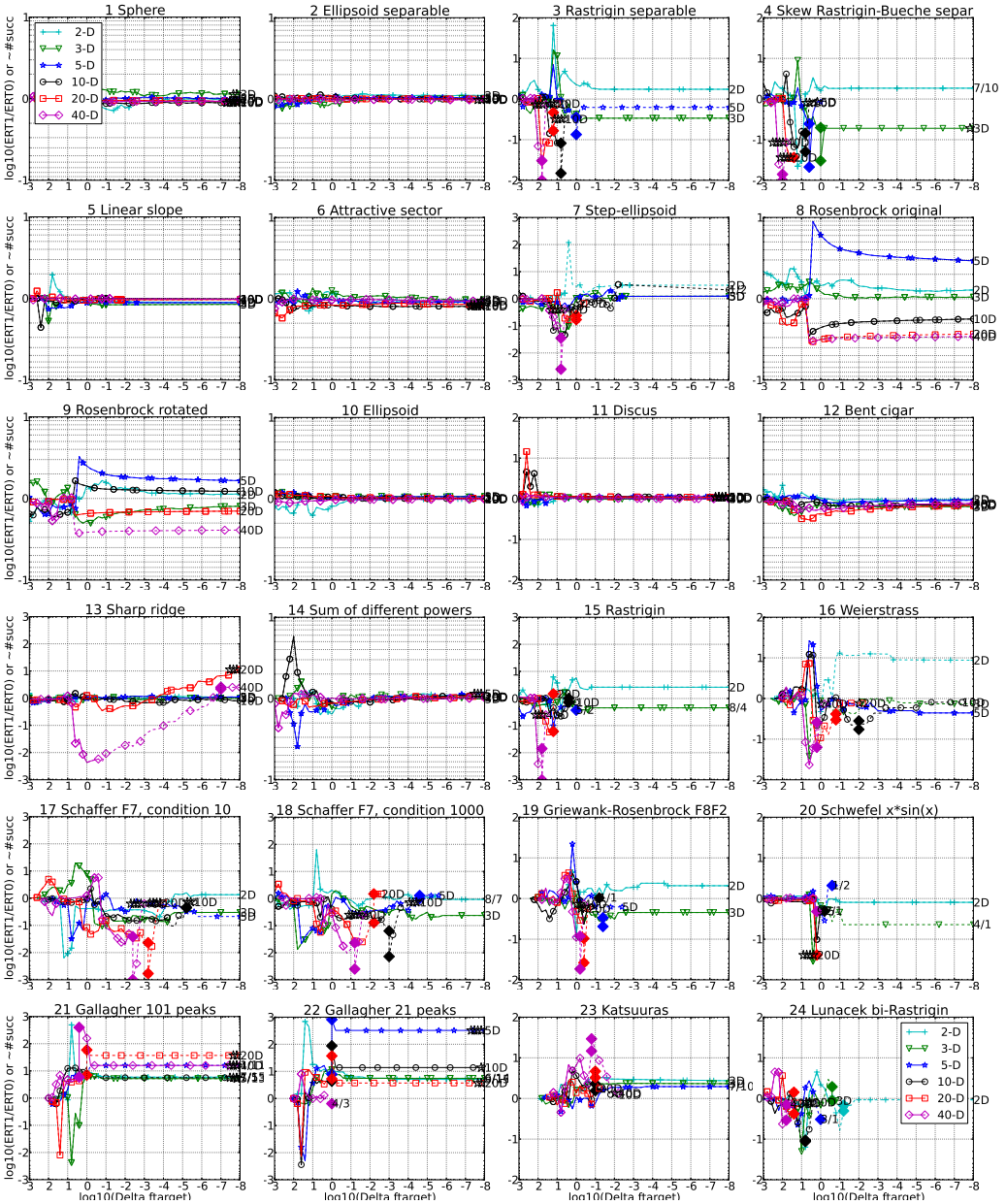

Figure 5.23: Ratio of ERT for Orthogonal-Sampling2 over ERT for CMA-ES versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇. Ratios $< 10^0$ indicate an advantage of Orthogonal-Sampling2, smaller values are always better. The line becomes dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall number of f-evaluations for the same algorithm on this function. Filled symbols indicate the best achieved Δf-value of one algorithm (ERT is undefined to the right). The dashed line continues as the fraction of successful trials of the other algorithm, where 0 means 0% and the y-axis limits mean 100%, values below zero for Orthogonal-Sampling2. The line ends when no algorithm reaches Δf anymore. The number of successful trials is given, only if it was in $\{1 \ldots 9\}$ for Orthogonal-Sampling2 (1^{st} number) and non-zero for CMA-ES (2^{nd} number). Results are significant with $p = 0.05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni correction within each figure.

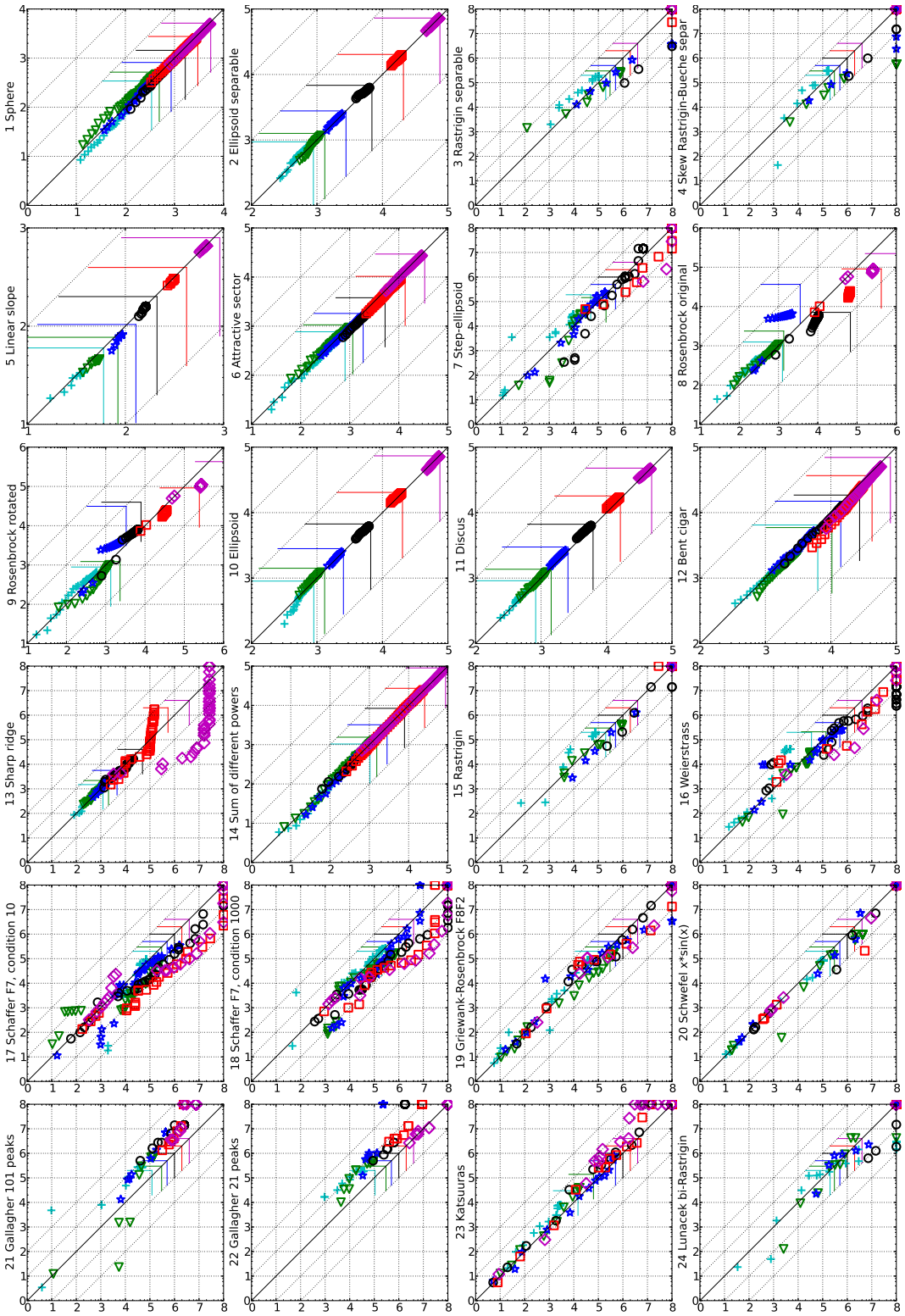

Figure 5.24: Expected running time (ERT in $\log_{10}$ of number of function evaluations) of CMA-ES (x-axis) versus Orthogonal-Sampling2 (y-axis) for 46 target values $\Delta f \in [10^{-8}, 10]$ in each dimension on functions f_1–f_{24}. Markers on the upper or right edge indicate that the target value was never reached. Markers represent dimension: 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40:◇.

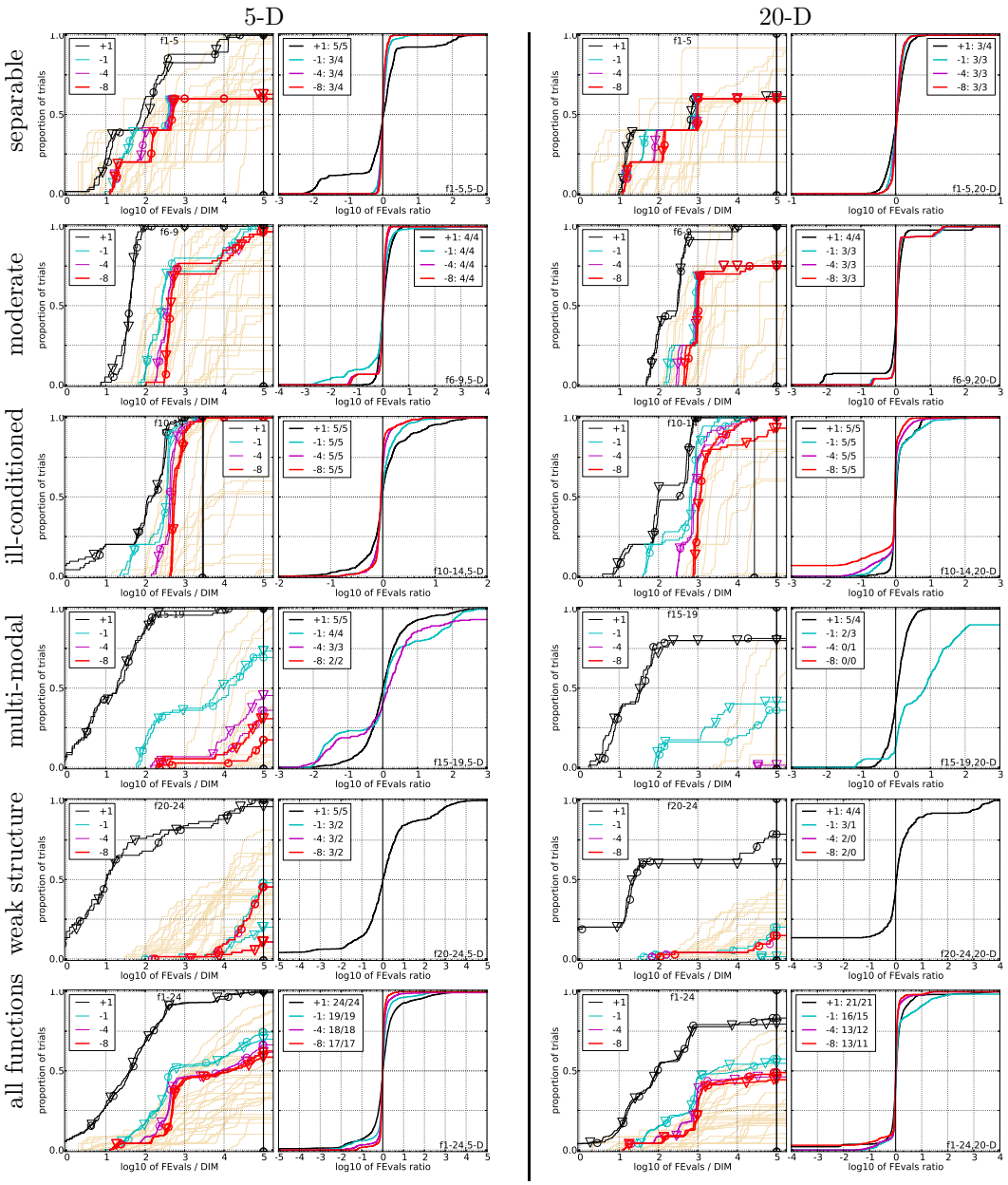

Figure 5.25: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for CMA-ES ($\circ$) and Orthogonal-Sampling2 ($\triangledown$). 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 CMA-ES divided by Orthogonal-Sampling2, 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 (CMA-ES 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: CMA	3.4(2)	15(5)	28(5)	41(4)	54(6)	15/15
2: OS2	3.1(2)	15(5)	29(5)	42(4)	54(4)	15/15
f_2	83	88	90	92	94	15/15
1: CMA	17(6)	21(3)	22(3)	24(3)	25(3)	15/15
2: OS2	17(7)	22(3)	23(3)	25(3)	26(2)	15/15
f_3	716	1637	1646	1650	1654	15/15
1: CMA	18(44)	∞	∞	∞	∞*5.0e5*	0/15
2: OS2	18(26)	2211(2281)	2200(2430)	2194(2272)	2189(2414)	2/15
f_4	809	1688	1817	1886	1903	15/15
1: CMA	34(40)	∞	∞	∞	∞*5.0e5*	0/15
2: OS2	23(21)	∞	∞	∞	∞*5.0e5*	0/15
f_5	10	10	10	10	10	15/15
1: CMA	7.2(3)	9.2(3)	9.3(3)	9.3(3)	9.3(3)	15/15
2: OS2	5.6(1)	8.2(1)	8.2(1)	8.2(1)	8.2(1)	15/15
f_6	114	281	580	1038	1332	15/15
1: CMA	2.3(0.9)	2.2(0.5)	1.6(0.2)	1.2(0.1)	1.2(0.1)	15/15
2: OS2	2.2(0.7)	2.0(0.3)	1.5(0.2)	1.1(0.2)	1.1(0.1)	15/15
f_7	24	1171	1572	1572	1597	15/15
1: CMA	5.6(3)	71(55)	118(166)	118(166)	116(163)	13/15
2: OS2	4.2(3)	84(71)	147(168)	147(168)	145(165)	13/15
f_8	73	336	391	410	422	15/15
1: CMA	3.2(1)	4.1(0.9)	4.6(1)	4.9(1.0)	5.1(1)	15/15
2: OS2	3.2(1)	17(45)	16(39)	15(37)	15(36)	15/15
f_9	35	214	300	335	369	15/15
1: CMA	7.1(5)	7.6(3)	6.8(2)	6.7(2)	6.5(2)	15/15
2: OS2	5.7(2)	15(2)	12(1)	11(1.0)	11(0.9)	15/15
f_{10}	349	574	626	829	880	15/15
1: CMA	4.2(0.9)	3.2(0.3)	3.2(0.3)	2.6(0.2)	2.6(0.2)	15/15
2: OS2	4.4(1)	3.3(0.5)	3.3(0.4)	2.7(0.3)	2.8(0.3)	15/15
f_{11}	143	763	1177	1467	1673	15/15
1: CMA	10(2)	2.3(0.3)	**1.7**(0.1)*	**1.5**(0.1)$^{\star 2}$	**1.4**(0.1)$^{\star 2}$	15/15
2: OS2	11(3)	2.6(0.3)	1.9(0.2)	1.6(0.2)	1.5(0.1)	15/15
f_{12}	108	371	461	1303	1494	15/15
1: CMA	12(15)	9.0(10)	8.8(9)	3.7(4)	3.5(4)	15/15
2: OS2	12(15)	7.8(10)	8.2(9)	3.5(3)	3.4(3)	15/15
f_{13}	132	250	1310	1752	2255	15/15
1: CMA	3.6(3)	5.8(1)	1.5(0.5)	1.8(0.3)	1.6(0.2)	15/15
2: OS2	3.6(2)	6.0(3)	1.8(0.4)	1.8(0.4)	1.8(0.4)	15/15
f_{14}	10	58	139	251	476	15/15
1: CMA	2.5(2)	3.8(0.8)	4.3(0.9)	5.8(0.8)	**4.7**(0.5)*	15/15
2: OS2	1.7(2)	3.4(1)	4.5(1.0)	6.0(0.9)	5.3(0.5)	15/15
f_{15}	511	19369	20073	20769	21359	14/15
1: CMA	17(58)	∞	∞	∞	∞*5.0e5*	0/15
2: OS2	5.5(0.8)	∞	∞	∞	∞*5.0e5*	0/15
f_{16}	120	2662	10449	11644	12095	15/15
1: CMA	1.4(1)	39(48)	45(45)	52(54)	50(52)	9/15
2: OS2	1.2(0.9)	38(47)	23(25)	23(30)	22(30)	12/15
f_{17}	5.2	899	3669	6351	7934	15/15
1: CMA	3.2(3)	6.6(18)	13(16)	48(58)	202(221)	4/15
2: OS2	2.2(1)	9.2(18)	9.3(12)	20(18)	43(44)	11/15
f_{18}	103	3968	9280	10905	12469	15/15
1: CMA	18(1)	12(15)	43(60)	668(757)	∞*5.0e5*	0/15
2: OS2	1.5(0.8)	4.3(6)	24(23)	∞	∞*5.0e5*	0/15
f_{19}	1	242	1.2e5	1.2e5	1.2e5	15/15
1: CMA	17(14)	2405(2507)	∞	∞	∞*5.0e5*	0/15
2: OS2	20(14)	1512(1607)	∞	∞	∞*5.0e5*	0/15
f_{20}	16	38111	54470	54861	55313	14/15
1: CMA	2.5(2)	∞	∞	∞	∞*5.0e5*	0/15
2: OS2	2.6(2)	∞	∞	∞	∞*5.0e5*	0/15
f_{21}	41	1674	1705	1729	1757	14/15
1: CMA	159(433)	**263**(253)*	**259**(263)*	**255**(259)*	**251**(256)*	11/15
2: OS2	325(833)	4203(4778)	4127(4692)	4071(4483)	4006(4411)	1/15
f_{22}	71	938	1008	1040	1068	14/15
1: CMA	478(1096)	**245**(297)$^{\star 3}$	**228**(259)$^{\star 3}$	**221**(245)$^{\star 3}$	**216**(245)$^{\star 3}$	13/15
2: OS2	1762(3521)	∞	∞	∞	∞*5.0e5*	0/15
f_{23}	3.0	14249	31654	33030	34256	15/15
1: CMA	1.8(1)	19(21)	13(17)	13(12)	12(14)	10/15
2: OS2	2.1(2)	15(15)	25(28)	25(29)	24(27)	7/15
f_{24}	1622	6.4e6	9.6e6	1.3e7	1.3e7	3/15
1: CMA	33(76)	∞	∞	∞	∞*5.0e5*	0/15
2: OS2	14(20)	∞	∞	∞	∞*5.0e5*	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: CMA	7.8(1)	20(2)	33(3)	46(3)	58(3)	15/15
2: OS2	7.5(2)	19(2)	31(2)	43(3)	**54**(3)*	15/15
f_2	385	387	390	391	393	15/15
1: CMA	35(3)	44(4)	47(2)	48(2)	50(2)	15/15
2: OS2	35(5)	44(5)	47(2)	49(1)	50(1)	15/15
f_3	5066	7635	7643	7646	7651	15/15
1: CMA	∞	∞	∞	∞	∞*2.0e6*	0/15
2: OS2	5741(6711)	∞	∞	∞	∞*2.0e6*	0/15
f_4	4722	7666	7700	7758	1.4e5	9/15
1: CMA	∞	∞	∞	∞	∞*2.0e6*	0/15
2: OS2	∞	∞	∞	∞	∞*2.0e6*	0/15
f_5	41	41	41	41	41	15/15
1: CMA	6.6(2)	7.7(1)	7.7(1)	7.7(1)	7.7(1)	15/15
2: OS2	6.4(0.6)	7.4(0.9)	7.5(1)	7.5(1)	7.5(1)	15/15
f_6	1296	3413	5220	6728	8409	15/15
1: CMA	1.7(0.5)	1.2(0.2)	1.1(0.1)	1.2(0.1)	1.2(0.1)	15/15
2: OS2	1.4(0.2)	**1.0**(0.1)*	**1.00**(0.1)*	**1.0**(0.1)$^{\star 2}$	**1.0**(0.1)$^{\star 3}$	15/15
f_7	1351	9503	16524	16524	16969	15/15
1: CMA	22(78)	∞	∞	∞	∞*2.0e6*	0/15
2: OS2	37(54)	∞	∞	∞	∞*2.0e6*	0/15
f_8	2039	4040	4219	4371	4484	15/15
1: CMA	4.1(1.0)	16(38)	15(36)	15(35)	15(34)	15/15
2: OS2	3.5(0.9)	5.3(0.8)	5.4(0.8)	5.4(0.8)	5.4(0.8)	15/15
f_9	1716	3277	3455	3594	3727	15/15
1: CMA	4.5(1)	10(0.6)	10(0.5)	10(0.5)	9.4(0.5)	15/15
2: OS2	4.3(1.0)	6.6(0.6)	6.6(0.5)	6.6(0.5)	6.6(0.5)	15/15
f_{10}	7413	10735	14920	17073	17476	15/15
1: CMA	1.8(0.3)	1.6(0.1)	1.2(0.0)	1.1(0.0)	1.1(0.0)	15/15
2: OS2	1.9(0.2)	1.6(0.1)	1.2(0.0)	1.1(0.0)	1.1(0.0)	15/15
f_{11}	1002	6278	9762	12285	14831	15/15
1: CMA	**10**(0.7)$^{\star 3}$	**1.9**(0.1)$^{\star 3}$	**1.4**(0.0)$^{\star 3}$	**1.2**(0.0)$^{\star 3}$	**1.0**(0.0)$^{\star 3}$	15/15
2: OS2	11(0.6)	2.2(0.1)	1.5(0.0)	1.3(0.0)	1.1(0.0)	15/15
f_{12}	1042	2740	4140	12407	13827	15/15
1: CMA	5.0(4)	5.6(3)	5.0(2)	2.1(0.7)	2.2(0.7)	15/15
2: OS2	2.8(2)	3.9(3)	3.8(2)	1.7(0.7)	1.8(0.7)	15/15
f_{13}	652	2751	18749	24455	30201	15/15
1: CMA	3.9(5)	30(56)	5.3(8)	5.0(6)	**4.7**(5)*	15/15
2: OS2	**2.3**(0.1)*	13(11)	4.1(4)	12(11)	31(36)	10/15
f_{14}	75	304	932	1648	15661	15/15
1: CMA	3.5(2)	3.6(0.6)	4.1(0.5)	6.0(0.6)	1.2(0.1)	15/15
2: OS2	2.8(1)	3.3(0.4)	4.1(0.5)	6.3(0.4)	1.3(0.1)	15/15
f_{15}	30378	3.1e5	3.2e5	4.5e5	4.6e5	15/15
1: CMA	934(1053)	∞	∞	∞	∞*2.0e6*	0/15
2: OS2	∞	∞	∞	∞	∞*2.0e6*	0/15
f_{16}	1384	77015	1.9e5	2.0e5	2.2e5	15/15
1: CMA	0.96(0.2)	∞	∞	∞	∞*2.0e6*	0/15
2: OS2	1.4(1)	377(390)	∞	∞	∞*2.0e6*	0/15
f_{17}	63	4005	30677	56288	80472	15/15
1: CMA	2.8(1)	8.4(19)	447(488)	∞	∞*2.0e6*	0/15
2: OS2	2.2(1)	1.3(3)	**10**(8)$^{\star 3}$	∞	∞*2.0e6*	0/15
f_{18}	621	19561	67569	1.3e5	1.5e5	15/15
1: CMA	1.5(0.6)	65(66)	∞	∞	∞*2.0e6*	0/15
2: OS2	1.2(0.2)	**3.4**(3)$^{\star 3}$	∞	∞	∞*2.0e6*	0/15
f_{19}	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
1: CMA	112(50)	∞	∞	∞	∞*2.0e6*	0/15
2: OS2	90(46)	∞	∞	∞	∞*2.0e6*	0/15
f_{20}	82	3.1e6	5.5e6	5.6e6	5.6e6	14/15
1: CMA	4.5(1)	∞	∞	∞	∞*2.0e6*	0/15
2: OS2	4.2(0.8)	∞	∞	∞	∞*2.0e6*	0/15
f_{21}	561	14103	14643	15567	17589	15/15
1: CMA	568(1107)	**171**(213)*	**165**(205)*	**155**(199)*	**137**(171)*	8/15
2: OS2	2376(3563)	∞	∞	∞	∞*2.0e6*	0/15
f_{22}	467	23491	24948	26847	1.3e5	12/15
1: CMA	862(1067)	384(397)	361(387)	336(360)	67(69)	3/15
2: OS2	6428(9640)	∞	∞	∞	∞*2.0e6*	0/15
f_{23}	3.2	67457	4.9e5	8.1e5	8.4e5	15/15
1: CMA	2.5(2)	92(103)	∞	∞	∞*2.0e6*	0/15
2: OS2	1.7(2)	427(504)	∞	∞	∞*2.0e6*	0/15
f_{24}	1.3e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
1: CMA	∞	∞	∞	∞	∞*2.0e6*	0/15
2: OS2	∞	∞	∞	∞	∞*2.0e6*	0/15

Table 5.12: 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 different Δf values. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. 1:CMA is CMA-ES and 2:OS2 is Orthogonal-Sampling2. 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.

5.4.8 BBOB result of derandomized step-size

Results from experiments according to [22] on the benchmark functions given in [19, 23] are presented in Figures 5.26, 5.27, 5.28, 5.29 and Table 5.13. The **expected running time** (ERT), used in the figures and table, depends on a given target function value, $f_t = f_{opt} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_t, summed over all trials and divided by the number of trials that actually reached f_t [22, 33]. **Statistical significance** is tested with the rank-sum test for a given target Δf_t using, for each trial, either the number of needed function evaluations to reach Δf_t (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration if available.

Figure 5.26 plots the expected function evaluation against dimension for both the competitive ESs. The derandomized step-size sampling method is getting better on function $f_6, f_{11}, f_7, f_8, f_9, f_{13}, f_{15}$ under the target precision 10^{-8}. The amount of the improvement is not big. However, it is not the whole story.

In Figure 5.27, the observation above is presented in details. The derandomized step-size method almost dominates all the functions except $f_{19}, f_{21}, f_{22}, f_{23}, f_{24}$. Figure 5.28 changes to another manner to compare the ERTs and basically tells the same story as Figure 5.27.

Then the most important characteristic is shown in Figure 5.29. The distribution curves for the derandomized step-size sampling method are increasing much fast than that of standard CMSA-ES for 5 dimension, target precision 10^{-8}. On 20 dimension, the difference is not very big for all the functions and the distribution function of derandomized sampling only increases faster for precision 10^{-1}. Finally, the specific ERT numbers under precision 10^{-8} are compared in Table 5.13.

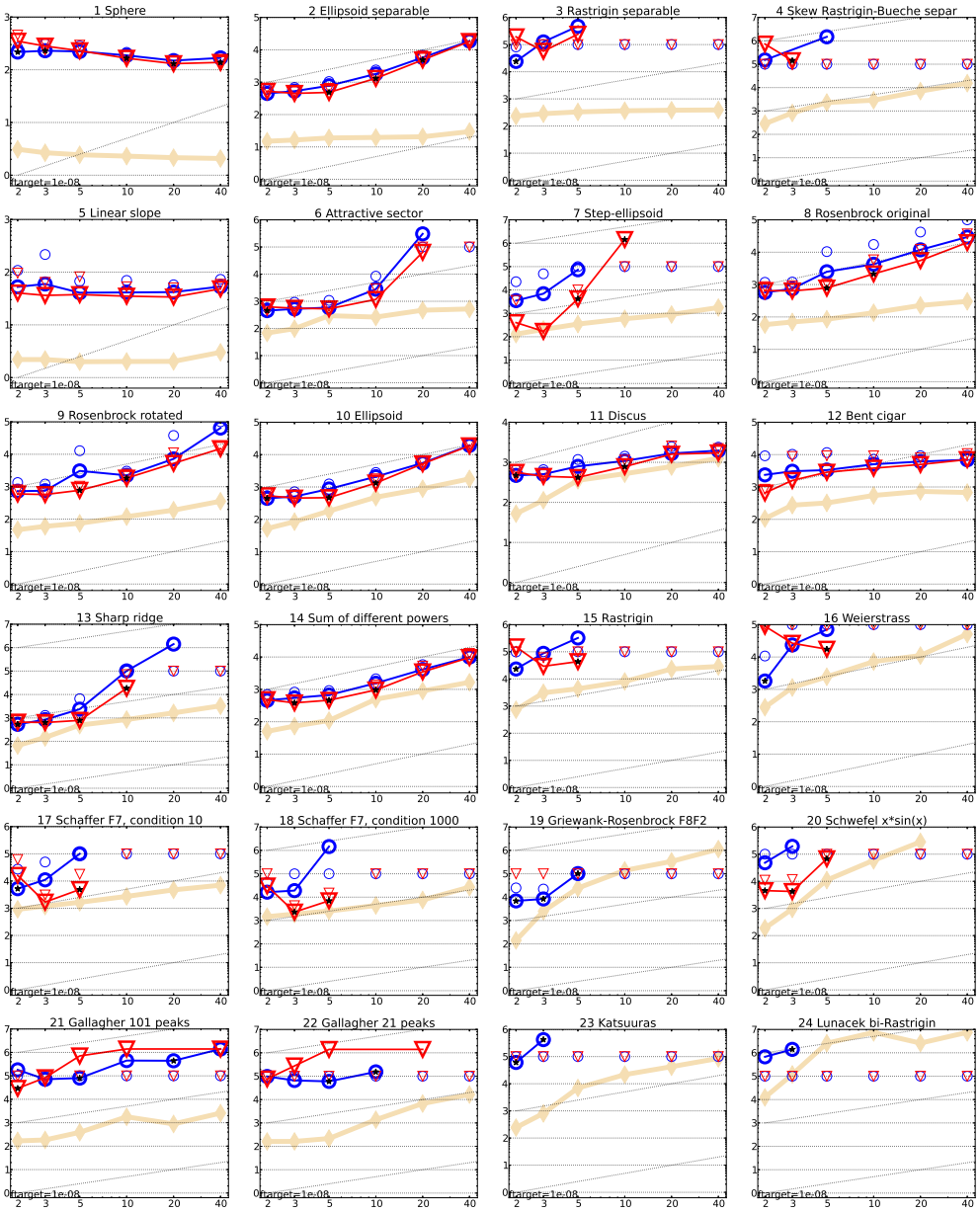

Figure 5.26: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function 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: ○:CMSA, ▽:Derandomized-stepsize CMSA

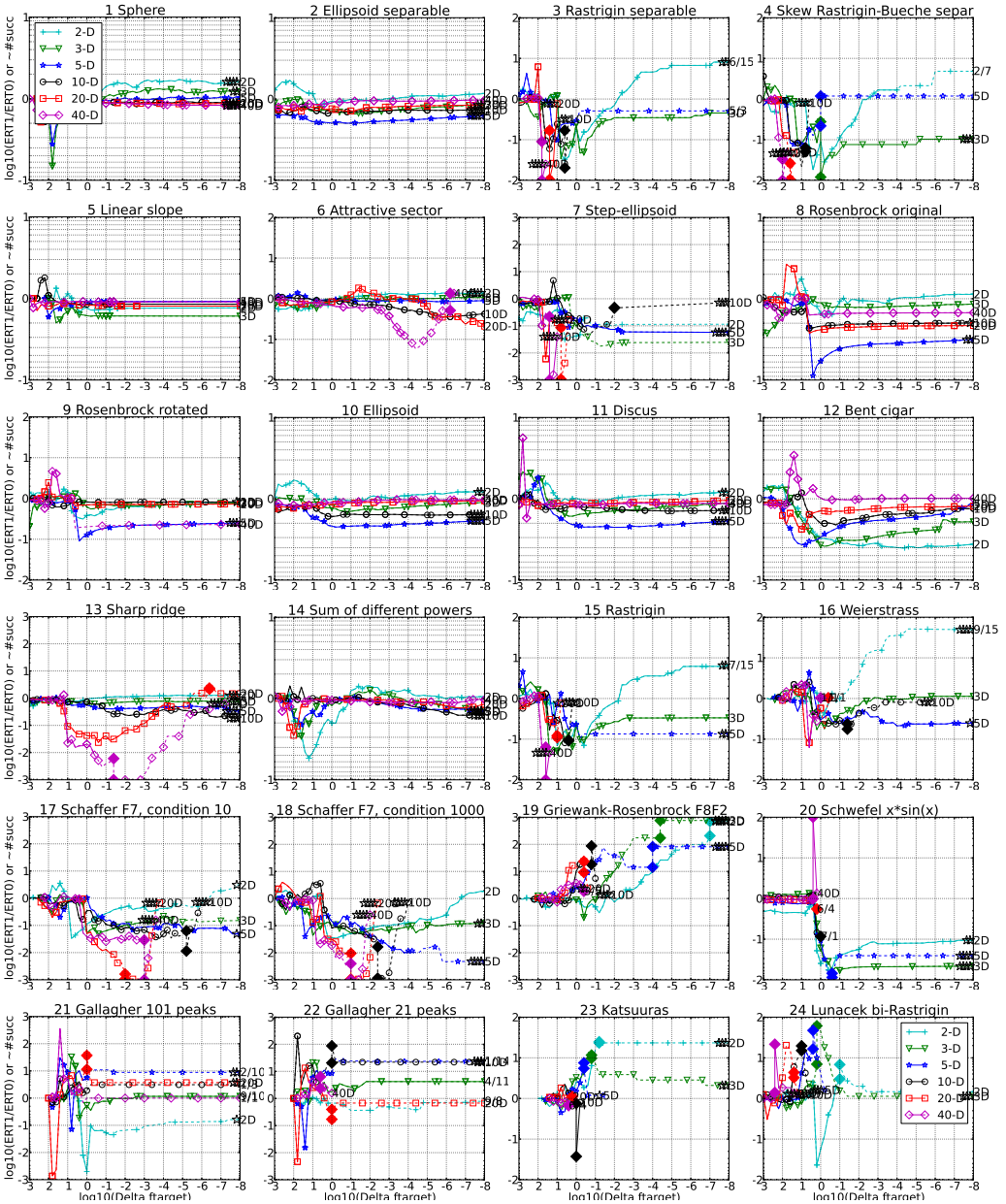

Figure 5.27: Ratio of ERT for Derandomized-stepsize CMSA over ERT for CMSA versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:◇. Ratios $< 10^0$ indicate an advantage of Derandomized-stepsize CMSA, smaller values are always better. The line becomes dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall number of f-evaluations for the same algorithm on this function. Filled symbols indicate the best achieved Δf-value of one algorithm (ERT is undefined to the right). The dashed line continues as the fraction of successful trials of the other algorithm, where 0 means 0% and the y-axis limits mean 100%, values below zero for Derandomized-stepsize CMSA. The line ends when no algorithm reaches Δf anymore. The number of successful trials is given, only if it was in $\{1 \ldots 9\}$ for Derandomized-stepsize CMSA (1^{st} number) and non-zero for CMSA (2^{nd} number). Results are significant with $p = 0.05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni correction within each figure.

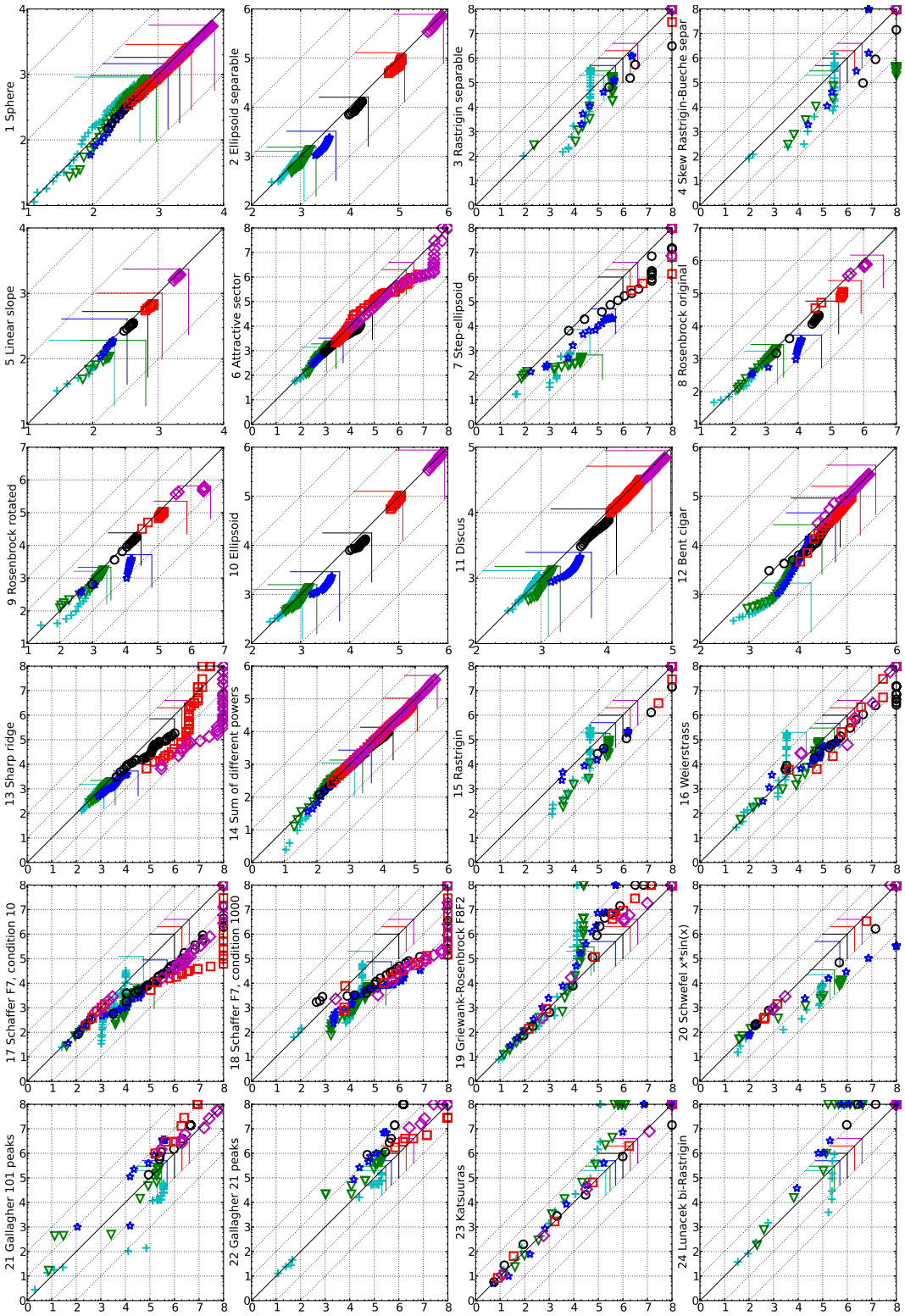

Figure 5.28: Expected running time (ERT in $\log_{10}$ of number of function evaluations) of CMSA (x-axis) versus Derandomized-stepsize CMSA (y-axis) for 46 target values $\Delta f \in [10^{-8}, 10]$ in each dimension on functions f_1–f_{24}. Markers on the upper or right edge indicate that the target value was never reached. Markers represent dimension: 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40:◇.

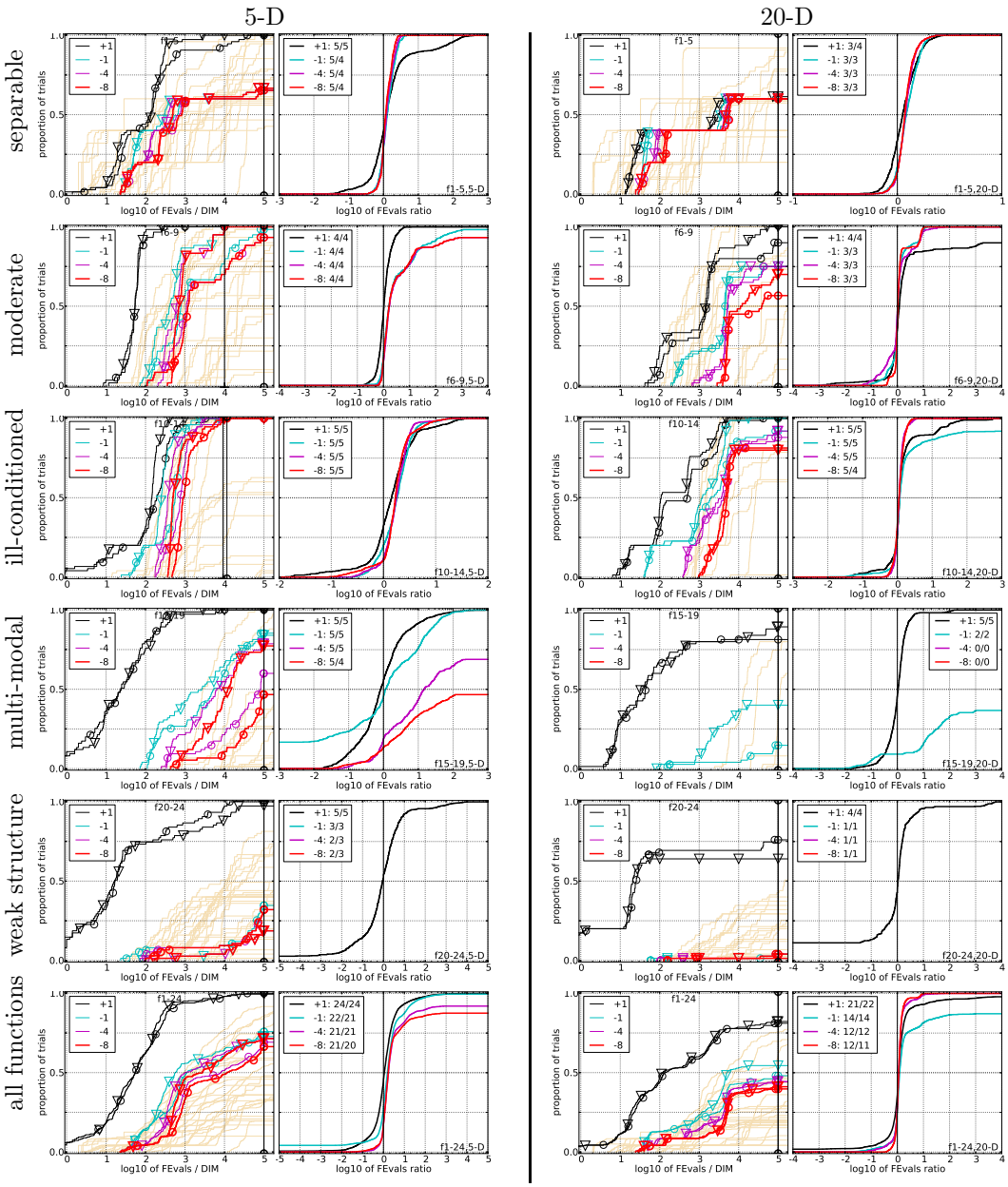

Figure 5.29: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for CMSA ($\circ$) and Derandomized-stepsize CMSA ($\triangledown$). 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 CMSA divided by Derandomized-stepsize CMSA, 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 (CMSA first).

5-D

Δf	1e+1	1e-1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_1}$	11	12	12	12	12	15/15
1: CMSA	8.4(6)	24(6)	43(10)	62(12)	81(8)	15/15
2: DS	5.4(4)	23(4)	42(5)	63(8)	85(10)	15/15
$\mathbf{f_2}$	83	88	90	92	94	15/15
1: CMSA	24(13)	33(9)	37(12)	39(11)	40(11)	15/15
2: DS	13(5)	**17**(5)$^{\star 3}$	**20**(5)$^{\star 3}$	**22**(5)$^{\star 3}$	**24**(5)$^{\star 3}$	15/15
$\mathbf{f_3}$	716	1637	1646	1650	1654	15/15
1: CMSA	29(53)	1444(1385)	1436(1519)	1433(1537)	1430(1383)	3/15
2: DS	2.9(3)	735(830)	732(807)	730(767)	728(822)	5/15
$\mathbf{f_4}$	809	1688	1817	1886	1903	15/15
1: CMSA	32(71)	4400(5184)	4087(4402)	3938(4375)	3903(4466)	1/15
2: DS	2.4(0.6)	∞	∞	∞	∞ *5.0e5*	0/15
$\mathbf{f_5}$	10	10	10	10	10	15/15
1: CMSA	13(7)	20(7)	20(7)	20(7)	20(7)	15/15
2: DS	11(3)	19(8)	19(8)	19(8)	19(8)	15/15
$\mathbf{f_6}$	114	281	580	1038	1332	15/15
1: CMSA	2.6(1)	3.1(1)	2.5(1.0)	2.0(0.7)	2.0(0.7)	15/15
2: DS	2.4(0.8)	2.8(0.6)	2.2(0.2)	1.7(0.2)	1.8(0.2)	15/15
$\mathbf{f_7}$	24	1171	1572	1572	1597	15/15
1: CMSA	7.5(5)	146(164)	233(230)	233(230)	230(264)	11/15
2: DS	5.9(2)	**15**(21)$^{\star 2}$	**13**(15)$^{\star 3}$	**13**(15)$^{\star 3}$	**13**(15)$^{\star 3}$	15/15
$\mathbf{f_8}$	73	336	391	410	422	15/15
1: CMSA	5.2(1)	32(56)	30(48)	29(46)	29(44)	15/15
2: DS	4.5(1)	7.7(3)	**8.3**(2)*	**8.6**(2)*	**8.9**(2)*	15/15
$\mathbf{f_9}$	35	214	300	335	369	15/15
1: CMSA	12(8)	64(114)	49(81)	45(72)	41(66)	15/15
2: DS	10(3)	**12**(6)$^{\star 2}$	**11**(4)$^{\star 2}$	**10**(3)$^{\star 2}$	**10**(3)$^{\star 2}$	15/15
$\mathbf{f_{10}}$	349	574	626	829	880	15/15
1: CMSA	4.8(3)	5.5(2)	5.9(2)	4.8(2)	4.8(2)	15/15
2: DS	2.9(1)	**2.5**(0.6)$^{\star 2}$	**2.8**(0.6)$^{\star 3}$	**2.4**(0.4)$^{\star 3}$	**2.5**(0.4)$^{\star 3}$	15/15
$\mathbf{f_{11}}$	143	763	1177	1467	1673	15/15
1: CMSA	10(4)	3.6(2)	2.8(1.0)	2.5(0.9)	2.3(0.8)	15/15
2: DS	6.1(2)	**1.6**(0.4)$^{\star 3}$	**1.3**(0.2)$^{\star 3}$	**1.2**(0.2)$^{\star 3}$	**1.2**(0.2)$^{\star 3}$	15/15
$\mathbf{f_{12}}$	108	371	461	1303	1494	15/15
1: CMSA	35(69)	25(35)	25(34)	11(14)	11(14)	15/15
2: DS	10(5)	11(7)	14(10)	6.4(5)	7.6(8)	15/15
$\mathbf{f_{13}}$	132	250	1310	1752	2255	15/15
1: CMSA	6.1(5)	12(7)	3.5(2)	3.4(1)	3.4(2)	15/15
2: DS	3.8(1)	**4.8**(2)$^{\star 2}$	**1.5**(0.5)$^{\star 3}$	**1.7**(0.5)$^{\star 3}$	**1.6**(0.4)$^{\star 3}$	15/15
$\mathbf{f_{14}}$	10	58	139	251	476	15/15
1: CMSA	5.3(6)	5.6(2)	6.2(2)	7.4(2)	5.8(1)	15/15
2: DS	3.7(3)	5.5(2)	5.2(0.8)	5.4(1)	**4.3**(0.6)$^{\star 2}$	15/15
$\mathbf{f_{15}}$	511	19369	20073	20769	21359	14/15
1: CMSA	6.4(1)	83(87)	80(94)	78(91)	76(82)	4/15
2: DS	4.5(2)	11(14)	11(13)	10(12)	10(12)	13/15
$\mathbf{f_{16}}$	120	2662	10449	11644	12095	15/15
1: CMSA	3.1(6)	29(40)	16(15)	30(38)	29(33)	11/15
2: DS	2.6(3)	7.0(7)	5.7(4)	7.2(5)	7.2(5)	15/15
$\mathbf{f_{17}}$	5.2	899	3669	6351	7934	15/15
1: CMSA	9.1(10)	5.9(12)	11(16)	20(20)	39(42)	9/15
2: DS	7.0(9)	0.84(0.3)	0.94(1)	1.5(1)	3.0(4)	15/15
$\mathbf{f_{18}}$	103	3968	9280	10905	12469	15/15
1: CMSA	21(1)	7.8(10)	55(57)	152(164)	585(581)	1/15
2: DS	2.3(2)	0.84(0.8)	**0.93**(0.8)$^{\star 3}$	**2.2**(3)$^{\star 3}$	**2.8**(3)$^{\star 3}$	15/15
$\mathbf{f_{19}}$	1	242	1.2e5	1.2e5	1.2e5	15/15
1: CMSA	26(25)	**229**(334)$^{\star 3}$	**4.2**(4)$^{\star 2}$	**4.2**(4)$^{\star 3}$	**4.2**(4)$^{\star 3}$	10/15
2: DS	29(25)	5114(5405)	60(65)	∞	∞ *5.0e5*	0/15
$\mathbf{f_{20}}$	16	38111	54470	54861	55313	14/15
1: CMSA	6.1(4)	∞	∞	∞	∞ *5.0e5*	0/15
2: DS	4.6(3)	**8.9**(11)$^{\star 3}$	**6.2**(9)$^{\star 3}$	**6.2**(7)$^{\star 3}$	**6.1**(7)$^{\star 3}$	11/15
$\mathbf{f_{21}}$	41	1674	1705	1729	1757	14/15
1: CMSA	2.7(3)	**194**(245)*	**235**(293)*	**232**(289)*	**228**(285)*	10/15
2: DS	24(3)	2115(2389)	2077(2199)	2049(2221)	2016(2134)	2/15
$\mathbf{f_{22}}$	71	938	1008	1040	1068	14/15
1: CMSA	206(411)	**310**(225)$^{\star 2}$	**297**(209)$^{\star 2}$	**288**(203)$^{\star 2}$	**281**(197)$^{\star 2}$	14/15
2: DS	1209(3521)	7465(8798)	6944(7935)	6729(7330)	6558(7260)	1/15
$\mathbf{f_{23}}$	3.0	14249	31654	33030	34256	15/15
1: CMSA	1.8(1)	511(561)	∞	∞	∞ *5.0e5*	0/15
2: DS	1.7(2)	∞	∞	∞	∞ *5.0e5*	0/15
$\mathbf{f_{24}}$	1622	6.4e6	9.6e6	1.3e7	1.3e7	3/15
1: CMSA	5.5(6)	∞	∞	∞	∞ *5.0e5*	0/15
2: DS	23(30)	∞	∞	∞	∞ *5.0e5*	0/15

20-D

Δf	1e+1	1e-1	1e-3	1e-5	1e-7	#succ
$\mathbf{f_1}$	43	43	43	43	43	15/15
1: CMSA	8.1(1)	21(3)	35(4)	48(4)	62(4)	15/15
2: DS	7.9(1)	19(2)	**31**(2)*	**42**(3)$^{\star 2}$	**55**(4)$^{\star 3}$	15/15
$\mathbf{f_2}$	385	387	390	391	393	15/15
1: CMSA	161(61)	258(61)	277(40)	285(35)	291(28)	15/15
2: DS	128(31)	**185**(38)$^{\star 2}$	**221**(34)$^{\star 2}$	**235**(46)*	**244**(52)*	15/15
$\mathbf{f_3}$	5066	7635	7643	7646	7651	15/15
1: CMSA	∞	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	**5819**(6514)$^{\star 2}$	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_4}$	4722	7666	7700	7758	1.4e5	9/15
1: CMSA	∞	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	∞	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_5}$	41	41	41	41	41	15/15
1: CMSA	16(4)	20(6)	21(6)	21(6)	21(6)	15/15
2: DS	13(4)	17(4)	17(4)	17(4)	17(4)	15/15
$\mathbf{f_6}$	1296	3413	5220	6728	8409	15/15
1: CMSA	2.1(0.9)	3.1(3)	18(10)	53(67)	266(298)	4/15
2: DS	1.7(0.6)	3.9(4)	17(26)	34(24)	63(80)	12/15
$\mathbf{f_7}$	1351	9503	16524	16524	16969	15/15
1: CMSA	1616(2018)	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	211(262)	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_8}$	2039	4040	4219	4371	4484	15/15
1: CMSA	16(5)	55(86)	55(82)	53(79)	53(77)	15/15
2: DS	17(7)	24(5)	25(5)	25(4)	25(4)	15/15
$\mathbf{f_9}$	1716	3277	3455	3594	3727	15/15
1: CMSA	19(5)	40(6)	40(6)	39(6)	38(6)	15/15
2: DS	19(5)	28(4)	29(4)	28(4)	28(4)	15/15
$\mathbf{f_{10}}$	7413	10735	14920	17073	17476	15/15
1: CMSA	9.1(3)	8.6(2)	6.8(0.8)	6.2(0.7)	6.2(0.7)	15/15
2: DS	7.4(4)	7.7(2)	6.2(0.7)	5.7(0.7)	5.8(0.7)	15/15
$\mathbf{f_{11}}$	1002	6278	9762	12285	14831	15/15
1: CMSA	11(2)	2.7(0.5)	2.3(0.5)	2.2(0.5)	2.1(0.8)	15/15
2: DS	10(0.9)	2.4(0.5)	2.1(0.4)	2.0(0.6)	2.0(0.8)	15/15
$\mathbf{f_{12}}$	1042	2740	4140	12407	13827	15/15
1: CMSA	11(14)	22(13)	20(10)	8.3(3)	8.5(3)	15/15
2: DS	**4.5**(0.2)*	15(11)	15(9)	6.6(3)	6.8(3)	15/15
$\mathbf{f_{13}}$	652	2751	18749	24455	30201	15/15
1: CMSA	112(70)	685(780)	205(233)	368(368)	937(1093)	1/15
2: DS	10(16)	**28**(17)*	**24**(26)*	200(235)	∞ *2.0e6*	0/15
$\mathbf{f_{14}}$	75	304	932	1648	15661	15/15
1: CMSA	3.8(2)	3.4(0.7)	5.6(0.8)	12(2)	3.6(1)	15/15
2: DS	3.6(1.0)	3.3(0.4)	5.3(0.6)	10(2)	2.8(0.8)	15/15
$\mathbf{f_{15}}$	30378	3.1e5	3.2e5	4.5e5	4.6e5	15/15
1: CMSA	922(1020)	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	101(112)	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{16}}$	1384	77015	1.9e5	2.0e5	2.2e5	15/15
1: CMSA	3.0(3)	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	4.3(2)	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{17}}$	63	4005	30677	56288	80472	15/15
1: CMSA	3.8(3)	502(545)	∞	∞	∞ *2.0e6*	0/15
2: DS	3.0(2)	**6.5**(5)*	**16**(16)$^{\star 3}$	∞	∞ *2.0e6*	0/15
$\mathbf{f_{18}}$	621	19561	67569	1.3e5	1.5e5	15/15
1: CMSA	9.0(0.7)	701(716)	∞	∞	∞ *2.0e6*	0/15
2: DS	1.2(0.4)	**6.7**(8)$^{\star 2}$	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{19}}$	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
1: CMSA	148(42)	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	139(36)	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{20}}$	82	3.1e6	5.5e6	5.6e6	5.6e6	14/15
1: CMSA	4.9(1)	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	4.3(1)	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{21}}$	561	14103	14643	15567	17589	15/15
1: CMSA	296(757)	625(709)	602(718)	566(631)	501(598)	3/15
2: DS	1782(3563)	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{22}}$	467	23491	24948	26847	1.3e5	12/15
1: CMSA	1501(2142)	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	3750(5356)	1192(1320)	1123(1283)	1043(1155)	208(237)	1/15
$\mathbf{f_{23}}$	3.2	67457	4.9e5	8.1e5	8.4e5	15/15
1: CMSA	2.5(4)	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	2.7(3)	∞	∞	∞	∞ *2.0e6*	0/15
$\mathbf{f_{24}}$	1.3e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
1: CMSA	∞	∞	∞	∞	∞ *2.0e6*	0/15
2: DS	∞	∞	∞	∞	∞ *2.0e6*	0/15

Table 5.13: 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 different Δf values. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. 1:CMSA is CMSA and 2:DS is Derandomized-stepsize CMSA. 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.

Chapter 6

Conclusion

The variety of derandomized techniques plays an important role to improve the performance of evolution strategies. Historically, the process of discovering undesired randomnesses and developing new techniques to remove them has already changed the ES adaptation mechanism from the mutative-self-adaption framework (MSC-ES) to the covariance-matrix-adaptation technique (CMA-ES) as argued in Chapter 2. Recall the hierarchical structure of ES in Chapter 2, this derandomization happens in the level 2 and helps to improve the reliability of the endogenous strategy parameter tuning. Recently, some unpleasant randomnesses are gradually discovered in the mutation operator or say the sampling process, which is located at the level 1. Then some new mutation operators are designated to remove that randomness without violating the operator's major functionalities. Although we still categorize such new mutation operators as instances of derandomization techniques, their essences differ from that in history.

The *mirrored sampling* technique is the first solution to the sampling errors in mutation. The technique is both quite simple in idea and implementation. It requires basically no additional computational cost compared to the standard mutation operator. It can be applied to any single parent ES algorithm without introducing any side effect. When the population is small, it actually improves the ES convergence velocity significantly as discussed in Section 3.2. If applied in multi-parental ES algorithm, a side effect, which is the systematic reduction of the sampling variance, occurs due to the pairwise cancellation of mirrored pair of offspring if they are selected at the same time. Such side effect can facilitate premature convergence and thus is undesired.

In order to solve the problem, a new variant of mirrored sampling, named *"noisy" mirrored sampling* technique is proposed and tested in BBOB software. The "noisy" mirrored sampling technique is only capable of reducing the side effect instead of completely removing it. The degree of the reduction can be controlled by two new parameters. However, the additional computation overheads of the "noisy" mirrored sampling is very high because it requires a N dimensional rotation matrix, which costs $O(N^2)$ matrix multiplications of N dimensional square matrix. In total, the time complexity is $O(N^5)$ if the matrix multiplication is implemented naively, or roughly $O(N^{4.38})$ if using Coppersmith–Winograd algorithm [16] for matrix multiplication. Thus, in either way, this technique is not going to be efficient. The experiment results (shown in Section 5.4.2) implies that it actually also reduces the amount of improvement of ERTs introduced by mirrored sampling. The "noisy" mirrored sampling

seems not as satisfactory as what we expected and works as our one attempt to solve the side effect in mirrored sampling.

Although the limitations of mirror sampling (small population, single parent) restrict its application, the motivation behind it, which is to generate more even mutation samples, is really essential and is the main topic of Chapter 4. The derandomized sampling is aiming at designing novel mutation operators to reduce the sampling errors in mutation without bringing additional side effects. Because the mutation vectors in ES contain two independent components, the direction and length of the vector, the derandomized sampling can be further developed in two branches, the direction derandomization and the step-size (length) derandomization.

The mirrored sampling itself could be considered as an example of derandomized direction method because the direction of some mutations are restricted. Furthermore, the idea of *orthogonal sampling* is proposed in Section 4.1 as a generalization of mirrored sampling. The idea is to generate pairwise orthogonal mutation vectors. The benefits of it is that: First, the difference between mutations are guaranteed to be large enough. In other words, the direction exploration is increased. Second, their is no side effects discovered so far. Thus, the systemic reduction of sampling variance is avoided.

The orthogonal sampling idea can be implemented in two different methods. The first method is named as ORTHOGONAL-SAMPLING1 and uses random rotations. Its computation costs is identical to that of "noisy" mirrored sampling. The second one is named as ORTHOGONAL-SAMPLING2 and uses Gram-Schmidt process. There is no additional parameters for this method and the additional computational overheads are just marginal. The empirical results in Section 5.4.6, 5.4.7 show that both of these two methods can improve the ES performance.

The *step-size derandomization* method is intended to solve the ambiguity in Covariance Matrix Adaptation (CMA) mechanism. Again, the no additional computations are needed by this new operator. However, it violates the bases of the Cumulative Step-size Adaptation (CSA) in CMA-ES so that we could not implement it into CMA-ES to verify its performance. Instead, it is applied in CMSA-ES which also exploits Covariance Matrix Adaptation technique. Its empirical results in Section 5.4.6 shows that such new mutation operator actually largely improves the performance of CMSA algorithm compared to the standard CMSA-ES.

As another potential task of this thesis, the bias of the operators in ES is also considered. The mirrored sampling technique introduces a bias, which is the the systematic reduction of the sampling variance. Such bias is reduced in some degree by the "noisy"-mirrored sampling technique. There is no bias discovered in complete direction derandomization sampling or step-size derandomization sampling. As a disturbing discovery in Section 2.2.4. The widely used recombination operator in MSC-ES is actually biased. Such bias is verified both experimentally and theoretically. A new unbiased recombination operator is also developed. However, no further comparisons and experiments are conducted for this problem. The problem is still a mystery at least for the author and we welcome any possible idea about it.

Bibliography

[1] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. Benchmarking the (1,4)-cma-es with mirrored sampling and sequential selection on the noiseless bbob-2010 testbed. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, GECCO '10, pages 1617–1624, New York, NY, USA, 2010. ACM.

[2] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. Benchmarking the (1,4)-cma-es with mirrored sampling and sequential selection on the noisy bbob-2010 testbed. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, GECCO '10, pages 1625–1632, New York, NY, USA, 2010. ACM.

[3] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. Mirrored variants of the (1,2)-cma-es compared on the noiseless bbob-2010 testbed. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, GECCO '10, pages 1551–1558, New York, NY, USA, 2010. ACM.

[4] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. Mirrored variants of the (1,2)-cma-es compared on the noisy bbob-2010 testbed. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, GECCO '10, pages 1575–1582, New York, NY, USA, 2010. ACM.

[5] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. Analyzing the impact of mirrored sampling and sequential selection in elitist evolution strategies. In *Foundations of Genetic Algorithms (FOGA 2011)*, pages 127–138, Schwarzenberg, Autriche, April 2011.

[6] Anne Auger, Dimo Brockhoff, and Nikolaus 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.

[7] Anne Auger and Nikolaus Hansen. Theory of evolution strategies: A new perspective. In A. Auger and B. Doerr, editors, *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, chapter 10, pages 289–325. World Scientific Publishing Company, 2011.

[8] Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1991.

[9] Hans-Georg Beyer. Toward a theory of 'evolution strategies': Some asymptotical results from the $(1,^+\lambda)$-theory. *Evolution Computation*, 1(2):165–188, June 1993.

[10] Hans-Georg Beyer. Toward a theory of evolution strategies: The (μ, λ)-theory. *Evol. Comput.*, 2(4):381–407, December 1994.

[11] Hans-Georg Beyer. *The theory of evolution strategies*. Springer-Verlag New York Incorporated, 2001.

[12] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies: A comprehensive introduction. 1(1):3–52, May 2002.

[13] Hans-Georg Beyer and Bernhard Sendhoff. Covariance matrix adaptation revisited — the cmsa evolution strategy —. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*, pages 123–132, Berlin, Heidelberg, 2008. Springer-Verlag.

[14] Å. Björck. Numerics of gram-schmidt orthogonalization. *Linear Algebra and its Applications*, 197–198(0):297 – 316, 1994.

[15] Dimo Brockhoff, Anne Auger, Nikolaus Hansen, Dirk V. Arnold, and Tim 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.

[16] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 1–6, New York, NY, USA, 1987. ACM.

[17] H. A. David and H. N. Nagaraja. *Order Statistics*. John Wiley & Sons, Inc., 2004.

[18] KT Fang, S Kotz, and KW Ng. *Symmetric Multivariate and Related Distributions Monographs on Statistics and Applied Probability*. London: Chapman and Hall Ltd. MR1071174, 1990.

[19] 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.

[20] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2010: Presentation of the noisless functions. Technical report, Citeseer, 2010.

[21] A.K. Gupta and D. Song. L_p-norm spherical distribution. *Journal of Statistical Planning and Inference*, 60(2):241 – 260, 1997.

[22] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report RR-7215, INRIA, 2010.

[23] 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.

[24] Nikolaus Hansen. The cma evolution strategy: A tutorial, 2005.

[25] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, June 2001.

[26] Nikolaus Hansen, Andreas Ostermeier, and Andreas Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 57–64, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[27] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions, Vol. 1 (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2 edition.

[28] Shuhei Kimura and Koki Matsumura. Genetic algorithms using low-discrepancy sequences. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1341–1346, New York, NY, USA, 2005. ACM.

[29] B.W. Lindgren. *Statistical Theory*. Chapman and Hall/CRC Texts in Statistical Science Series. Chapman and Hall, 1993.

[30] Travis E. Oliphant. *Guide to Numpy*. December 2006.

[31] Andreas Ostermeier, Andreas Gawelczyk, and Nikolaus Hansen. A derandomized approach to self-adaptation of evolution strategies. *Evol. Comput.*, 2(4):369–380, December 1994.

[32] Andreas Ostermeier, Andreas Gawelczyk, and Nikolaus Hansen. Step-size adaption based on non-local use of selection information. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, PPSN III, pages 189–198, London, UK, UK, 1994. Springer-Verlag.

[33] Kenneth Price. Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation*, pages 153–157, 1997.

[34] K.V. Price. Differential evolution vs. the functions of the 2nd iceo. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 153–157, 1997.

[35] Ingo Rechenberg. Evolutionsstrategie: Optimierung technisher systeme nach prinzipien der biologischen evolution. 1973.

[36] Hans-Paul Schwefel. *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*. Technische Universität, Berlin., 1965.

[37] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 1981.

[38] D Song and AK Gupta. L_p-norm uniform distribution. *Proceedings of the American Mathematical Society*, 125(2):595–602, 1997.

[39] Olivier Teytaud and Sylvain Gelly. Dcma: yet another derandomization in covariance-matrix-adaptation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 955–963, New York, NY, USA, 2007. ACM.