Universiteit
Leiden
The Netherlands

Algorithm design for mixed-integer black-box optimization problems with uncertainty

Thomaser, A.M.

Citation
Thomaser, A. M. (2024, October 22). *Algorithm design for mixed-integer black-box optimization problems with uncertainty*. Retrieved from https://hdl.handle.net/1887/4104741

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/4104741

Note: To cite this publication please use the final published version (if applicable).

Algorithm Design for Mixed-Integer Black-Box Optimization Problems with Uncertainty

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op dinsdag 22 oktober 2024
klokke 11:30 uur

door

André Thomaser

geboren te München, Duitsland
in 1996

Promotor:

Prof.dr. T.H.W. Bäck

Co-promotor:

Dr. A.V. Kononova

Promotiecomissie:

Prof.dr. M.M. Bonsangue
Prof.dr. A. Plaat
Dr. N. van Stein
Dr. E. Raponi
Prof.dr. B. Sendhoff (Technical University Darmstadt, Germany)
Prof.dr. K.M. Malan (University of Pretoria, South Africa)

Copyright © 2024 André Thomaser

This thesis was written as part of the research project newAIDE under the consortium leadership of BMW AG with the partners Altair Engineering GmbH, divis intelligent solutions GmbH, MSC Software GmbH, Technical University of Munich, TWT GmbH. The project is financially supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision of the German Bundestag.

To my parents.

“No problem can withstand the assault of sustained thinking.”

Voltaire, 1694 - 1778

Contents

1 Introduction **1**

1.1 Background 1
1.2 Research Questions 2
1.3 Contribution and Structure of the Thesis 3
1.4 Publications 4

2 Preliminaries **7**

2.1 Optimization 7
2.2 Evolutionary Algorithms 8
2.3 Evolution Strategies 9
2.4 Covariance Matrix Adaption Evolution Strategy 10
2.4.1 Variants 12
2.4.2 Box Constraint Handling 14
2.4.3 Integer Handling 16
2.5 Objective Function Landscape 17
2.5.1 Objective Function Properties 17
2.5.2 Exploratory Landscape Analysis 18
2.6 Benchmarking and Tuning 20
2.6.1 Benchmark Problems 20
2.6.2 Performance Metric 21
2.6.3 Parameter Tuning 23
2.7 Uncertainty Quantification 24
2.7.1 Static and Dynamic Allocation 25
2.7.2 Dynamic Allocation for Ranking and Selection 26
2.7.3 Confidence Interval Sequences 27
2.7.4 Uncertainty Quantification in Ranking and Selection 31

3 Engineering Problems **35**
3.1 Antilock Braking System . . . 36
3.2 Active Rollover Protection . . . 38
3.3 Vehicle Dynamics Modeling and Simulation . . . 39
3.4 ABS Benchmark Dataset . . . 40

4 Tuning CMA-ES Parameters **43**
4.1 Methodology . . . 45
4.1.1 Similarity Quantification . . . 46
4.1.2 Artificial Functions . . . 47
4.2 Analysis of Landscape Similarities . . . 48
4.2.1 Experimental Setup . . . 48
4.2.2 Results . . . 49
4.2.3 Conclusion . . . 55
4.3 Brute Force Search . . . 56
4.3.1 Experimental Setup . . . 56
4.3.2 Results . . . 58
4.3.3 Summary . . . 63
4.4 Meta-Optimization . . . 64
4.4.1 Experimental Setup . . . 65
4.4.2 Results . . . 66
4.5 Tuning for Higher Dimensional Problems . . . 68
4.5.1 Experimental Setup . . . 68
4.5.2 Results . . . 69
4.6 Comparison of Meta-Optimization Algorithms . . . 71
4.6.1 Experimental Setup . . . 72
4.6.2 Results . . . 73
4.7 Conclusion . . . 75

5 Handling Discretization **77**
5.1 Function Discretization . . . 77
5.2 EA for Integer Programming . . . 80
5.3 Experimental Setup . . . 81
5.4 Results . . . 81
5.4.1 Success Rates . . . 83
5.4.2 ECDF . . . 87
5.5 Conclusion . . . 90

6 Uncertainty Quantification **91**
6.1 Noisy Test Function 92
6.2 Methodology 94
6.2.1 Dynamic Allocation Policy 95
6.2.2 Top-μ Selection and Ranking 97
6.2.3 CMA-ES and Bound Adaption for UQiS 101
6.2.4 CMA-ES Convergence 103
6.2.5 Conclusion 109
6.3 Application to Real-World Problems 110

7 Conclusion and Outlook **115**

Bibliography **i**

List of Abbreviations **xvii**

Samenvatting **xix**

Summary **xxi**

Curriculum Vitae **xxiii**

Chapter 1

Introduction

1.1 Background

The automotive industry is confronted with increasing competition, requiring the manufacturers to develop technologically advanced vehicles while meeting tighter schedules. As a solution approach to succeed in this highly competitive environment, the companies combine state-of-the-art simulation technologies with the growing availability of computing resources to improve product quality, accelerate time to market and optimize costs.

An important task in the automotive industry is designing the parameters of vehicle dynamics control systems. Traditionally, this process is carried out on test tracks with the physical vehicle and relies heavily on the expertise of the engineer. However, virtual simulation environments are being developed to pre-design the parameters of a control system virtually and create a preliminary parameter configuration that can serve as the basis for the final tuning on the test track. This is accomplished by modeling the complete vehicle and its control systems. By employing optimization algorithms, the virtual pre-design can be transformed into an automated, data-driven decision-making process. The aim thereby is to provide a scalable, efficient and robust method for the industry to cope with the increasing competition.

To fully utilize these advanced methodologies, numerous open questions and challenges need to be addressed in order to tailor the process for industrial application. This thesis concentrates on the following research questions (Section 1.2). An overview of the contributions and the structure of this thesis is presented in Section 1.3.

1.2 Research Questions

This thesis is centered around a primary research question (RQ):

> *How can an optimization algorithm be effectively and efficiently designed to solve computationally expensive real-world problems, such as the design of vehicle dynamics control system parameters?*

To address this overarching question, the thesis will delve into seven sub-questions, which are presented as the main research questions below.

The basic prerequisite in applying an optimization algorithm is the availability of a formal definition of the objective function to be optimized. Thus, the first research question arises:

RQ 1: *How can the desired behavior of the control system and the vehicle be objectified to define the real-world problem mathematically by an objective function?*

Once the optimization problem is formulated, identifying a suitable class of optimization algorithms becomes possible based on the properties of the problem. Besides these properties, the performance of an optimization algorithm is influenced by its parameters, leading to the second and third research questions:

RQ 2: *How can an optimal parameter configuration of an optimization algorithm for a specific real-world problem be determined?*

RQ 3: *Does a single parameter configuration suffice for a class of related real-world problems, such as the design of vehicle dynamics control system parameters, or is each problem unique, necessitating a tailored parameter configuration?*

Considering that tuning the parameters of an optimization algorithm is an optimization task in itself, the fourth research question is posed:

RQ 4: *What methods can efficiently determine the optimal parameters of an optimization algorithm for solving a specific optimization problem?*

Many real-world problems, including vehicle dynamics design, involve discrete parameters. This leads to the fifth research question:

RQ 5: *How can the discretization of parameter values be handled within an optimization algorithm, and how does this affect the performance of an optimization algorithm?*

Finally, the variability of the objective function due to measurement errors, external disturbances or the stochastic nature of a system must be addressed, leading to the sixth and seventh research questions:

RQ 6: *How can the uncertainty inherent in the variability of an objective function be quantified?*

RQ 7: *Can this uncertainty quantification be incorporated into an optimization algorithm to reduce computational resources?*

1.3 Contribution and Structure of the Thesis

This thesis systematically addresses the seven research questions posed in Section 1.2. Chapter 2 lays the groundwork with a comprehensive overview of the theoretical underpinnings of optimization, the intricacies of algorithm tuning and the principles of uncertainty quantification.

Chapter 3 focuses on the design of vehicle dynamics control systems. The desired behavior of two vehicle dynamics control systems is objectified to define the engineering problems mathematically by objective functions. In addition, a comprehensive dataset is created to represent real-world optimization problems for five different vehicle settings (Section 3.4). This dataset removes the need to run extensive simulations and, thus, reduces the computational cost of evaluations from several minutes to milliseconds, enabling benchmarking and algorithm design directly on the five real-world problems.

In Chapter 4, the properties of the five different real-world problems derived from the created dataset (Chapter 3) and the performance impacts of different parameter configurations on the optimization algorithms are analyzed. A method for tuning the parameters of the optimization algorithm to computationally expensive problems is presented, in which surrogate problems are used that approximate the properties of the original optimization problem. The effectiveness of this method for tuning optimization algorithms for real-world applications is validated through extensive experiments. In addition, Section 4.6 provides a comparison of different optimization algorithms in the context of parameter tuning.

In Chapter 5, the effects of discretized parameter values on the performance of optimization algorithms are analyzed. A method for handling discretization is considered. In addition, recommendations are given for adjusting the parameters of the algorithm to handle discrete variables more efficiently.

Chapter 6 presents a novel methodology that combines uncertainty quantification with optimization algorithms. This method quantifies the uncertainty in order to reduce the number of evaluations required. The effectiveness of the methodology and its potential for saving evaluations are examined using a test function and the aforementioned dataset.

In Chapter 7, the thesis is concluded with a discussion of the results and implications. Moreover, directions for future research are given.

1.4 Publications

Portions of this thesis incorporate material previously published in manuscripts by the author and collaborators, as detailed in the list of contributions below. These publications focus on aspects of the development of the method for tuning the parameters of optimization algorithms (Chapter 4) for computationally expensive real-world problems (Chapter 3). The first publication quantifies and analyzes the landscape of the optimization problem using features. The second publication examines the effect of tuning the optimization algorithm parameters on solution quality or wall-clock time. The third publication analyzes the impact of parameter discretization (Chapter 5) on the performance of various optimization algorithms. The fourth publication deals with identifying similar problems for tuning optimization algorithms. The final publication compares different optimization algorithms for optimizing the optimization algorithm.

1) André Thomaser, Anna V. Kononova, Marc-Eric Vogt, and Thomas Bäck. One-Shot Optimization for Vehicle Dynamics Control Systems: Towards Benchmarking and Exploratory Landscape Analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, pages 2036–2045, New York, NY, USA, 2022. Association for Computing Machinery

2) André Thomaser, Marc-Eric Vogt, Anna V. Kononova, and Thomas Bäck. Transfer of Multi-objectively Tuned CMA-ES Parameters to a Vehicle Dynamics Problem. In Michael Emmerich, André Deutz, Hao Wang, Anna V. Kononova, Boris Naujoks, Ke Li, Kaisa Miettinen, and Iryna Yevseyeva, editors, *Evolutionary Multi-Criterion Optimization*, pages 546–560, Cham, 2023. Springer Nature Switzerland

3) André Thomaser, Jacob de Nobel, Diederick Vermetten, Furong Ye, Thomas Bäck, and Anna V. Kononova. When to Be Discrete: Analyzing Algorithm Performance on Discretized Continuous Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, pages 856–863, New York, NY, USA, 2023. Association for Computing Machinery

4) André Thomaser, Marc-Eric Vogt, Thomas Bäck, and Anna V. Kononova. Real-World Optimization Benchmark from Vehicle Dynamics: Specification of Problems in 2D and Methodology for Transferring (Meta-)Optimized Algorithm Parameters. In *Proceedings of the 15th International Joint Conference on Computational Intelligence*, pages 31–40. SCITEPRESS - Science and Technology Publications, 2023 (best paper award)

5) André Thomaser, Marc-Eric Vogt, Thomas Bäck, and Anna V. Kononova. Optimizing CMA-ES with CMA-ES. In *Proceedings of the 15th International Joint Conference on Computational Intelligence*, pages 214–221. SCITEPRESS - Science and Technology Publications, 2023

Chapter 2

Preliminaries

This chapter provides a brief introduction to optimization and evolutionary algorithms. Furthermore, methods for analyzing the landscape of an optimization problem are presented, as well as a description of performance measurements and the benchmark problems used. Finally, techniques for quantifying and handling uncertainty in noisy optimization problems are discussed based on the overview given by [32].

2.1 Optimization

Optimization is employed in a wide field of scientific and technical disciplines, ranging from economics and finance to computer science and engineering. The quest for optimizing a system is based on the understanding that this system exhibits a specific performance that can be improved. To this end, a measurable objective function must be defined to compare the quality of different solutions. The main target, then, is to determine the most favorable state for a particular system. An optimization problem with only one objective is called a single-objective optimization problem.

Mathematically, solving a single-objective optimization problem is defined as the task of identifying a solution $\mathbf{x}^*$ within a set $\mathcal{X}$ of feasible solutions that minimizes the objective function $f(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}$ [76]:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}). \tag{2.1}$$

Hence, the objective function value of the global minimum satisfies:

$$f^* = f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall\, \mathbf{x} \in \mathcal{X}. \tag{2.2}$$

Without loss of generality, any maximization problem can be transformed into an equivalent minimization problem by multiplying the objective function by minus one:

$$\arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) = \arg\min_{\mathbf{x}\in\mathcal{X}} -f(\mathbf{x}). \tag{2.3}$$

The analytical form of an objective function of a real-world optimization problem is often unknown or difficult to access. In such cases, the objective function only provides an output without additional information when receiving an input. Such objective functions are referred to as black-box [8].

Furthermore, optimization problems are commonly classified as either continuous or discrete. For continuous problems, the feasible set is a subset of the real number domain $\mathcal{X} \subseteq \mathbb{R}^d$. For discrete problems, the feasible set is a subset of the integer domain $\mathcal{X} \subseteq \mathbb{Z}^d$ [8]. Problems involving a combination of both discrete and continuous variables fall under the category of mixed-integer optimization [37].

Black-box optimization problems are characterized by the lack of derivative information. Gradient-based methods are inapplicable. Therefore, solving black-box optimization problems requires using sampling-based heuristic algorithms. These heuristics are iterative approximation algorithms that explore the search space to locate near-optimal solutions without relying on the gradient of the objective function. A prominent variant among these heuristics are Evolutionary Algorithms (EAs).

2.2 Evolutionary Algorithms

EAs are a class of optimization algorithms inspired by biological evolution, where a population of individuals competes for limited resources, and only the fittest individuals survive. In general, the idea behind all EAs is to start with a set of candidate solutions that are randomly generated and evaluated using a fitness measure. The candidates with the highest fitness are selected to produce the next generation through recombination by combining parts of two or more parents to produce offspring and mutation by altering a single candidate to produce a new candidate. This fitness evaluation, selection and variation cycle continues until a satisfactory solution is found or a computational limit is reached. [31]

EAs are driven by two main forces: variation operators (recombination and mutation), which introduce diversity and novelty, and selection, which enhances the average solution quality. Through the iterative procedure (Algorithm 1), the average fitness of the population is maximized or minimized. Thereby, the fitness of a candidate solution is defined by a fitness function, which is essentially the objective function. Bäck et al. [13] provide an overview of the most important developments in the field of EAs in recent decades.

Algorithm 1 General scheme of EAs [31].

```
INITIALIZE population with random candidate solutions
EVALUATE fitness of each candidate
while termination condition not met do
    SELECT parents from the population
    RECOMBINE parents
    MUTATE resulting offspring
    EVALUATE fitness of the new candidates
    SELECT individuals for the next generation
end while
return best solution found
```

2.3 Evolution Strategies

Evolutionary Strategys (ESs) represent a subclass of EAs. Originating from the work of Ingo Rechenberg [106] and Hans-Paul Schwefel [115], ESs were initially designed to solve real-valued parameter optimization problems. Unlike other EAs that encode solutions as binary strings or other discrete structures, ESs operate directly on real-valued vectors.

The main components of ESs are similar to those of other EAs, with an emphasis in ESs on mutation and selection as the primary mechanisms for exploration and exploitation. Two key features of ESs are the use of a multivariate normal distribution for the mutation operator [12] and self-adaptive mechanisms for the mutation rate, which allows to dynamically adjust the mutation step size of the algorithm in response to the landscape of the fitness function [17]. A significant improvement in ESs is achieved with the introduction of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [56, 57].

2.4 Covariance Matrix Adaption Evolution Strategy

CMA-ES represents a breakthrough in the field of evolutionary computation and has emerged as a leading method for tackling continuous single-objective optimization problems [47]. The following description provides an overview of the basic principles of CMA-ES based on the tutorial by Hansen [50]. Figure 2.1 illustrates a CMA-ES run on the two-dimensional sphere function.

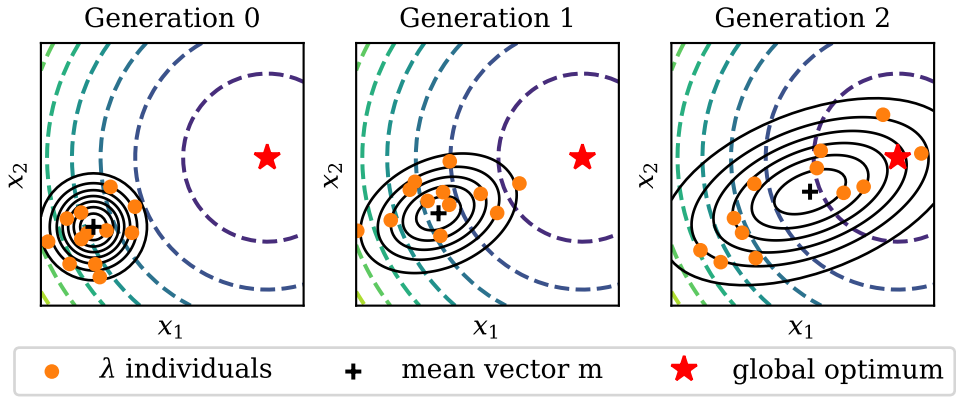

Figure 2.1: Visualization of a CMA-ES run with a population of 10 offspring on the two-dimensional sphere function for the first three generations.

In each generation g of CMA-ES, a new population of λ offspring is generated. The offspring $\mathbf{x}_{i:\lambda}^{(g)}$ are sampled from a multivariate normal distribution with a mean vector $\mathbf{m}^{(g)} \in \mathbb{R}^d$, a covariance matrix $\mathbf{C}^{(g)} \in \mathbb{R}^{d \times d}$ and a standard deviation $\sigma^{(g)} \in \mathbb{R}_{>0}$:

$$\mathbf{x}_k^{(g+1)} \sim \mathbf{m}^{(g)} + \sigma^{(g)} \mathcal{N}(0, \mathbf{C}^{(g)}) \quad \forall\, k = 1, ..., \lambda. \tag{2.4}$$

The motivation for such a mutation mechanism is to tailor the distribution of emerging populations to the local topography of the fitness landscape. Therefore, CMA-ES dynamically adjusts the mean vector $\mathbf{m}^{(g)}$, the covariance matrix $\mathbf{C}^{(g)}$ and the standard deviation $\sigma^{(g)}$. Initially, the top μ parents with the highest fitness within the population are selected ($\mu < \lambda$). Utilizing the specified weights w_i, the updated mean vector $\mathbf{m}^{(g+1)}$ is calculated as the weighted mean of these μ selected parents, and the learning rate c_m is usually set to 1:

$$\mathbf{m}^{(g+1)} = \mathbf{m}^{(g)} + c_\mathrm{m} \sum_{i=1}^{\mu} w_i (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}), \quad \sum_{i=1}^{\mu} w_i = 1. \tag{2.5}$$

The covariance matrix $\mathbf{C}$ is initially set to the identity matrix $\mathbf{I}$ and evolves in each generation through two key updates: the rank-μ update and the rank-one update. While the rank-μ update leverages information from the current population, the rank-one update is derived from the evolution path $\mathbf{p}_\mathrm{c} \in \mathbb{R}^d$. The evolution path, starting at $\mathbf{p}_\mathrm{c} = 0$, captures the trajectory of successful adaptations in the search space across several generations.

$$\mathbf{C}^{(g+1)} = (1 - c_1 - c_\mu)\mathbf{C}^{(g)} + c_1 \underbrace{\mathbf{p}_\mathrm{c}^{(g+1)}\mathbf{p}_\mathrm{c}^{(g+1)^T}}_{\text{rank-one update}} + c_\mu \underbrace{\sum_{i=1}^{\mu} w_i \, \mathbf{y}_{i:\lambda}^{(g+1)} \left(\mathbf{y}_{i:\lambda}^{(g+1)}\right)^T}_{\text{rank-}\mu\text{ update}}, \quad (2.6)$$

$$\mathbf{p}_\mathrm{c}^{(g+1)} = (1 - c_\mathrm{c})\mathbf{p}_\mathrm{c}^{(g)} + \sqrt{c_\mathrm{c}(2 - c_\mathrm{c})\mu_\mathrm{eff}} \, \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{c_\mathrm{m} \, \sigma^{(g)}}, \quad (2.7)$$

$$\mu_\mathrm{eff} = \left(\sum_{i=1}^{\mu} w_i^2 \right)^{-1}, \quad (2.8)$$

$$\mathbf{y}_{i:\lambda}^{(g+1)} = \frac{\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}}. \quad (2.9)$$

The standard deviation σ regulates the mutation step size. To independently control the step size from the covariance matrix update, the cumulative step length adaptation (CSA) is employed. CSA adjusts the standard deviation σ by utilizing the conjugate evolution path $\mathbf{p}_\sigma \in \mathbb{R}^d$. This evolution path $\mathbf{p}_\sigma$ captures the cumulative magnitude and direction of mutations over time. A short evolution path indicates that recent steps effectively cancel each other out, leading to a lack of directional progress. The step size should then be decreased. In contrast, a long evolution path indicates consistent steps in a particular direction, suggesting that increasing the step size could accelerate convergence. As a reference for the evolution path length, the expected length of a vector sampled from a standard multivariate normal distribution $E\,\|\mathcal{N}(0, \mathbf{I})\|$ is used. The CSA mechanism applies an exponential update to the standard deviation σ, moderated by a damping factor d_σ:

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

$$\mathbf{p}_\sigma^{(g+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_\mathrm{eff}} \, \mathbf{C}^{(g)^{-\frac{1}{2}}} \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{c_\mathrm{m} \, \sigma^{(g)}}. \quad (2.11)$$

The learning rates c_1, c_c, c_μ and c_σ govern the covariance matrix and the step size adaptation, thereby directly influencing the behavior of CMA-ES. The following mathematical expressions define the recommended default parameters:

$$c_c = \frac{4 + \frac{\mu_{eff}}{d}}{d + 4 + \frac{2 \cdot \mu_{eff}}{d}}, \tag{2.12}$$

$$c_1 = \frac{2}{(d + 1.3)^2 + \mu_{eff}}, \tag{2.13}$$

$$c_\mu = \min\left(1 - c_1, 2 \cdot \frac{\mu_{eff} - 2 + \frac{1}{\mu_{eff}}}{(d + 2)^2 + \mu_{eff}}\right), \tag{2.14}$$

$$c_\sigma = \frac{\mu_{eff} + 2}{d + \mu_{eff} + 5}. \tag{2.15}$$

The default number of offspring λ in a population varies with the dimensionality of the optimization problem d. Typically, the selection ratio μ_r, which represents the proportion of parents selected from the population, is set to 0.5:

$$\lambda = 4 + \lfloor 3 \cdot \ln(d) \rfloor, \tag{2.16}$$

$$\mu = \lfloor \mu_r \cdot \lambda \rfloor. \tag{2.17}$$

2.4.1 Variants

Several variants of CMA-ES have been introduced to enhance various aspects of the algorithm [12]. Below is a curated selection of the most impactful variants.

- **Active Update** [65]: Extends the original adjustment of the covariance matrix by including both the most and least successful candidate solutions. Negative weights are assigned to the latter to actively discourage the search from suboptimal regions of the search space. This promotes a more efficient exploration of the solution space.

- **Elitism** [131]: Modifies the selection mechanism of the standard (μ, λ)-CMA-ES, where the top μ offspring with the highest fitness within the population of λ offspring are selected as parents to produce the next generation. In contrast, in the $(\mu+\lambda)$-CMA-ES, both the μ parents and the λ offspring together compete for survival, and the top μ individuals are carried forward to the next generation. This ensures that the highest quality solution so far is always preserved.

- **Mirrored Sampling** [21]: Only half of the λ offspring of a new population are sampled from a multivariate normal distribution. The other half consists of the mirror images of the first set, obtained by negating the sampled vectors. This mirrored sampling technique is employed to improve the uniformity in the distribution of search points across the search space and potentially the exploration capabilities of CMA-ES.

- **Orthogonal Sampling** [134]: The vectors representing the offspring are orthonormalized by applying the Gram-Schmidt process [41, 113]. The goal is to prevent the overlap of search directions. Particularly in high-dimensional optimization problems, orthogonal sampling can lead to a more structured exploration of the search space by CMA-ES.

- **Weighted Recombination** [57]: Recombination in CMA-ES is accomplished by adjusting the mean vector $\mathbf{m}$ as the weighted sum of the top μ offspring. The weights w_i are determined by the following formula $w_i = \log(\mu+\frac{1}{2}) - \frac{\log(i)}{\sum_j w_i}$, where i is the rank of the offspring within the population. The logarithmic scaling of the offspring systematically favors the best offspring and also takes into account the contributions of the other offspring. This represents a compromise between exploitation and exploration. As an alternative to the logarithmic weights, equal weights $w_i = \frac{1}{\mu}$ can be assigned to all top μ offspring. Through this equal weighting, the diversity in the recombination step is increased.

- **Restart** [10]: If CMA-ES encounters stagnation or becomes stuck in a local optimum, a restart is performed. This strategy rejuvenates the search process by reinitializing the algorithm with a new population. Moreover, at each restart, the population size can be increased by a factor to enhance global search capabilities, an approach known as increasing population (IPOP) [9], or alternated between a smaller and a larger population to balance fine-tuned local search with broader exploration, known as bi-population (BIPOP) [48].

Employing a tailored configuration of these different variants that is carefully chosen to align with the unique characteristics and demands of the particular optimization task has the potential to significantly enhance the overall performance and robustness of CMA-ES [131, 132].

2.4.2 Box Constraint Handling

For numerous real-world optimization problems, the region of feasible solutions is restricted due to physical constraints or process dependencies. The most rudimentary form of such restrictions are box constraints, which restrict the feasible solution space $\mathcal{X} = [l, u]^d$ by a lower bound l and an upper bound u for each dimension, ensuring that $l \leq x_i \leq u$ for all x_i. Under these constraints, the feasible region $\mathcal{X}$ constitutes a hypercube, which is the origin of the term box constraint. In the general case of different lower and upper bound values per dimension, $\mathbf{l} \in \mathbb{R}^d$ and $\mathbf{u} \in \mathbb{R}^d$, $\mathcal{X}$ is a hyperrectangle. This hyperrectangle can be transformed into a hypercube by min-max normalization [98].

There are numerous methods available to prevent an optimization algorithm from generating infeasible solutions. The specific method for box constraint handling significantly influences the performance of CMA-ES [18]. In the following, the four repair methods, *projection*, *reflection*, *wrapping* and *reinitialization*, are described. Additionally, an explanation of the feasibility preserving method *resampling* is given. The terminology adopted in this thesis follows the convention established in the existing literature [5, 18, 137].

In general, repair methods are based on transforming an infeasible individual to a feasible one by a mapping process, denoted as $M : \mathbb{R}^d \to \mathcal{X}$, which is frequently applied in a coordinate-wise manner. This ensures that the solutions adhere to the predefined constraints of the problem space.

Projection enforces the constraints by clipping all out-of-bounds values to the closest bound of the feasible region:

$$T_i(x_i) = \begin{cases} x_i & l \leq x_i \leq u, \\ u & x_i > u, \\ l & x_i < l. \end{cases} \tag{2.18}$$

Reflection corrects infeasible values by mirroring them back into the feasible region based on the extent of the constraint violation:

$$T_i(x_i) = \begin{cases} x_i & l \leq x_i \leq u, \\ T_i(u + (u - x_i)) & x_i > u, \\ T_i(l + (l - x_i)) & x_i < l. \end{cases} \tag{2.19}$$

Wrapping treats the search space as a torus. Infeasible values are shifted by $(u - l)$:

$$T_i(x_i) = \begin{cases} x_i & l \leq x_i \leq u, \\ T_i(x_i - (u - l)) & x_i > u, \\ T_i(x_i - (u - l)) & x_i < l. \end{cases} \tag{2.20}$$

Reinitialization replaces the infeasible value by a value ξ sampled from a uniform distribution within the lower bound l and upper bound u of the feasible region:

$$T_i(x_i) = \begin{cases} x_i & l \leq x_i \leq u, \\ \xi & x_i < l \text{ or } x_i > u. \end{cases} \tag{2.21}$$

Resampling preserves the feasibility of individuals by repeating the mutation operator until a feasible individual is generated. To avoid the possibility of an infinite loop, a repair method should be employed after a predefined number of unsuccessful attempts.

In contrast to resampling, with a repair method, the optimization algorithm effectively perceives an altered landscape outside the feasible region since infeasible solutions are repaired before being evaluated (Figure 2.2).

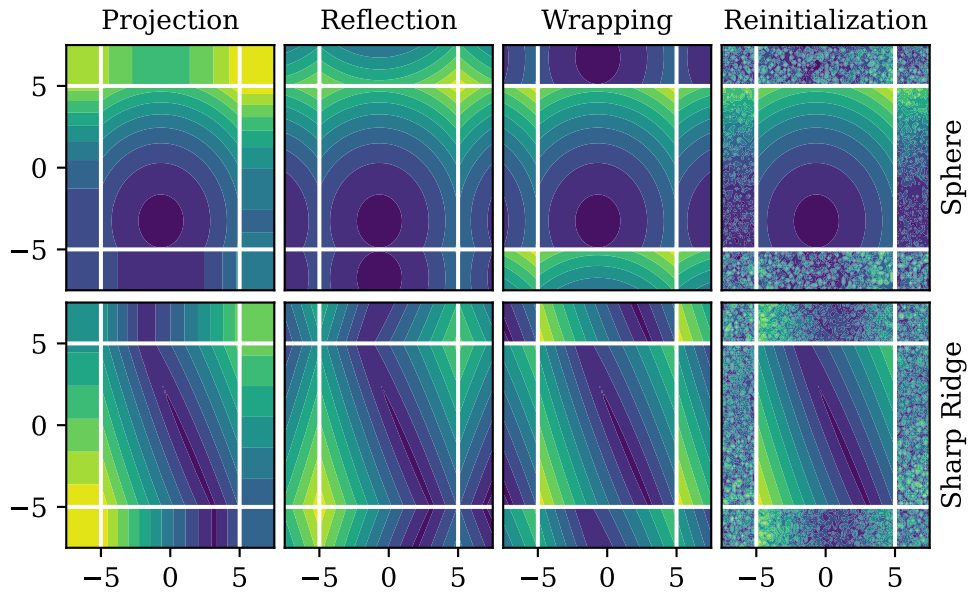

Figure 2.2: Landscape modification outside the feasible region $\mathcal{X} \in [-5, 5]^2$ for the two-dimensional sphere function and the sharp ridge function after the application of the four repair methods: projection, reflection, wrapping and reinitialization.

In a comprehensive comparison of different box constraint handling methods for CMA-ES, Biedrzycki [18] demonstrated that reflection and resampling consistently outperform other techniques across a wide range of problems and dimensionalities. However, resampling requires additional iterations within the algorithm in order to generate practicable solutions. This can lead to additional computational effort and, thus, to a reduction in overall efficiency. Furthermore, projection leads to a bias towards the boundaries. This is particularly beneficial if the optimum is actually located on the boundaries.

2.4.3 Integer Handling

CMA-ES is a highly effective method for continuous optimization problems [47]. Yet, many practical applications involve mixed-integer problems with both continuous and discrete variables. In such cases, discrete variables are constrained to specific values. A widely used method for applying CMA-ES in these scenarios is to discretize the search space for discrete variables by rounding the continuous values to the nearest feasible discrete value. This process creates a landscape where each discrete value is surrounded by a plateau of an identical fitness value. The objective function is discretized.

However, this discretization strategy through rounding introduces a potential pitfall for CMA-ES: stagnation of the optimization process. This is the case if the variance of the mutation distribution within the CMA-ES becomes smaller than the granularity of the discretization due to self-adaptation. In other words, the mutation steps generated by CMA-ES become too small to traverse the flat fitness landscape regions [49]. This can cause the optimization algorithm to become stuck on one of the plateaus introduced by the discretization, preventing the optimization algorithm to progress toward potentially better solutions.

To address the stagnation issue of CMA-ES in discretized optimization landscapes. Hamano et al. [46] propose an extension to CMA-ES, called the CMA-ES with Margin (CMA-ESwM). CMA-ESwM ensures that the marginal probabilities of generating individuals beyond the discretization step size are sufficiently high.

This is achieved by adding a diagonal matrix $\mathbf{A}$ to the mutation distribution. The mutation distribution is then represented as $\mathcal{N}(\mathbf{m}, \sigma^2\mathbf{ACA}^T)$. Both the matrix $\mathbf{A}$ and the mean vector $\mathbf{m}$ are then adaptively adjusted in each generation of CMA-ES to ensure that the probability of generating mutations beyond the discretization step size is at least greater than a margin α. As a default value for the margin, Hamano

et al. [46] recommend $\alpha = \frac{1}{\lambda\, d}$. The margin extension is an affine transformation of the mutation distribution. It is important to note that, when the margin α is set to zero, CMA-ESwM reverts to the original CMA-ES, since no correction is applied. Experiments conducted on the BBOB-mixint testbed [130] show that CMA-ESwM outperforms several other methods, especially in scenarios involving higher-dimensional problems [45].

2.5 Objective Function Landscape

In the field of optimization, the concept of a landscape provides a powerful analogy for visualizing the domain of possible solutions. Each location in this landscape corresponds to a candidate solution and the height at a specific location represents the value of the objective function for that solution. Within this landscape, depending on whether the goal is to maximize or minimize the objective function, optimization algorithms seek the highest peak or the deepest valley. Higher-dimensional landscapes require an abstraction of the idea of peaks and valleys into a multidimensional space in which each additional dimension represents a new variable.

An objective function landscape generally consists of three elements [122]: A set of potential solutions $X \subset \mathcal{X}$ to the problem, a notion of distance between solutions and an objective function $f(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}$. Such a representation provides a framework for discussing the characteristics of an objective function in terms of its local and global structures within the search space. Understanding the topology of the landscape is critical because it profoundly affects the performance and effectiveness of optimization algorithms.

An objective function landscape can be described by high-level properties (Section 2.5.1) and quantified with specific low-level features (Section 2.5.2). Some properties are derived theoretically others via sampling. Further information about the analysis of objective function landscapes can be found in the literature [86, 87, 101].

2.5.1 Objective Function Properties

In the context of continuous single-objective optimization, the objective function landscape can be described by high-level properties [52, 89]. The following list provides a concise overview of important properties, though it is not exhaustive:

- **Separability**: Indicates whether the objective function can be decomposed into lower-dimensional, independent subproblems, each containing only a subset of the variables.
- **Multimodality**: A multimodal landscape is characterized by several local optima. The presence of multiple peaks and valleys complicates the search for the global optimum, as algorithms risk becoming ensnared in suboptimal solutions.
- **Global Structure**: The overarching "architecture" of the landscape, including the distribution and arrangement of optima, shapes the global search strategy.
- **Variable Scaling**: Captures the anisotropic nature of the solution space, reflecting the magnitude of changes in the objective function value to alterations in distinct dimensions.
- **Conditioning**: The condition of a function refers to how the output value responds to small changes in input variables. Poorly conditioned functions, where output values vary wildly with small input variations, may have narrow and steep valleys.
- **Plateaus**: Regions with little to no change in the objective function value.
- **Noisiness**: Stochastic fluctuations in the landscape (Section 2.7).

2.5.2 Exploratory Landscape Analysis

Exploratory Landscape Analysis (ELA) [88] describes the high-level features of an objective function landscape with quantifiable metrics using mathematical and statistical techniques. Based on a sample of points evaluated with the objective function, ELA computes a set of low-level features that capture the nuanced properties of the landscape. The stochastic nature of creating the sample results in a variability in the feature values. Hence, the accuracy of these features depends on the sampling strategy and sample size. Renau et al. [107] recommend using Sobol' sequences and Kerschke et al. [71] recommend a sample size of at least 50 times the problem dimensionality.

A variety of ELA features have been developed and are accessible across various platforms, with the flacco package serving as a comprehensive repository for these features within a single framework [73]. Some of the key features considered are briefly described in the following:

- **Levelset** [88]: These features are computed using the mean cross-validated misclassification error of different classifiers.

- **y-Distribution** [88]: This group of features is based on the distribution skewness and kurtosis of the objective function values. Kurtosis assesses the peak value of the distribution and indicates whether it is relatively flat or sharp compared to a normal distribution. In addition, the number of peaks in the distribution provides information about the multimodality of the landscape.

- **Meta-Model** [88]: Linear and quadratic regression models, with or without interaction terms, are trained on the sample data. The features are derived from the resulting quality and the coefficients of these models.

- **Dispersion** [85]: This feature set compares the dispersion among the initial sample points with subsets that are defined by function value thresholds.

- **Nearest Better Clustering** [70, 103]: Analyzes the topology of the landscape by examining the distance ratios and correlations between the nearest neighbors of a point and those with better fitness, as well as the number of points to which the point is the nearest better neighbor.

- **Principal Component Analysis**: Insight is gained from a principal component analysis performed on the sampled points.

- **Information Content** [91]: Measures the smoothness, ruggedness and neutrality of the landscape through a random walk.

While this collection of features provides a comprehensive toolkit for landscape analysis, it is important to acknowledge the existence of additional features that are not within the scope of this study. Some are designed explicitly for low-dimensional spaces or require further objective function evaluations, which limits their applicability in specific scenarios. However, the large number of available features can also lead to redundant features [120].

Examples of applications of ELA are the analysis of similarities of problems across benchmark sets [120], the selection of an optimization algorithm [72] or the parameter tuning of an optimization algorithm [15]. Muñoz et al. [92] give an overview of the research field combining feature-based landscape analysis and algorithm selection for continuous black-box optimization problems.

2.6 Benchmarking and Tuning

The benchmarking of optimization algorithms involve the systematical assessment and comparison of the performance of optimization algorithms in solving standardized test functions or real-world optimization problems. Benchmarking reveals the strengths and weaknesses of different algorithms. The insights about the behavior of an optimization algorithm gained from benchmarking facilitate the selection of a specific optimization algorithm for a given optimization problem.

However, the benchmarking process must be carefully designed to provide meaningful results. Bartz-Beielstein et al. [14] emphasize the importance of clear objectives, precise problem definitions, appropriate algorithm selection, relevant performance metrics and thorough analysis.

2.6.1 Benchmark Problems

An integral part of benchmarking optimization algorithms is the selection of appropriate benchmark problems. A widely accepted benchmark suite for single objective continuous optimization problems is the Black-Box Optimization Benchmarking (BBOB) [54]. The BBOB suite consists of a collection of 24 noiseless test functions, each designed to evaluate different aspects of optimization algorithms.

Table 2.1 provides an overview of the 24 noiseless BBOB functions. These functions are typically categorized into five subsets based on categories ranging from simple separable landscapes to complex multi-modal terrains with weak global structures. This categorization enables a targeted assessment of the capabilities and limitations of an algorithm in different optimization scenarios.

A problem instance is a concrete example with defined parameters belonging to a broader problem class. Within the BBOB framework, problem instances are created by transforming the base functions. Examples of transformations are rotating and moving the input area. These transformations are defined by randomly generated parameters that remain consistent for each instance. In other words, the conditions remain unchanged for any algorithm that is re-evaluated on the same instance. However, a critical aspect of this process is the change in the location of the global optimum across different instances. As a result, the BBOB framework prevents algorithms that are biased [77] in the direction of the optimum, e.g. due to favorable starting positions, from gaining an unfair advantage.

Table 2.1: Overview of the 24 noiseless BBOB functions [54].

ID	Function Name	Description
1	Sphere Function	
2	Ellipsoidal Function, original	Separable unimodal
3	Rastrigin Function	functions with
4	Büche-Rastrigin Function	global structure
5	Linear Slope Function	
6	Attractive Sector Function	
7	Step Ellipsoidal Function	Functions with low
8	Rosenbrock Function, original	or moderate conditioning
9	Rosenbrock Function, rotated	
10	Ellipsoidal Function, rotated	
11	Discus Function	Functions with
12	Bent Cigar Function	high conditioning
13	Sharp Ridge Function	and unimodal
14	Different Powers Function	
15	Rastrigin Function	
16	Weierstrass Function	Multi-modal
17	Schaffers F7 Function	functions with
18	Schaffers F7 Function, moderately ill-conditioned	adequate global structure
19	Composite Griewank-Rosenbrock F8F2	
20	Schwefel Function	
21	Gallagher's Gaussian 101-me Peaks Function	Multi-modal functions
22	Gallagher's Gaussian 21-hi Peaks Function	with weak
23	Katsuura Function	global structure
24	Lunacek bi-Rastrigin Function	

2.6.2 Performance Metric

Benchmarking the performance of black-box optimization algorithms is generally based on the evaluation of two primary indicators: the quality of the solutions obtained and the costs expended. Thus, the performance of an optimization algorithm can be assessed from two perspectives: fixed-target or fixed-budget [14].

In the fixed-budget scenario, a limited budget of resources is given and the performance of the optimization algorithm is determined by the quality of the found solution. Thus, using a fixed budget approach reflects the cost constraints of real-world problems. In the fixed-target scenario, a specific target is defined. The performance is measured by the costs expended to reach the target. This provides easily understandable results, such as the relative speed of different algorithms in achieving a specific solution quality. A drawback of the fixed-target scenario is that a concept for handling runs that do not reach the predefined goal is required.

Algorithms can also be evaluated based on anytime performance, which considers the performance trajectory of the algorithm over time, as depicted in time quality diagrams. This metric summarizes the performance of an optimization algorithm along the complete optimization run in one value [66, 139].

The costs expended are typically quantified by the number of objective function evaluations or the total computation time. The choice between these two metrics depends on the relative computational overhead of the operations of the algorithm compared to the required time to evaluate candidate solutions. In scenarios where the evaluation time significantly surpasses the overhead of the algorithm, as often in real-world applications where, e.g., expensive simulations are involved, the number of objective function evaluations is the preferred measure.

Any performance measure must take into account not only the search cost but also the solution quality achieved. However, when dealing with stochastic optimization algorithms such as CMA-ES, multiple runs are essential to obtain an average quality score that ensures a robust evaluation of the algorithm's performance. The following describes three sophisticated performance metrics that are commonly used.

- **Success Rate (SR)**: Measures the proportion of independent runs of an algorithm that achieve a predefined solution quality within a given budget (such as function evaluations or time). The SR is particularly useful for assessing the robustness of an algorithm. A higher SR indicates that the predefined solution quality is reached more often across a defined number of repeated optimization runs.

- **Expected Running Time (ERT)** [10]: Returns for a given target the average number of function evaluations, also called average hitting time, needed to reach that target. If the success rate is less than 100%, a penalty is assigned based on the number of unsuccessful runs.

- **Area Under the Curve (AUC)** [53, 139]: The AUC of the Empirical Cumulative Distribution Function (ECDF) serves as an anytime performance metric that encapsulates the success of the algorithm over time. The ECDF tracks the proportion of targets within a predefined set T that have been achieved by the algorithm for each allocated budget b. Specifically, the ECDF value at budget b reflects the proportion of targets in T for which the best solution so far found by the algorithm is at least as good as the target. A higher AUC value means better performance. The algorithm is more likely to meet or exceed the targets across a range of budgets.

2.6.3 Parameter Tuning

The parameters of an optimization algorithm have a decisive influence on the behavior and, accordingly, the performance of the algorithm. For example, the learning rates and variants of CMA-ES (Section 2.4) can be tuned for specific functions or classes of functions [4, 131, 142]. However, manually tuning parameters can be a tedious and time-consuming process. In automatic parameter tuning, first proposed by [11, 42], this challenge is addressed by defining the parameter tuning task as a meta-optimization problem. Meta-optimization differs from the primary goal of solving the original optimization problem but complements it (Figure 2.3).

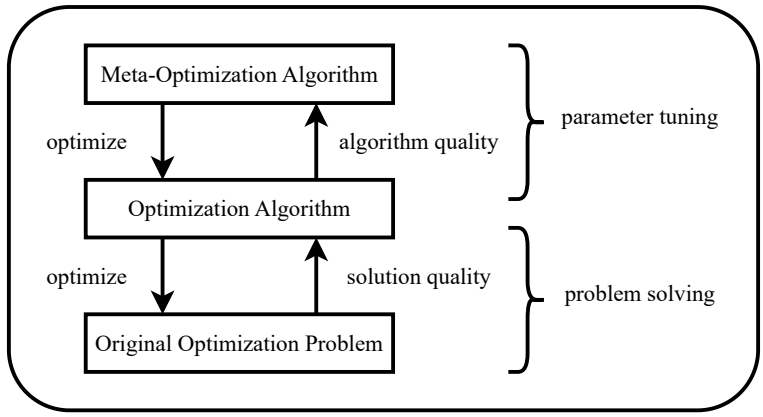

Figure 2.3: Schematic representation of the two optimization problems: (1) problem-solving, where an optimization algorithm seeks an optimal solution to the original optimization problem, and (2) parameter tuning, where a meta-optimization algorithm optimizes the parameters of the primary optimization algorithm to improve its performance [11, 30, 42].

Two optimization problems can be distinguished: solving the original problem and tuning the parameters of the optimization algorithm [30]. The former involves the optimization algorithm seeking an optimal solution to the given original optimization problem. The latter entails the use of a meta-optimization algorithm that optimizes the parameters of the optimization algorithm for solving the original problem. Tuning the parameters of an optimization algorithm is similar in concept to hyperparameter tuning in machine learning [138].

2.7 Uncertainty Quantification

Many real-world optimization problems exhibit noise or non-deterministic characteristics. This means that if exactly the same input parameter values are reevaluated, the objective function value varies. Reasons for this variability are often measurement errors, external disturbances or the inherent stochastic nature of the system under consideration. The following overview is based on [32].

Each evaluation of the objective function with the same input parameters $\mathbf{x}$ can be considered as a unique scenario z_i drawn from the set of all possible scenarios Ω_Z. In this context, an evaluation is not fixed to a particular scenario but depends on a random variable $Z \in \Omega_Z$. Thus, the objective function is then described as a random variable $f(\mathbf{x}, Z)$ as well. Therefore, the optimization task defined in Equation 2.1 requires reformulation. The revised objective is to find the solution $\mathbf{x}^*$ that minimizes the expected value E of the objective function across the randomness entailed by scenario Z:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{X}} \mathrm{E}_{Z \in \Omega_Z} \left[f(\mathbf{x}, Z) \right].^1 \tag{2.22}$$

Determining the true mean $\mathrm{E}_Z \left[f(\mathbf{x}, Z) \right]$ in closed form requires that the distribution of $f(\mathbf{x}, Z)$ be known and tractable. This is seldom possible for black-box optimization problems. Recent studies [1, 3, 68, 79, 105] have surveyed a spectrum of potential strategies to address this issue, highlighting the three principal approaches, which are described in the following:

- **Explicit Averaging**: The true mean $\mathrm{E}_Z \left[f(\mathbf{x}, Z) \right]$ is substituted with its sample estimate $\hat{\mathrm{E}}_Z \left[f(\mathbf{x}, Z) \right]$. This is achieved by resampling and computing the average objective function value across a number of K scenarios:

$$\mathrm{E}_Z \left[f(\mathbf{x}, Z) \right] \approx \hat{\mathrm{E}}_Z \left[f(\mathbf{x}, Z) \right] := \frac{1}{K} \sum_{i=1}^{K} f(\mathbf{x}, z_i), \tag{2.23}$$

  where z_i is sampled from the random variable distribution, i.e., $z_i \overset{s}{\sim} Z$. The sample estimates are denoted by the hat-$\hat{\square}$-symbol. If K is equal to the number of all possible scenarios $|\Omega_Z|$ and $z_1 \neq \ldots \neq z_i$ is ensured by sampling Without Replacement (WoR) from Ω_Z, the sample mean is equal to the true mean. Consequently, $\hat{\mathrm{E}}_Z \left[f(\mathbf{x}, Z) \right]$ can be used as a substitute without any associated uncertainty.

[1] To enhance readability, subsequent expressions will not include the set restrictions $\in \mathcal{X}$ and $\in \Omega_Z$.

- **Implicit Averaging**: Population-based methods, such as EAs, can leverage the collective data of all individuals in the population to estimate the true mean implicitly. Each individual is evaluated only once and the aggregated results of these evaluations guide the evolutionary process toward more favorable solutions. By using such an approach, the need for explicit averaging is avoided [6, 7, 36].

- **Surrogate Assistance**: A surrogate model is utilized to approximate the true $\mathrm{E}_Z[f(\mathbf{x}, Z)]$. This surrogate is subsequently employed for optimization purposes.

2.7.1 Static and Dynamic Allocation

The calculation of the sample estimate $\hat{\mathrm{E}}_Z[f(\mathbf{x}_k, Z)]$ for each individual $\mathbf{x}_k$ within the population requires the specification of the number of scenarios K_k to be evaluated (Equation 2.23). Choosing K_k less than the number of all possible scenarios $|\Omega_Z|$ decreases the number of evaluations but increases the uncertainty in the estimation of the true mean. Two principal schemes can be distinguished, the Static Allocation (SA) and the Dynamic Allocation (DA) of evaluations to each individual in the population [105]. The initial research on explicit averaging employed SA schemes that equally distribute the available evaluation budget N among the λ individuals:

$$\sum_{k=1}^{\lambda} K_k = N \quad , \quad K_1 = \cdots = K_\lambda. \tag{2.24}$$

The variance of the sample mean estimate can serve as a measure of uncertainty and according to estimation theory, the following applies [58, 109]:

$$\mathrm{Var}_Z\left[\hat{E}_Z[f(\mathbf{x}_k, Z)]\right] \quad \sim \quad \frac{\mathrm{Var}_Z[f(\mathbf{x}_k, Z)]}{K_k}. \tag{2.25}$$

From this, two conclusions can be drawn. First, the uncertainty in the sample mean estimate decreases as the number of scenarios K_k increases. Second, the uncertainty in the sample mean estimate is directly proportional to the true variance of the underlying distribution of $f(\mathbf{x}_k, Z)$. Therefore, selecting K_k proportional to $\mathrm{Var}_Z[f(\mathbf{x}_k, Z)]$ is more efficient for achieving equally uncertain sample mean estimates across individuals. DA schemes iteratively and adaptively determine the value of K_k for each individual $\mathbf{x}_k$ separately.

2.7.2 Dynamic Allocation for Ranking and Selection

During the selection phase of EAs, the individuals of a population are typically ranked on the basis of their objective function values. Therefore, a consensus of researchers suggests that the evaluation budget should be dynamically allocated among individuals [22, 23, 43, 55, 58]. The aim of this DA is to reduce uncertainty in the ranking process, especially in identifying the top-μ performing individuals, known as top-μ selection. Reducing the uncertainty in the ranking or in identifying the top-μ individuals does not necessarily mean that the uncertainty in the mean performance estimate of each individual is also reduced.

DA schemes for the identification of the top-μ performing individuals are part of the broader domain of Ranking and Selection (RaS). This area has seen the development of many RaS methods, which have been comprehensively reviewed by various authors [20, 28, 29, 38, 40, 59, 61, 93, 94, 99, 140, 141]. The methods for RaS can essentially be formulated as Bayesian or frequentist inference. Bayesian methods rely on a parametric and functional characterization of the probability distribution of $f(\mathbf{x}, Z)$ across both different individuals and scenarios. In other words, Bayesian approaches treat the parameter to be estimated as a random variable with a prior distribution, whereas frequentist approaches treat the parameter to be estimated as a fixed and deterministic quantity [112]. Some methods for RaS are described in the following.

Jiang et al. [67] select the top-μ individuals with a minimal number of evaluations by assuming sub-Gaussian distributions of $f(\mathbf{x}, Z)$. With a user-defined confidence, the lower and upper confidence bounds for each individual are derived to allocate evaluations to the individuals further. However, the method may fail if the actual distributions differ from a sub-Gaussian distribution. Chen et al. [24] propose a selection method that works with bounded objective values without assuming a distribution of $f(\mathbf{x}, Z)$. However, both methods lack an intermediate uncertainty quantification for stopping the allocation procedure before the user-defined confidence is met.

Hansen et al. [55] propose a method for assessing uncertainty in the current top-μ selection by observing the number of rank changes among the individuals of the current population after additional evaluations. If the uncertainty exceeds a certain threshold, more evaluations are assigned. However, this method is not based on statistical rank estimation theory. Groves et al. [43] present a Bayesian counterpart to the approach by Hansen et al., using Bayesian probabilities of rank changes to quantify uncertainty.

2.7.3 Confidence Interval Sequences

If the functional parametric form of the distribution of the random variable $f(\mathbf{x}, Z)$ is unknown or cannot be assumed, statistical estimation theory often resorts to non-parametric or distribution-free estimators and Confidence Intervals (CIs). However, traditional CIs are designed for static, one-time uncertainty quantification with a fixed number of evaluations, as is the case in SA. When the number of evaluations is not predetermined and additional evaluations are allocated adaptively, the repeated use of CIs for uncertainty quantification can inflate error rates [44, 111, 117, 118, 135]. Therefore, Confidence Interval Sequences (CISs) provide a more appropriate alternative to traditional CIs. Unlike static CIs, CISs provide a quantification of uncertainty over a series of intervals. This is particularly valuable in DA schemes as CISs allow repeated uncertainty quantification without the risk of statistical error inflation. Compared to traditional CIs, the use of CISs is preferable in situations where adaptability is required and the number of evaluations is not predetermined.

Several nonparametric, distribution-free CISs have been developed recently [60, 80, 95, 111, 116, 136]. Howard and Ramdas [60] proposed a method for the construction of CISs for quantiles and not for the mean of bounded random variables by applying sequential hypothesis tests. Kuchibhotla and Zheng [80] employ more powerful concentration inequalities to obtain CISs for the mean. In contrast, based on similar assumptions [95, 111] adopt a game-theoretic perspective on stochastic processes to construct comparably tight CISs. The CISs developed by Waudby-Smith and Ramdas [136] is considered state-of-the-art and provides the tightest CISs when sampling Without Replacement (WoR) is possible.

A CIS is defined as a sequence $\left\{CI_K\right\}_{K=1}^{|\Omega_Z|}$ of CIs for which

$$\mathbb{P}\Big[\mathrm{E}_Z\left[f(\mathbf{x}, z_i)\right] \in CI_K \quad \forall\, K = 1, \ldots, |\Omega_Z|\Big] \geq 1 - \alpha, \tag{2.26}$$

and where $\alpha \in [0, 1]$ denotes the user-defined CIS significance level. Smaller significance levels correspond to a higher probability that the CIS covers the true mean, which tends to lead to wider CIs. Figure 2.4 shows the CISs for two different significance levels. As more and more samples are observed (i.e., the larger K becomes), the CIs become tighter and tighter around the sample mean. Once all $|\Omega_Z|$ scenarios have been observed, the sample mean converged to the true mean.

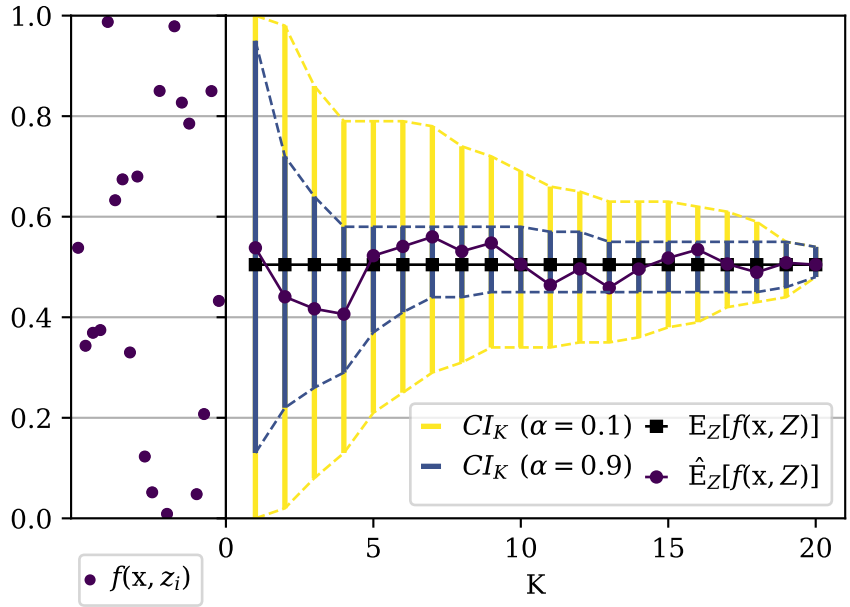

Figure 2.4: Left: The $|\Omega_Z| = 20$ uniformly distributed values $\{f(\mathbf{x}, z_1), \ldots, f(\mathbf{x}, z_{|\Omega_Z|})\}$. Right: Illustration of two CISs, each consisting of a CI after $K = 1, \ldots, 20$ evaluations for the significance levels $\alpha \in \{0.1, 0.9\}$. Each CI_K is constructed with the K sampled values available after K evaluations. In addition, the true mean $\mathrm{E}_Z[f(\mathbf{x}, Z)]$ and the estimated (sample) mean $\hat{\mathrm{E}}_Z[f(\mathbf{x}, Z)]$ are displayed along with the number of sampled values K.

Probabilistic Confidence Interval Sequence

Waudby-Smith and Ramdas [136] introduced a novel method to obtain a CIS for the true mean of a random variable. The approach is rooted in a game-theoretic framework and assumes that the sampled values from the random variable all lie within the interval $[0, 1]$. The underlying concept of this method is based on the concept of betting.

A multi-round betting game is set up for each $\eta \in [0, 1]$ being the mean. The games are played in parallel. In each round of a game, a bet on the upcoming observation can be placed. No wealth is gained or lost if the true mean equals η. But if there is a discrepancy between η and the true mean, smart betting can result in a financial gain. The betting strategy can be adaptive and may vary across the different games. The construction of the $(1 - \alpha)$ confidence set after K bets is determined by the ensemble obtained from the η values for which the wealth has not exceeded the threshold of $1/\alpha$. The true mean of the random variable is then in this set with high probability [136].

The method introduced by Waudby-Smith and Ramdas [136] represents a significant advancement in the construction of confidence intervals. This method generalizes the traditional approach in two key dimensions.

First, the method introduced employs nonparametric (composite) tests that do not rely on assumptions about the underlying distribution of the data. Thus, the approach is applicable to a larger variety of data distributions.

Second, to address the DA of samples and to overcome the limitation of CIs to a predetermined sample size, test (super)martingales are used. In the game-theoretic viewpoint, these are referred to as hedged capital process CP (i.e., the accumulated wealth from sequential betting) for each $\eta \in [0,1]$. Different betting schemes from game theory can be employed.

When the objective values $f(\mathbf{x}, z_i)$ are sampled WoR from the finite and non-random set $\left\{f(\mathbf{x}, z_1), \ldots, f\left(\mathbf{x}, z_{|\Omega_Z|}\right)\right\}$ through sampling scenarios z_i WoR from Ω_Z, only the order of observed objectives is uncertain. The WoR hedged capital process CP after K evaluation is defined as (Sections 5.2 and 5.3 of [136]):

$$\begin{aligned} CP_K^{WoR}(\eta) := \max\Bigg\{ & \theta \prod_{i=1}^{K}\left(1 + \lambda_i^+(\eta)\cdot\left(f(\mathbf{x}, z_i) - \eta_i^{WoR}\right)\right), \\ & (1-\theta)\prod_{i=1}^{K}\left(1 - \lambda_i^-(\eta)\cdot\left(f(\mathbf{x}, z_i) - \eta_i^{WoR}\right)\right)\Bigg\}, \\ & \text{with } CP_0^{WoR}(\eta) := 1 \\ & \text{and } \eta_K^{WoR} := \frac{|\Omega_Z|\cdot\eta - \sum_{i=1}^{K-1} f(\mathbf{x}, z_i)}{|\Omega_Z| - (K-1)}, \end{aligned} \tag{2.27}$$

where the parameter θ is dividing one's capital into two proportions: θ and $1-\theta$.

In the following, the construction of the CIS for $\mathrm{E}_Z[f(\mathbf{x}, Z)]$ when the scenarios z_i are sampled WoR from Ω_Z is described (Theorem 4 in [136]). When the real-valued predictable sequences $\left(\dot{\lambda}_K^+\right)_{K=1}^{\infty}$ and $\left(\dot{\lambda}_K^-\right)_{K=1}^{\infty}$ are not depending on η and with $c \in [0, 1[$ for each $K \geq 1$ applies:

$$\lambda_K^+(\eta) := \min\left(\left|\dot{\lambda}_K^+\right|, \frac{c}{\eta_K^{WoR}}\right), \ \lambda_K^-(\eta) := \min\left(\left|\dot{\lambda}_K^-\right|, \frac{c}{1-\eta_K^{WoR}}\right), \tag{2.28}$$

then

$$\mathrm{p}CI_K^{WoR} := \left\{\eta \in [0,1] : CP_K^{WoR}(\eta) < 1/\alpha\right\} \tag{2.29}$$

forms a $(1-\alpha)$-CIS for $\mathrm{E}_Z[f(\mathbf{x}, Z)]$. $\mathrm{p}CI_K^{WoR}$ is a CI of the true mean for each $K \geq 1$.

Waudby-Smith et al. [136] recommend to set the truncation level $c := 1/2$ and $\dot{\lambda}_K^+ = \dot{\lambda}_K^- = \lambda_K$ with

$$\lambda_: = \sqrt{\frac{2\log(2/\alpha)}{\hat{\sigma}_{K-1}^2 K \log(K+1)}}, \tag{2.30}$$

where $\hat{\sigma}_K^2$ and $\hat{\mu}_K$ can be interpreted as variance and mean estimators of $f(\mathbf{x}, z_i)$, respectively:

$$\hat{\sigma}_K^2 := \frac{1/4 + \sum_{i=1}^K \left(f(\mathbf{x}, z_i)) - \hat{\mu}_i\right)^2}{K+1}, \tag{2.31}$$

$$\hat{\mu}_K := \frac{1/2 + \sum_{i=1}^K f(\mathbf{x}, z_i)}{K+1}. \tag{2.32}$$

A closed-form computation of the CIS from Equation (2.29) is not feasible, given the infinite set of potential values $\eta \in [0,1]$. Therefore, in practice, $\mathrm{p}CI_K^{WoR}$ is determined through a grid search on $\eta \in \left\{0, \frac{1}{\eta_{\text{breaks}}}, \frac{2}{\eta_{\text{breaks}}}, \ldots, 1\right\}$ with, for example, $\eta_{\text{breaks}} = 100$.

Logical Confidence Interval Sequence

In addition to the previously described CIS based on probabilistic reasoning, Shekhar et al. [116] proposed a method for constructing CIS based purely on logical reasoning when sampling WoR. After evaluating K samples, $(|\Omega_Z| - K - 1)$ samples remain. Under the assumption that all values fall within the interval $[0,1]$, the following CIS

$$\mathrm{l}CI_K^{WoR} := \left[\frac{1}{|\Omega_Z|}\sum_{i=1}^K f(\mathbf{x}, z_i)\,,\ \frac{1}{|\Omega_Z|}\sum_{i=1}^K f(\mathbf{x}, z_i) + \frac{1}{|\Omega_Z|}\cdot(|\Omega_Z| - K - 1)\right] \tag{2.33}$$

for $\mathrm{E}_Z\left[f(\mathbf{x}, Z)\right]$ satisfies

$$\mathbb{P}\Big[\mathrm{E}_Z\left[f(\mathbf{x}, z_i)\right] \in \mathrm{l}CI_K^{WoR} \quad \forall\, K = 1, \ldots, |\Omega_Z|\Big] = 1. \tag{2.34}$$

The logical CIS from Equation (2.33) can be intersected with the probabilistic CIS from Equation (2.29) to produce valid, yet even tighter CIS [116]. Figure 2.5 shows the CIS with $|\Omega_Z| = 20$. Although the probabilistic CIS provides considerably narrower confidence intervals for a significant portion of the sequential sampling process, the logical CIS provides tighter intervals at the beginning for $K = 1$ and in the later phases from $K = 16$ to $K = 20$.

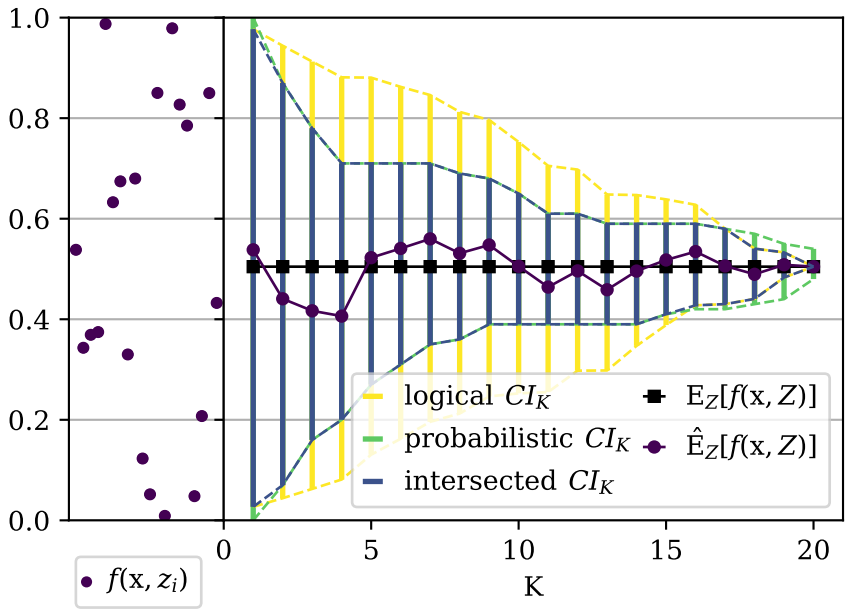

Figure 2.5: Left: The $|\Omega_Z| = 20$ uniformly distributed values $\{f(\mathbf{x}, z_1), \ldots, f(\mathbf{x}, z_{|\Omega_Z|})\}$. Right: Illustration of the resulting probabilistic CIS (Equation 2.29) and logical CIS (Equation 2.33), as well as the intersection, which is also a valid CIS. Each CI_K is constructed with the K sampled values available after K evaluations. The significance level α for the probabilistic CIS is set to 0.3. The true mean $\mathrm{E}_Z[f(\mathbf{x}, Z)]$ and the estimated (sample) mean $\hat{\mathrm{E}}_Z[f(\mathbf{x}, Z)]$ are displayed along with the number of sampled values K.

2.7.4 Uncertainty Quantification in Ranking and Selection

To effectively reduce the uncertainty in the ranking or in the selection of the top-μ individuals within a population of λ individuals, establishing a dependable quantitative measure of uncertainty is imperative. A prevalent approach for quantifying certainty in both ranking and selection involves calculating the probability that a given ranking or selection is accurate. This approach, therefore, gives rise to two central terms in the field of RaS: the Probability of Correct Ranking (PCR) and the Probability of Correct Selection (PCS) [20, 59, 99].

When given a finite set of individuals $\{\mathbf{x}_1, \ldots, \mathbf{x}_\lambda\}$, where each individual $\mathbf{x}_k$ is associated with an objective random variable $f(\mathbf{x}_k, Z)$ and the true mean objective value of each individual can be distinguished ($\mathrm{E}_Z[f(\mathbf{x}_1, Z)] \neq \ldots \neq \mathrm{E}_Z[f(\mathbf{x}_\lambda, Z)]$), then the true rank $\mathrm{rank}(\mathbf{x}_k)$ and the estimated rank $\hat{\mathrm{rank}}(\mathbf{x}_k)$ of an individual $\mathbf{x}_k$ within the population of λ individuals can be formally defined as:

$$\text{rank}(\mathbf{x}_k) := \left| \left\{ \mathbf{x}_j, j = 1, \ldots, \lambda \quad \middle| \quad \mathrm{E}_Z \left[f(\mathbf{x}_j, Z) \right] \leq \mathrm{E}_Z \left[f(\mathbf{x}_k, Z) \right] \right\} \right|, \tag{2.35}$$

$$\hat{\text{rank}}_{\text{frq}}(\mathbf{x}_k) := \left| \left\{ \mathbf{x}_j, j = 1, \ldots, \lambda \quad \middle| \quad \hat{\mathrm{E}}_Z \left[f(\mathbf{x}_j, Z) \right] \leq \hat{\mathrm{E}}_Z \left[f(\mathbf{x}_k, Z) \right] \right\} \right|. \tag{2.36}$$

Using the definition of the true rank and the estimated rank, in the frequentist setting the PCR and PCS can be defined as:

$$\text{PCR}_{\text{frq}} := \mathbb{P}\left[\text{rank}(\mathbf{x}_k) = \hat{\text{rank}}_{\text{frq}}(\mathbf{x}_k) \quad \forall i = 1, \ldots, \lambda \right], \tag{2.37}$$

$$\begin{aligned} \text{PCS}_{\text{frq}} := \mathbb{P}\Big[\Big\{ j = 1, \ldots, \lambda \,\Big|\, \text{rank}(\mathbf{x}_j) \leq \mu \Big\} \; = \\ \Big\{ j = 1, \ldots, \lambda \,\Big|\, \hat{\text{rank}}_{\text{frq}}(\mathbf{x}_j) \leq \mu \Big\} \Big]. \end{aligned} \tag{2.38}$$

In the frequentist setting, the true mean value is considered a fixed but unknown quantity and can be estimated through the sample mean (Equation 2.23). Consequently, within the frequentist setting, the probabilities PCR_{frq} and PCR_{frq} cannot be computed as a direct measure of the certainty in ranking and top-μ selection for a given dataset.

Klein et al. [74] propose a simple and effective method to construct Rank Intervals (RIs). Based on the CIs of the individuals, the RI RI_k of an individual k within a population of λ individuals is defined as follows:

$$\begin{aligned} & RI_k := \{|\Lambda_{Lk}| + 1,\ |\Lambda_{Lk}| + 2, \ldots,\ |\Lambda_{Lk}| + |\Lambda_{Ok}| + 1\}, \\ \text{with:} \quad & I_k := \{1, \ldots, \lambda\} \setminus \{i\}, \\ & \Lambda_{Lk} := \{j \in I_k \mid \max(CI_j) \leq \min(CI_k)\}, \\ & \Lambda_{Rk} := \{j \in I_k \mid \max(CI_k) \leq \min(CI_j)\}, \\ & \Lambda_{Ok} := I_k \setminus (\Lambda_{Lk} \cup \Lambda_{Rk}). \end{aligned} \tag{2.39}$$

The set Λ_{Lk} contains the indices of all the individuals that are clearly left of (i.e., smaller than) individual k, while the set Λ_{Rk} holds all the indices of individuals (other than k) that are clearly right of (i.e., larger than) individual i. Λ_{Li}, Λ_{Ri} and Λ_{Oi} are mutually exclusive: $\Lambda_{Lk} \dot{\cup} \Lambda_{Rk} \dot{\cup} \Lambda_{Ok} = I_k$. Thus, Λ_{Ok} contains the indices of individuals (except k) whose CI overlaps with the CI of individual k.

Rising [108] proposes two uncertainty measures: The Uncertainty Quantification in Ranking (UQiR) and the Uncertainty Quantification in Selection (UQiS) within a population of λ individuals:

$$UQiR_\lambda := \frac{1}{\lambda} \sum_{\mu=1}^{\lambda} exc_\mu, \tag{2.40}$$

$$UQiS_\mu := exc_\mu, \tag{2.41}$$

where exc_μ denotes the number of individuals minus μ that are among the top-μ individuals of the current population, based on the sampled objective values and the subsequently constructed CIs:

$$exc_\mu := \left| \left\{ k \in \{1, \dots, \lambda\} \mid |\Lambda_{Lk}| + 1 \leq \mu \right\} \right| - \mu. \tag{2.42}$$

Figures 2.6 illustrates the RIs based on exemplarily given CIs of six individuals. The calculated UQiS of the top three individuals $UQiS_3$ equals two because according to the RIs, five individuals could be among the top three individuals.

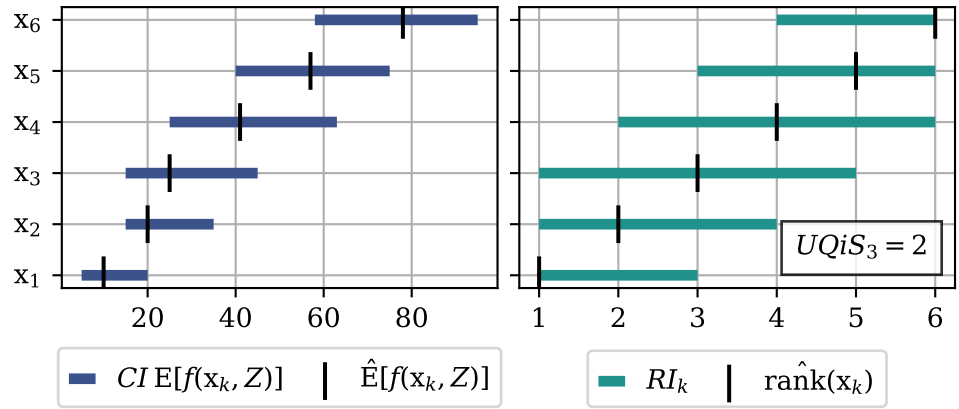

Figure 2.6: *Left*: CI for the true mean and sample mean for each individual in $\{x_1, \dots, x_6\}$. *Right*: Resulting RIs and sample ranks. The UQiS of the top-3 individuals $UQiS_3$ equals 2.

Chapter 3

Engineering Problems

Vehicle dynamics control systems (VDCSs) have become a cornerstone in the automotive industry. These systems use advanced control algorithms that, in conjunction with a range of sensors and actuators, dynamically modulate the vehicle's response, taking into account different driving conditions [104]. Therefore, VDCSs significantly improve both the safety and driving dynamics of modern vehicles providing enhanced driving pleasure.

The basis for modern VDCSs was laid by the development of the Antilock Braking System (ABS) [75] and the Electronic Stability Control (ESC) [78]. The introduction of these two systems in road vehicles has been shown to significantly improve braking performance and reduce the number of traffic accidents and fatalities by mitigating skidding and loss of control [25, 33, 34, 81].

The behavior of VDCSs depends on the precise calibration of system parameters designed to achieve optimal performance. Recent advances in simulation technology, coupled with the exponential growth in computational resources, have paved the way for the virtual pre-design of these parameters. To facilitate this virtual pre-design process, objective characteristic values (CVs) are required for assessing the performance of system parameters. These CVs are integral to the formulation of an objective function for determining the optimal parameters using an optimization algorithm.

In Sections 3.1 and 3.2, two such objective functions are defined mathematically for two different VDCSs. Furthermore, a brief overview of vehicle dynamics simulation and modeling is given (Section 3.3). Finally, a dataset created for benchmarking and algorithm design throughout this thesis is described (Section 3.4).

3.1 Antilock Braking System

The ABS is designed to prevent wheels from locking and to maximize the brake forces exerted by the tires during braking by adjusting brake pressure to keep brake slip within an optimal range. This reduces the braking distance. Moreover, the driver maintains control and can steer the vehicle even in an emergency braking situation. The brake slip s is the amount by which the wheel's circumferential speed v_{wheel} is behind the vehicle's linear speed (road speed) v_{vehicle} [75]:

$$s = \frac{v_{\text{vehicle}} - v_{\text{wheel}}}{v_{\text{vehicle}}} \cdot 100\%. \tag{3.1}$$

The longitudinal brake force that can be transmitted is proportional to the coefficient of friction μ_{x}. Figure 3.1 illustrates the relationship between the coefficient of friction and brake slip during straight-line braking with ABS for various road conditions. The ranges in which the ABS keeps the brake slip are shaded blue. The curves for dry, wet and ice demonstrate that ABS can significantly reduce braking distances compared to scenarios where the wheels lock up ($s = 100\%$). Snow leads to a unique situation where a wedge of snow accumulates in front of locked wheels, aiding in deceleration. In this case, the benefit of ABS lies in preserving steerability.

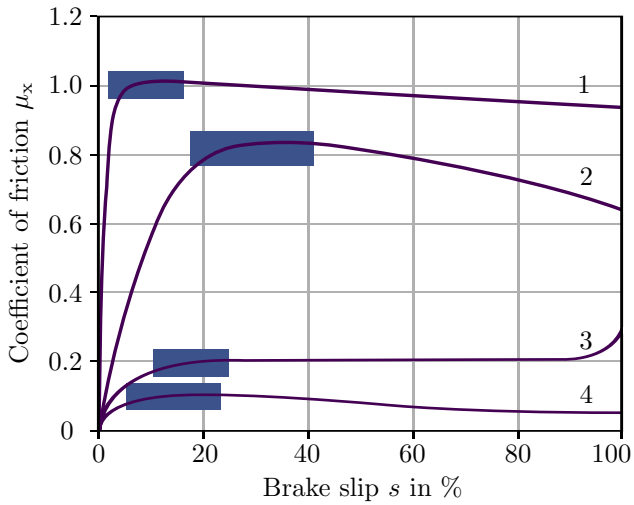

Figure 3.1: Relationship between the coefficient of friction μ_{x} and brake slip s during straight-line braking for various road conditions (1: dry, 2: wet, 3: snow, 4: ice). ABS keeps the brake slip in the shaded blue ranges. Figure is adapted from [75].

A standard maneuver for assessing a vehicle's braking performance is the emergency straight-line full-stop braking maneuver with ABS fully engaged [63]. A braking maneuver is defined, consisting of the following three phases (Figure 3.2):

(1) Acceleration of the vehicle to a maximum velocity of 103.5 km/h,
(2) Coasting the vehicle in neutral without accelerating or braking to 103 km/h,
(3) Braking of the vehicle with maximum deceleration until the vehicle stops.

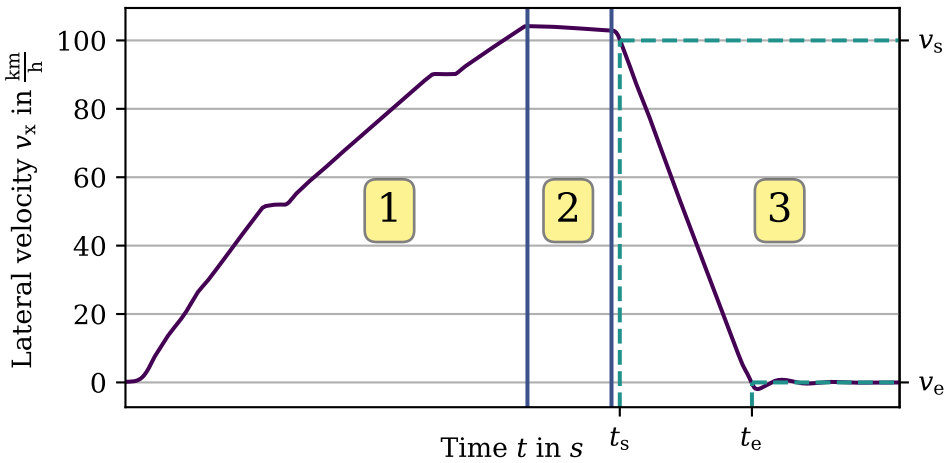

Figure 3.2: Illustration of the lateral velocity v_x of a vehicle over the three defined phases (1: acceleration, 2: coasting, 3: braking) of an emergency straight-line full-stop braking maneuver for calculating the braking distance from the start velocity $v_s = 100$ km/h at start time t_s to the end velocity $v_e = 0$ km/h at end time t_e.

The braking distance y is a CV for the ABS performance and is defined as the integral of the vehicle's longitudinal velocity v_x over time from the start velocity $v_s = 100$ km/h at time t_s to the end velocity $v_e = 0$ km/h at time t_e:

$$y = \int_{t_s}^{t_e} v_x(t)\, dt. \tag{3.2}$$

The objective is to find an optimal parameter configuration $\mathbf{x}^*$ for the d ABS parameters within the feasible input space $\mathbf{x} \in \mathbb{D}^d \subset \mathbb{R}^d$ that minimize the braking distance $y(\mathbf{x})$, as defined in Equation (3.2):

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{D}^d} y(\mathbf{x}). \tag{3.3}$$

3.2 Active Rollover Protection

At the limit of driving dynamics, the imminent risk of a vehicle rollover is characterized by one or more wheels lifting off the ground. High lateral force build-up can cause vehicles to rollover. In general, rollovers can be either tripped or untripped [104]. Tripped rollovers occur due to the influence of an external lateral force applied to the vehicle. For example, when the vehicle hits a curb. Untripped rollovers, on the other hand, occur as a result of sharp steering, such as when cornering at high speed or making a quick lane change. The resulting lateral forces on the tires cause the vehicle to roll over. In these scenarios, the Active Rollover Protection (ARP) as part of the ESC can intervene to prevent a vehicle rollover by stabilizing the vehicle through selective wheel braking and a reduction of engine torque.

Standardized maneuvers are employed to assess the driving behavior and the effectiveness of controller interventions, such as from the ARP. The Sine with dwell (SWD) [64] is one such maneuver. During the SWD maneuver, the vehicle initially drives at a constant speed of 80 ± 2 km/h in a straight line. A steering machine then imposes a sinusoidal steering input at a frequency of 0.7 Hz, incorporating a dwell period of 500 ms at the peak of the second half-wave. The amplitude of the steering wheel angle is set to a predetermined multiple of the characteristic steering wheel angle $\delta_{0.3g}$, which is ascertained from previous slowly increasing steer tests. These preliminary tests are designed to establish the vehicle's characteristic steering response at a lateral acceleration of 0.3g, providing a baseline for the SWD maneuver.

The induced steering angle causes a pronounced oversteer response in the vehicle, which can be critical to rollover, especially in vehicles with a high center of gravity. To ensure stability, several criteria must be satisfied. At no point should two wheels simultaneously lift more than 5 cm off the ground [19]. Additionally, the yaw rate should decrease to a specified fraction of its peak value within a certain time frame after the steering angle reverses direction [64]. Besides these stability criteria, an agility criterion is specified: the lateral displacement of the vehicle's center of gravity from its original path during straight-ahead driving must surpass a defined threshold [64].

Dourson [27] demonstrated through simulations that the parameters of ARP, which maximize the velocity of the vehicle one second after the SWD maneuver, also meet the specified stability and agility criteria. However, it should be noted that the used simulation model had limitations that prevented the modeling of a road edge contact and tire separation.

3.3 Vehicle Dynamics Modeling and Simulation

Various vehicle models with different complexity levels have been developed to simulate vehicle dynamics accurately. These vehicle models range from single-track to twin-track and extend to sophisticated multibody system models, each with their respective stages of expansion [90, 114]. With a higher model complexity, the required computing resources increase. Consequently, the choice of a vehicle model is guided by the principle that it should be sufficiently but not excessively detailed for the respective application.

A twin-track model implemented in MATLAB/Simulink [123] is employed for the simulation of a full vehicle with VDCSs, such as ABS or ARP. A twin-track model provides a balance between computational efficiency and the necessary level of complexity. The mechanical vehicle is modeled as a five-body system (one car body plus four wheels) with 16 degrees of freedom. The key components of the model are equations of motion, tires, drivetrain, aerodynamics, suspension, steering and braking. The control system is represented by sensors, logic and actuators. The simulation of the interaction between these modeled components enables the simulation of a closed control loop.

The primary phenomena that affect vehicle dynamics occur between the tire and the road surface. Thus, the tire model is an essential simulation component. The employed MF-Tyre/MF-Swift tire model [119] has been developed based on Pacejka's Magic Formula [97]. This tire model can simulate the steady-state and transient behaviors of a tire under various slip conditions. In addition, curved regular grid (CRG) tracks [133] are utilized for the road surface representation. A CRG track provides detailed three-dimensional road profiles with high precision along a predefined reference line while optimizing memory usage.

The described vehicle dynamics simulation is integrated into an overarching workflow. This workflow is designed to connect seamlessly with user input or an optimization algorithm. The vehicle dynamics control system parameters can be automatically adjusted and the simulation can be started. Once a vehicle dynamics simulation is completed, the results are post-processed to calculate the CVs. These CVs are then used to determine the objective function value. Moreover, simulation runs are parallelized and executed asynchronously. Consequently, several simulation instances can run independently without waiting for each other.

3.4 ABS Benchmark Dataset

The vehicle dynamics simulation described in Section 3.3 is computationally expensive. Benchmarking and designing optimization algorithms typically require multiple optimization runs for statistical analysis. Thus, the real-world problems are not practical for extensive experimentation. To replace the computationally expensive simulation, a dataset is created using the workflow described in Section 3.3.

For the creation of this dataset, two ABS parameters with significant influence on the ABS control behavior, denoted as x_1 and x_2, are considered. The range of each parameter is defined by a lower bound lb and upper bound ub: $x_1 \in [-5, 6]$ and $x_2 \in [-5, 4]$. For both x_1 and x_2, only a discrete set of values D_i with a resolution of 0.1 is permitted, resulting in 111 distinct possibilities for x_1 and 91 for x_2. The two-dimensional input space $\mathbb{D}^2 = \times_{i=1}^{2} D_i$ is defined by the Cartesian product. The total number of possible combinations for x_1 and x_2 is 10 101.

Generally, an optimal parameter configuration is only optimal for one vehicle setting. Five different vehicle settings are considered, with each setting consisting of a vehicle load and a tire (Table 3.1).

Table 3.1: Explanation of the vehicle settings.

Name	Tires	Vehicle Load
y_1	High performance	Partially loaded
y_2	Medium performance	Partially loaded
y_3	Under performance	Partially loaded
y_4	High performance	Fully loaded
y_5	High performance	Little loaded

The braking distance (Equation 3.2) is sensitive to slight variations in environmental conditions, vehicle characteristics and the functionality of ABS. In order to reduce the resulting variation in braking distance, the performance of a parameter configuration is averaged across the braking distances obtained from ten individual simulation runs.

Moreover, to accommodate algorithms designed for continuous input spaces, the problem is treated as quasi-continuous: for any input $\mathbf{x} \in \mathbb{R}^2$ within the specified bounds, the ABS performance is approximated by rounding x_1 and x_2 to the nearest valid points within the discrete input space $\mathbb{D}^2 = \times_{i=1}^{2} D_i$. In summary, the objective is to find a parameter setting $\mathbf{x}$ that minimizes the mean braking distance $y_i(\mathbf{x})$ for a vehicle setting i across 10 simulations:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{D}^2} \frac{1}{10}\sum_{k=1}^{10} y_i(\mathbf{x}). \tag{3.4}$$

By exhaustively simulating every combination of the two ABS parameters (a brute-force approach), the relationship between x_1, x_2 and the braking distance can be mapped for each setting considered.

Figure 3.3 shows the resulting mean braking distances across 10 simulations y_i (Equation 3.4) for each of the 10 101 possible combinations of the two ABS parameters x_1 and x_2 for the five vehicle settings (Table 3.1). All braking distances of a particular vehicle setting i are specified as the distance in meters to the corresponding optimum.

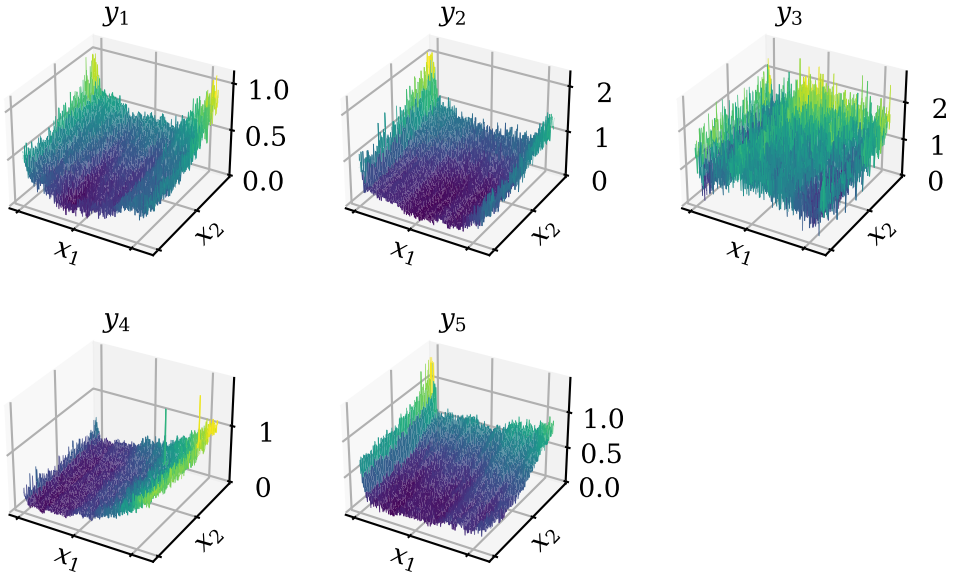

Figure 3.3: The resulting mean braking distances across 10 simulations y_i (Equation 3.4) for each of the 10 101 possible combinations of the two ABS parameters x_1 and x_2 for the five vehicle settings i (Table 3.1). The braking distances are specified as the distance in meters to the corresponding optimum $y_{i,\text{opt}}$. The objective is minimization, thus dark blue-purple indicates better solutions and yellow worse.

Chapter 4

Tuning CMA-ES Parameters

In various scientific and technical domains, ranging from economics and finance to computer science and engineering, optimization algorithms are applied to solve complex optimization problems. The performance thereby depends on the specific employed parameter configuration of the optimization algorithm. The identification of the most suitable parameter configuration of an optimization algorithm for a specific optimization problem or a class of optimization problems requires comparing the performance of different parameter configurations. One approach to this challenge is to frame automatic parameter tuning as a secondary optimization problem (Section 2.6.3).

However, an accurate assessment of the performance of an optimization algorithm requires solving the original optimization problem repeatedly. This is contrary to the primary goal in practical applications: solving the actual optimization problem. Moreover, many real-world optimization problems are computationally expensive, making it impractical to tune the parameters of the optimization algorithm directly on the original optimization problem.

Nevertheless, parameter tuning can significantly increase the performance of an optimization algorithm. The challenge is to select the optimal parameters for the optimization algorithm to solve a given computationally expensive black-box optimization problem without any prior experience in solving the original optimization problem. Consequently, in order to tune the parameters of the optimization algorithm, surrogate optimization problems that mimic the key characteristics of the original optimization problem and are less expensive to evaluate are needed. The knowledge gained from these surrogate optimization problems can then be transferred to the original optimization problem.

This Chapter is based on the results from [126, 127]. Section 4.1 introduces the proposed parameter tuning method for computationally expensive black-box optimization problems.

A set of five two-dimensional optimization problems from the field of vehicle dynamics (Section 3.4) serves as the real-world optimization problems for evaluating this method. The landscape properties of these five real-world optimization problems are analyzed to identify functions with similar properties as surrogate optimization problems (Section 4.2). Subsequently, the most effective CMA-ES configurations for these similar functions are determined and applied to the five real-world problems. The performance of the proposed method is then assessed by the performance of the transferred CMA-ES configurations on the original real-world optimization problems. Two distinct optimization approaches for parameter tuning are considered: a brute-force search within a limited CMA-ES parameter space (Section 4.3) and a comprehensive parameter tuning using a meta-optimization algorithm (Section 4.4).

The feasibility of creating a comprehensive dataset is more practical in lower dimensions, as the number of possible parameter combinations increases exponentially with higher dimensions. Thus, the real-world problems considered are two-dimensional and, therefore, relatively easy to solve. However, the focus of this chapter is not on solving these optimization problems per se but on tuning the parameters of the optimization algorithm and evaluating the effectiveness of transferring configurations tuned on surrogate optimization problems. Consequently, the fidelity with which surrogate optimization problems replicate the landscape properties of the original problems is prioritized over problem dimensionality. This requires actual real-world optimization problems since benchmark problems, such as the 24 BBOB functions (Section 2.6.1), cannot fully capture the intricacies and complexities present in real-world scenarios. The five real-world problems differ in the vehicle setting. The objective function, and thus the optimization problem from an engineering perspective, is the same. In this way, it can be investigated how different the optimization landscapes are for a class of real-world optimization problems, such as different vehicle settings, and further, whether the proposed method can find different optimal parameter configurations for each of the five real-world optimization problems.

However, to demonstrate the applicability of the proposed method in higher dimensions, Section 4.5 examines five instances of the ten-dimensional Büche-Rastrigin function from the BBOB benchmark suite as an original optimization problem. Furthermore, in Section 4.6, a comparative analysis of several meta-optimization algorithms is conducted. Finally, the results of this chapter are concluded in Section 4.7.

4.1 Methodology

Selecting the optimal algorithm parameters for an optimization problem without solving the original problem requires predicting the performance of the algorithm without prior problem-solving attempts. One approach is to assess the performance of an algorithm on optimization problems that share properties similar to those of the original problem.

However, to assess the performance of parameter configurations of an optimization algorithm in the first place, multiple full optimization runs must be executed. To ensure computational efficiency, it is advisable to perform assessments on computationally inexpensive problems that share key characteristics with the original optimization problem. Benchmark problems can serve as surrogate optimization problems for tuning the parameters of an optimization algorithm. In the following, these surrogate problems are referred to as tuning references. After identifying the most effective parameter configuration using the tuning references, this configuration can be applied to the original optimization problem.

This thesis presents a structured process for efficiently solving black-box optimization problems by tuning the parameters of an optimization algorithm with the use of tuning references. The process consists of four distinct steps (Figure 4.1):

1) identification of similar optimization problems (tuning references),
2) tuning the optimization algorithm's parameters on tuning references,
3) transfer of the best parameter configuration on tuning references, and
4) application of the tuned optimization algorithm to solve the original problem.

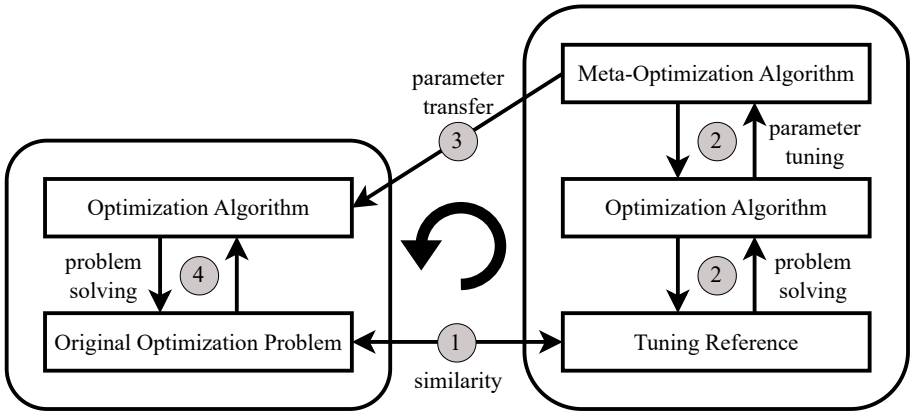

Figure 4.1: Schematic representation of the methodology for parameter tuning by employing computationally inexpensive tuning references similar to the computationally expensive original optimization problem.

The following sections address some unanswered details within this process, such as how to quantify the similarity between optimization problems (Section 4.1.1) and how to acquire a large set of surrogate problems from which the tuning references can be selected (Section 4.1.2).

4.1.1 Similarity Quantification

The assessment of similarity between two optimization problems is determined by the characteristics of their respective tasks. These characteristics include the properties of the objective function landscape. To systematically capture and quantify the high-level characteristics of the objective function landscape, Exploratory Landscape Analysis (ELA) is employed (Section 2.5). ELA provides a set of features for each problem based on a limited sample of points. These features characterize the low-level properties of the objective function landscape.

However, the raw feature set generated by ELA can contain overlapping or redundant information. Principal Component Analysis (PCA) [69] is employed to reduce the dimensionality of the feature set while retaining the most salient information.

The refined feature set obtained from PCA allows for a quantification of the degree of similarity between two distinct optimization problems p_1 and p_2. This quantification is achieved through the computation of the city block distance between their respective feature vectors, $\mathbf{f}_{p_1}$ and $\mathbf{f}_{p_2}$:

$$d_{\text{sim}}(p_1, p_2) := \sum_i |\mathbf{f}_{p_1,i} - \mathbf{f}_{p_2,i}|. \tag{4.1}$$

The city block distance, also known as the Manhattan distance, is particularly suited for this task since it is sensitive to variations in feature space. The distance between two vectors is measured as the sum of the absolute differences between coordinates.

A smaller distance d_{sim} is indicative of a greater degree of similarity between the problems in terms of their ELA features. Conversely, a larger distance points to a more pronounced dissimilarity. In a nutshell, this metric can quantify the similarity for comparing and contrasting optimization problems based on their feature vectors. The computed distance can guide the selection of suitable optimization problems for use as surrogate problems.

4.1.2 Artificial Functions

Standard benchmark problems can be utilized to tune the parameters of an optimization algorithm. For instance, the Black-Box Optimization Benchmarking (BBOB) suite provides a set of 24 diverse functions. These functions range from separable functions with a clear global structure to complex multi-modal functions with no global structure (Section 2.6.1). However, in order to expand the range of potential tuning references and encompass a greater variety of landscapes, the generation of artificial functions is considered.

An artificial function (AF) is constructed for a specific purpose and is often defined by a mathematical expression. These functions can be used in various contexts. An example of an application is algorithm tuning. Here an artificial function may be designed, e.g., to have multiple local minima to challenge optimization algorithms.

Tian et al. [129] proposed a method for generating a large number of artificial functions. This method employs a randomly constructed tree structure to generate an artificial function. Operands are placed in the leaf nodes. Operators are placed in the non-leaf nodes. The operands and operators are selected with an associated probability from a set of seven operands and 20 operators. Operands are either real numbers or representations of the decision variables, such as the decision vector $\mathbf{x}$ or the first decision variable x_1. The operators are classified into four binary operators, such as addition or subtraction, eleven unary operators, such as the sine function or the square root, and five vector-oriented operators, such as the sum or the mean of a vector.

The hierarchical tree representation of an artificial function is then translated into a mathematical expression. Once created, this expression represents the artificial function and returns a function value when evaluated at a point $\mathbf{x} \in \mathcal{X}$. Both the generation and the evaluation processes of these artificial functions are computationally inexpensive. Furthermore, the desired dimensionality of the artificial function can be specified beforehand.

The instance-generating mechanism of the BBOB suite [48] is replicated by shifting and rotating the artificial functions after generation. This allows to generate several instances of the same artificial function. The optimization landscape is fundamentally influenced by the objective of maximization or minimization. Therefore, the negation of an artificial function is created by multiplying the original artificial function by minus one. Figure 4.2 presents five instances of a two-dimensional randomly generated artificial function along with their negative counterparts.

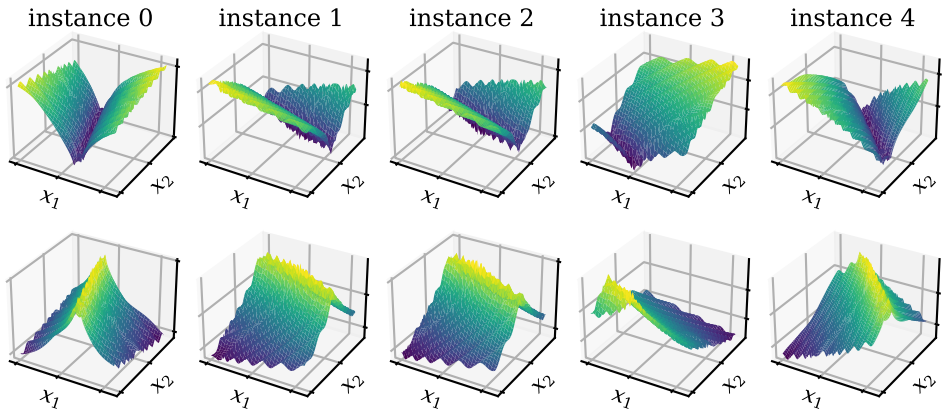

Figure 4.2: Five instances of a two-dimensional randomly generated artificial function (top row) and their negative counterparts (bottom row).

4.2 Analysis of Landscape Similarities

To obtain tuning references for the five two-dimensional, real-world problems from vehicle dynamics design (Section 3.4), a large set of artificial functions is generated, and the ELA features of these problems are computed for the similarity quantification (Section 4.2.1). The similarity between the real-world problems, the generated artificial functions and benchmark functions from the BBOB suite is analyzed, and the functions with the highest similarity (Equation 4.1) to the real-world problems are identified (Section 4.2.2).

4.2.1 Experimental Setup

In addition to the 24 BBOB functions (Section 2.6.1), a set of 100 000 two-dimensional artificial functions is generated. Five instances of each artificial function are considered, as well as their negative counterparts. This results in a total of 120 BBOB functions and 1 000 000 unique artificial functions. Each BBOB function is denoted as $\text{BBOB}_{id,i}$, where id corresponds to the specific identifier within the BBOB suite (Table 2.1), and i denotes the instance number. The 100 000 artificial functions are systematically numbered and referred to as $\text{AF}_{\text{number},i}$. For the negated versions of these functions, the notation is extended to $\text{AF}_{\text{number},i,\text{n}}$. The input domain for all functions, including the artificial and BBOB functions, is aligned with the input space of the five real-world problems y_i, $x_1 \in [-5, 6]$ and $x_2 \in [-5, 4]$.

To quantify similarity using ELA, the focus is on features that can be computed efficiently without additional resampling. This approach results in 55 individual features, categorized into five groups: classical ELA (distribution, level and meta-features), information content, dispersion, nearest better clustering and principal component analysis (Section 2.5.2). Five features (ela_level.lda_qda_{10, 25, 50}, ic.eps_{s, ratio}) were found to be infeasible to compute across all functions and were subsequently omitted from the analysis. In the context of ELA, a sample size of 50 times the dimensionality of the problem space is suggested as a balanced trade-off between accuracy and computational effort when classifying the BBOB functions using ELA features [71]. To improve the precision of feature estimation, a Sobol' sequence design [96, 121] with 1 000 samples is employed. To ensure equal contribution of each feature to the similarity quantification, the feature values are min-max scaled to $[0, 1]$. The feature computation was successfully completed for 99.5% of the randomly generated artificial functions. The remaining 0.5% of cases where feature calculation failed are primarily caused by the emergence of "not a number" values within the function values or the presence of a flat fitness landscape. To remove redundant features and reduce the dimensionality of the feature space, PCA is used. A cumulative variance greater than 0.999 results in a dimensionality of the feature space of 31.

The artificial functions are generated using the Python implementation provided by [83], which itself is based on the methodology outlined in [129]. This implementation is extended with the described instance generation mechanism. The pflacco package [102] is utilized for the calculation of ELA features. pflacco offers a native Python implementation of the extensive collection of ELA features available in the flacco R package [73].

4.2.2 Results

The landscape properties of each function are described by a feature vector, which is then reduced using PCA. Figure 4.3 shows the location of the functions based on the two primary principal components derived from the PCA. The BBOB functions occupy only a subset of the possible space. The gaps within this space are filled by the randomly generated artificial functions, suggesting a broader exploration of landscape characteristics. The real-world problems (Table 3.1) are scattered over a relatively large area in the principal component space, suggesting notable dissimilarities within their respective landscapes. In particular, only problems y_1 and y_2 are close to each other, indicating a high degree of similarity.

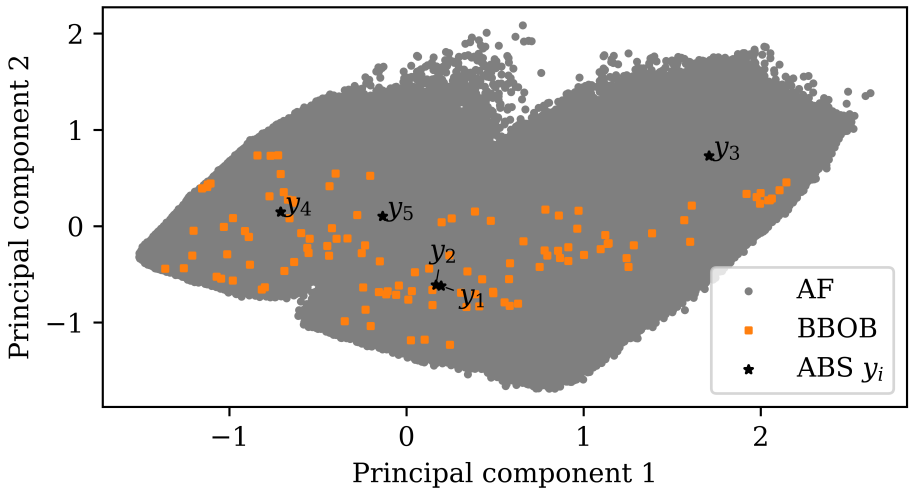

Figure 4.3: Positions defined by the two main PCA components of the 50 ELA features for each artificial function (AF), the BBOB functions and the ABS real-world problems y_i.

Employing Equation 4.1, the exact distances quantifying the similarity between the five real-world problems y_i and the other functions can be calculated. Figure 4.4 presents the range of these distances. The smallest distance value is observed between problems y_1 and y_2, with a value of 0.84. In stark contrast, the distance between problems y_2 and y_3 is the greatest with 6.4. The relatively small distance of 0.84 between y_1 and y_2 implies that these two problems share similar landscapes, whereas the other real-world problems demonstrate distinct dissimilarities from both y_1 and y_2, as well as among themselves.

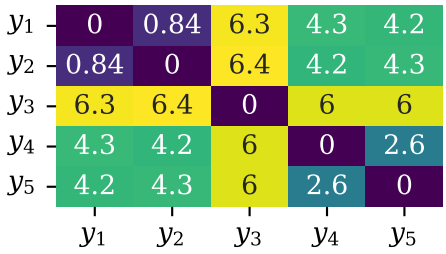

Figure 4.4: Similarity distances of the five real-world problems y_i to each other (as defined in Equation 4.1).

Figure 4.5 illustrates the distances between the real-world problems and their ten closest artificial counterparts, as well as the ten BBOB functions most similar to each real-world problem. On average, the artificial functions exhibit a distance of 1.2 from the real-world problems, while the BBOB functions show an average distance of 2.8. Based on the ELA features, the artificial functions resemble the real-world problems closer compared to the BBOB functions. Nonetheless, apart from y_1 and y_2, the real-world problems tend to be more dissimilar to each other than even the most similar artificial function identified for each respective problem.

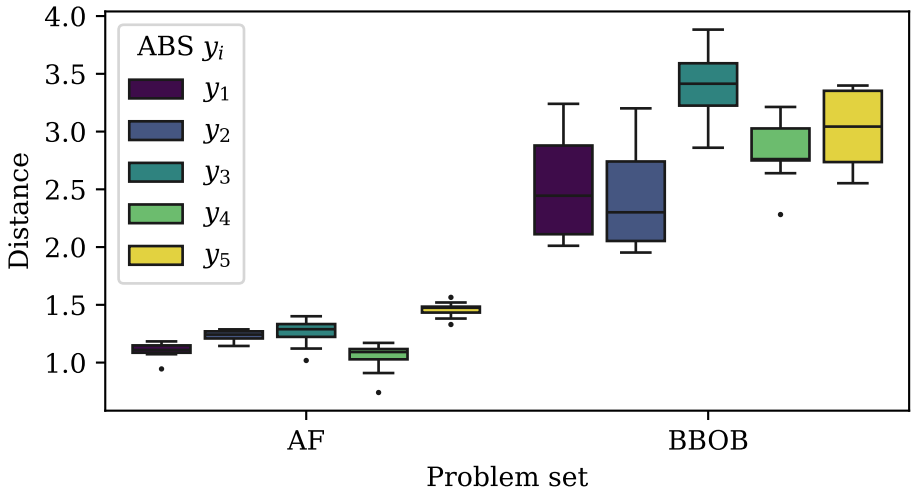

Figure 4.5: Similarity distance between the five real-world problems y_i and their ten most similar artificial functions (AF) and BBOB functions (as defined in Equation 4.1).

Figure 4.6 illustrates the landscape of the five real-world problems y_i, alongside the landscapes of the five artificial functions that exhibit the highest degree of similarity to each of them. All real-world problems y_i are characterized by an inherent noise component, contributing to a highly multi-modal landscape. This is similarly reflected in the landscapes of the corresponding artificial functions.

Moreover, apart from y_3 (under performance tires), the real-world problems exhibit a global structure. Notably, the global structure of y_1 and y_2 is similar, which aligns with the small distance calculated (Figure 4.4). The artificial functions that are identified as similar to these real-world problems successfully replicate this global structure. In fact, certain artificial functions are similar to both y_1 and y_2, such as $\text{AF}_{17523,4}$ and $\text{AF}_{27980,4}$.

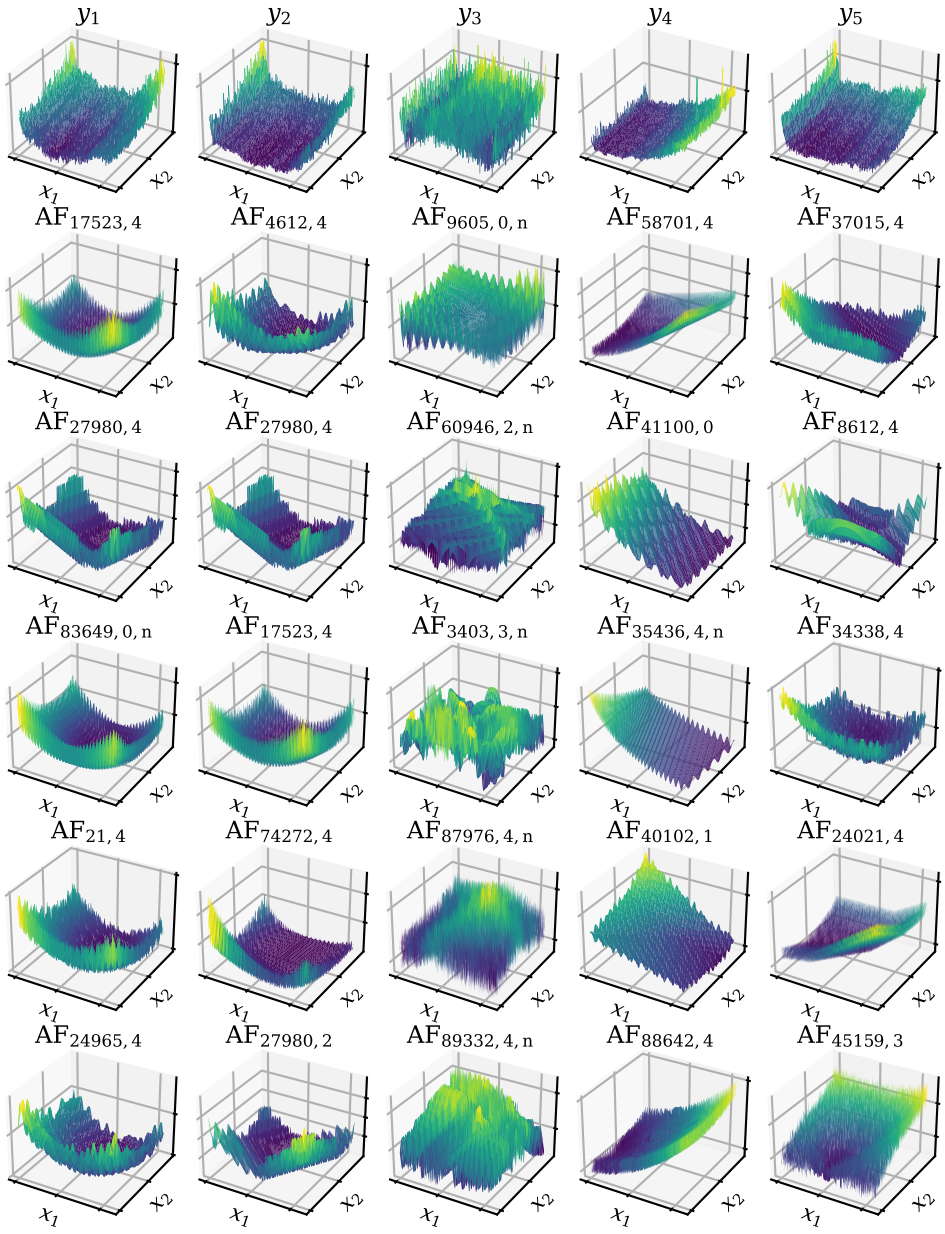

Figure 4.6: The landscapes of the five two-dimensional real-world problems y_i and the five most similar artificial functions (AF) to each problem y_i. The objective is minimization. Thus, dark blue-purple indicates better solutions and yellow worse.

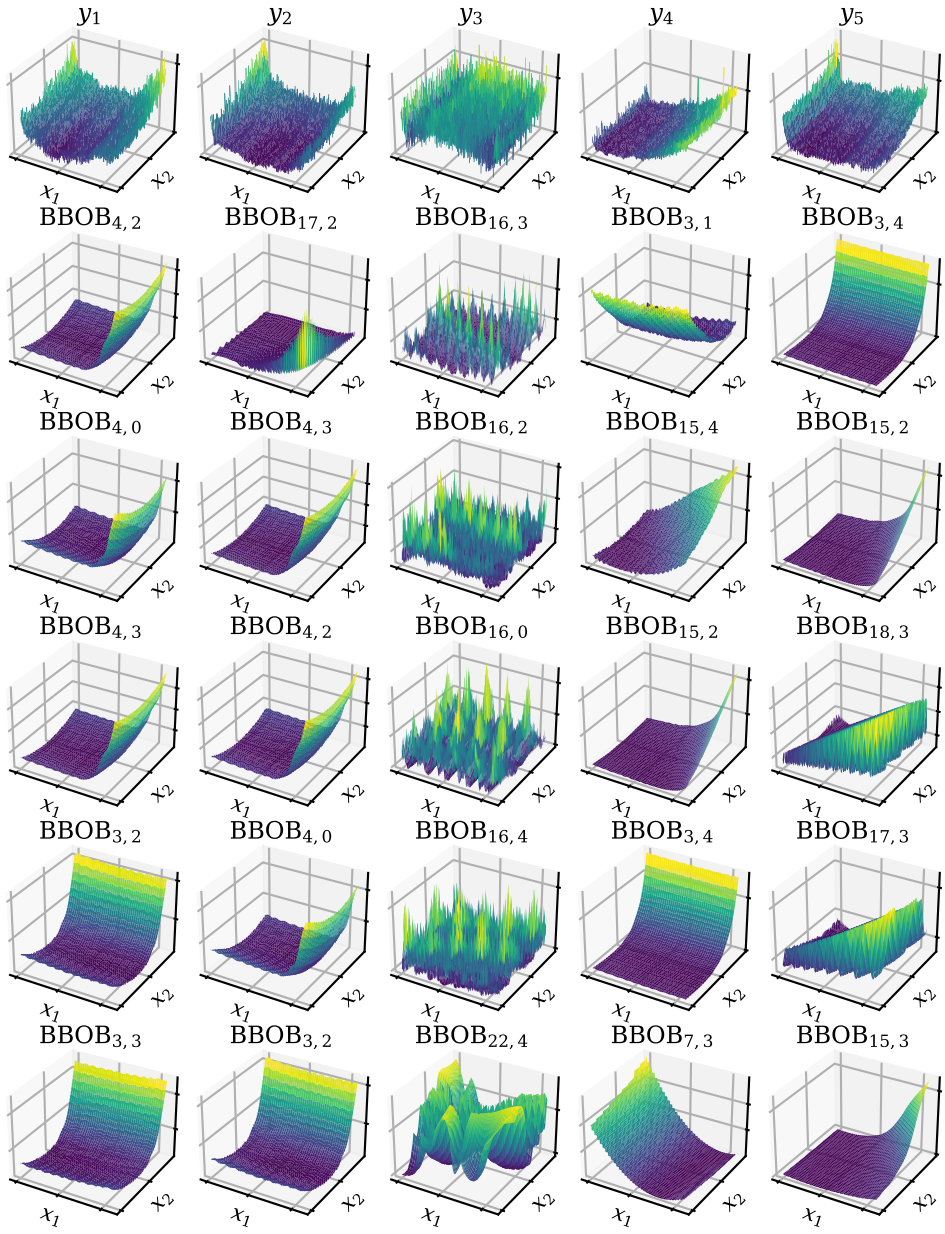

Figure 4.7: The landscapes of the five two-dimensional real-world problems y_i and the five most similar BBOB functions to each problem y_i. The objective is minimization. Thus, dark blue-purple indicates better solutions and yellow worse.

Figure 4.7 illustrates the landscapes of the five real-world problems y_i, along with the landscapes of the five BBOB functions from the considered set of 120 that exhibit the highest degree of similarity to each of them. The BBOB functions capture essential landscape properties. This is especially noticeable for y_3, where the four most similar BBOB functions are all instances of the Weierstrass function BBOB_{16}. Similar to the landscape of y_3, the Weierstrass function is characterized by high multimodality and a weak global structure. For y_1 and y_2, nine out of the ten most similar BBOB functions are instances of either the Rastrigin function BBOB_3 or the Büche-Rastrigin function BBOB_4. Furthermore, the Rastrigin function is identified as the most similar BBOB function for both y_4 and y_5. Notably, the Rastrigin function BBOB_{15} exhibits similarities to both y_4 and y_5. Overall, some BBOB functions, in particular the Rastrigin function, are similar to all four real-world problems: y_1, y_2, y_4 and y_5, indicating shared high-level landscape properties with these four real-world problems and also among the four real-world problems.

However, essential landscape properties of the real-world problems are not resampled by the most similar BBOB functions. For example, the Rastrigin function and the Büche-Rastrigin function are separable. Thus, each variable can be optimized independently of the others, which does not reflect the more complex interactions between the real-world problem parameters, x_1 and x_2. Overall, the BBOB functions do not resemble the global structure of the real-world problems as much as the artificial functions.

The global structure and appearance of functions have a strong influence on the similarity between these functions. Figure 4.8 shows the first five instances of the sphere function. Due to shifting and rotation, the global minimum for each instance is located in a different position. This, combined with the bounded decision space, results in vastly different landscapes.

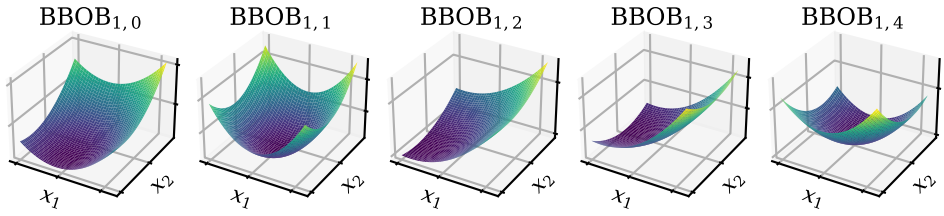

Figure 4.8: The landscapes of the first five instances of the sphere function. The objective is minimization. Thus, dark blue-purple indicates better solutions and yellow worse.

The dissimilarity between the five considered instances of the sphere function (Figure 4.8) is quantified by calculating the distances according to Equation 4.1. For example, the calculated distance to instance 1 exceeds 5 for instances 2 and 3. In contrast, the instances with the smallest distance are 1 and 4, with a distance of 1.6.

This pattern of high dissimilarity across the instances of the same BBOB function can also be observed for other BBOB functions [84]. The reason for this is that the ELA features are specifically designed to distinguish between instances of the same BBOB function.

4.2.3 Conclusion

The randomly generated artificial functions successfully augment the BBOB function set by covering a wide range of landscapes with different properties. With the proposed distance metric (Equation 4.1) to quantify the similarity between landscapes of optimization problems, similar function landscapes to the five two-dimensional real-world problems can be identified.

Furthermore, the process of randomly generating an enormous number of artificial functions yields landscapes that are more similar to real-world problems than the few considered instances of the 24 BBOB functions. This similarity is supported by both quantitative comparisons and visual inspection of the plotted landscapes, confirming that the considered ELA features effectively capture the essential high-level properties and global structure of the landscapes.

Although instances of the same BBOB functions share many high-level landscape properties, such as modality or separability, instances of the same BBOB function can exhibit considerable variation in their ELA feature values. This variation is also evident in their visual representations. Moreover, functions that appear visually similar do not necessarily share the same high-level properties, such as separability.

In the following sections, the ten most similar artificial functions and the ten most similar BBOB functions to each of the five real-world problems are selected as tuning references for further analysis. Due to duplicates within the 100 similar functions initially identified, only a total of 34 distinct BBOB functions and 44 distinct artificial functions are ultimately selected.

4.3 Brute Force Search

The aim of the following study is to investigate the transferability of the performance of CMA-ES parameter configurations from the selected tuning references to the five two-dimensional real-world problems (Section 3.4). A limited set of CMA-ES parameters and options is considered. Using a brute force search approach, all possible combinations of these parameter options are generated and evaluated on the five real-world problems, as well as on the 34 selected BBOB functions and the 44 selected artificial functions (Section 4.3.1). This method provides a table containing a performance value for each parameter configuration across all functions considered. The collected data is subsequently used to analyze the correlation between the performance on the tuning references and the real-world problems (Section 4.3.2). Furthermore, the specific parameter options that lead to either strong or weak performance on the functions are examined in order to identify similarities between the tuning references and the original optimization problems in terms of the impact of specific parameter options on these functions.

4.3.1 Experimental Setup

CMA-ES is considered a state-of-the-art method for solving single-objective continuous black-box optimization problems. The performance of CMA-ES thereby depends on the chosen parameters and variants (Section 2.4). Table 4.1 summarizes the parameters and variants considered for the brute force search, along with their respective options. Combining these options results in a total of 864 different parameter combinations.

Table 4.1: Parameters of the IPOP-CMA-ES with value options considered in the brute force search. The option 0 for each parameter is the default configuration.

Parameter	Description	Option 0	Option 1	Option 2
λ	Number of offspring	6	12	18
μ_r	Selection ratio $\mu_\mathrm{r} = \frac{\mu}{\lambda}$	$\frac{1}{2}$	$\frac{1}{6}$	$\frac{5}{6}$
σ_0	Initial standard deviation	2	4	6
bc	Box constraint handling	Projection	Reflection	
Active	Covariance matrix update	off	on	
Elitism	Selection strategy	(μ, λ)	$(\mu + \lambda)$	
om	Orthogonal mirrored sampling	off	on	
Weights	Option for recombination	logarithmic	equal	
Restart	Restart strategy	IPOP		

Because a restart can utilize the remaining evaluation budget and thus has the potential to improve performance, the IPOP restart strategy is selected and always enabled by default. The CMA-ES default parameter configuration, as referenced in the following section, is the configuration with parameter option 0 set for each parameter.

To assess the performance of CMA-ES parameter configurations in solving optimization problems, the Area Under the Curve (AUC) of the ECDF is utilized. This metric gauges the effectiveness of an optimization algorithm in terms of anytime performance (Section 2.6.2). The ECDF curves are computed using 81 target values logarithmically distributed from 10^{8} to 10^{-8}. Furthermore, the AUC values are normalized by the evaluation budget. To use the same target values for all functions, the objective functions are shifted by the function value at the global optimum. The adjusted objective functions now return the distance to the global optimum.

The performance of a solution candidate for the real-world problems is given by the mean braking distance over ten braking maneuvers (Equation 3.4). The objective function of the real-world problem returns the distance to the global best braking distance. As the input parameters for the five real-world problems are discrete, the returned objective function values are also discrete. For instance, for the real-world problem y_1, the second-best solution is already 0.0005 m less optimal, and the third-best is 0.00146 m less optimal. Therefore, achieving a smooth approach and convergence to the optimal solution by the optimization algorithm is not possible for the real-world problems. Moreover, due to noise and simulation accuracy, a difference in braking distance measured in millimeters cannot be regarded as significant. In the real world, such a small difference is within the measurement tolerance and, therefore, practically indistinguishable. Therefore, target values as precise as 10^{-8} are not suitable. Instead, the performance of a parameter configuration on the real-world problems is evaluated using target values of 0.2 m, 0.1 m, 0.05 m and 0.01 m. A solution that is within 1 cm of the optimal braking distance is deemed adequate for solving the real-world problems.

To evaluate the performance of a parameter configuration on the two-dimensional problems, the mean AUC value is calculated over 200 independent runs, each with an evaluation budget of 2 000 evaluations. For the artificial functions, an exhaustive search is conducted using CMA-ES with the default parameter configuration, and the evaluation budget is increased to 200 000 evaluations. This ensures that at least one high-quality solution is discovered.

The study employs a modular implementation of CMA-ES [26, 131], which supports various CMA-ES variants that can be combined in an arbitrary manner.

4.3.2 Results

Each of the 864 parameter configurations is evaluated across all five real-world problems and the tuning references. For each real-world problem, the configuration that achieves the highest average AUC value across the ten most similar tuning references is chosen. This reduces the risk of overfitting to a single function. Two sets are considered as tuning references: the artificial functions and the BBOB functions. In addition to the configurations derived from tuning references, the six parameter configurations that exhibit the highest AUC values on each real-world problem and across all five real-world problems are selected for comparison.

Figure 4.9 presents the AUC values obtained on the five real-world problems alongside the average AUC across all. These values are expressed as the relative improvement in the AUC value in percent compared to the default configuration.

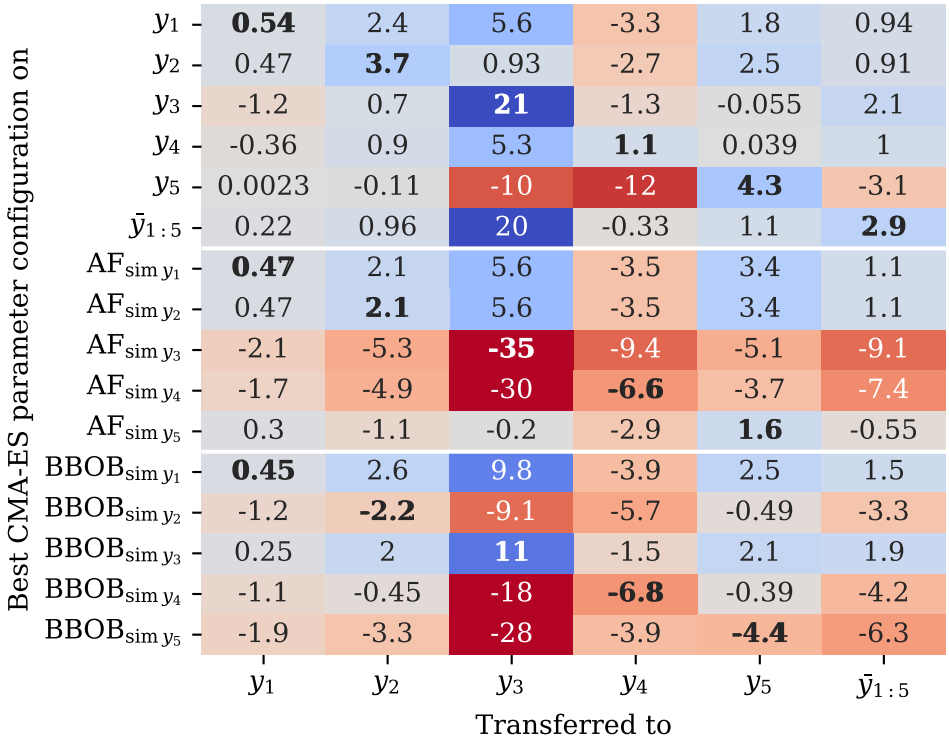

Figure 4.9: Relative improvement in percent of AUC to the default configuration when transferred to the five real-world problems and across all $\bar{y}_{1:5}$. The best configuration on each and across all real-world problems, across the ten most similar artificial functions $\mathrm{AF}_{sim\ y_i}$ and across the ten most similar BBOB function $\mathrm{BBOB}_{sim\ y_i}$ to each real-world problem $\bar{y}_{1:5}$ are listed. The values on the diagonals are each highlighted in bold.

The optimal parameter configuration for each real-world problem consistently outperforms the default configuration. The degree of improvement varies across the problems. The improvement for y_1 is 0.54%, and for y_4 1.1%. In contrast, y_2 and y_5 see more notable gains of 3.7% and 4.3%, respectively. The most substantial enhancement is observed for y_3, with an impressive improvement exceeding 21%. The best configuration across all real-world problems achieves only marginal performance improvements for y_1, y_2 and y_5, and a slight decrease for y_4. However, this configuration remains the best on average due to the significant 20% enhancement on y_3. The weak global structure in the landscape of y_3 impedes finding the solution. This is reflected in the AUC value of the default configuration, which is only 0.49. In contrast, the AUC values for the other problems are 0.98 for y_1, 0.89 for y_2, 0.91 for y_4 and 0.76 for y_5. Therefore, y_3 presents the greatest opportunity for performance gains due to its initially lower AUC value.

When the best parameter configurations are exchanged between the similar problems y_1 and y_2, an improvement in performance is observed, reflecting their similarity. However, applying these parameter configurations to the more distinct problems y_3, y_4 and y_5 yields variable results: performance deteriorates for y_4 but improves for y_5 and y_3. Interchanging configurations among the real-world problems tend not to result in significant performance improvements and can actually reduce performance. For example, employing the best configuration for y_5 on y_3 and y_4 results in performance degradation of roughly 10% and 12%, respectively. Also, transferring any of the best parameter configurations from y_1, y_2, y_3 or y_5 to y_4 results in performance degradation.

The transfer of the best parameter configuration from the ten most similar artificial functions to each of the five real-world problems leads to increased performance on y_1, y_2 and y_5 compared to the default configuration. However, for y_3 and y_4, the resulting performance is significantly worse. Moreover, these two best configurations on $\mathrm{AF}_{sim\, y_3}$ and $\mathrm{AF}_{sim\, y_4}$ also perform very poorly when applied to the other three real-world problems. The best configurations on $\mathrm{AF}_{sim\, y_1}$ and $\mathrm{AF}_{sim\, y_2}$ are identical. The reason for this is that y_1 and y_2 are very similar to each other and six of the ten most similar artificial functions are the same for both y_1 and y_2. The performance improvement gained through the transfer from similar artificial functions is noteworthy when compared to the actual best configurations for these problems.

The transfer of the best parameter configuration from the ten most similar BBOB functions to each of the five real-world problems leads to increased performance only on y_1 and y_3 compared to the default configuration.

The observed performance degradation for certain parameter configurations, when transferred to the real-world problems, may be attributed to specific parameter options that yield favorable results on the tuning references but underperform on the original problems. To identify these problematic options, an examination of the parameter options that are successful on the real-world problems is conducted. Accordingly, the 20 most effective parameter configurations for each of the five real-world problems are analyzed. Figure 4.10 illustrates the frequency of occurrence of each option for every parameter within these configurations.

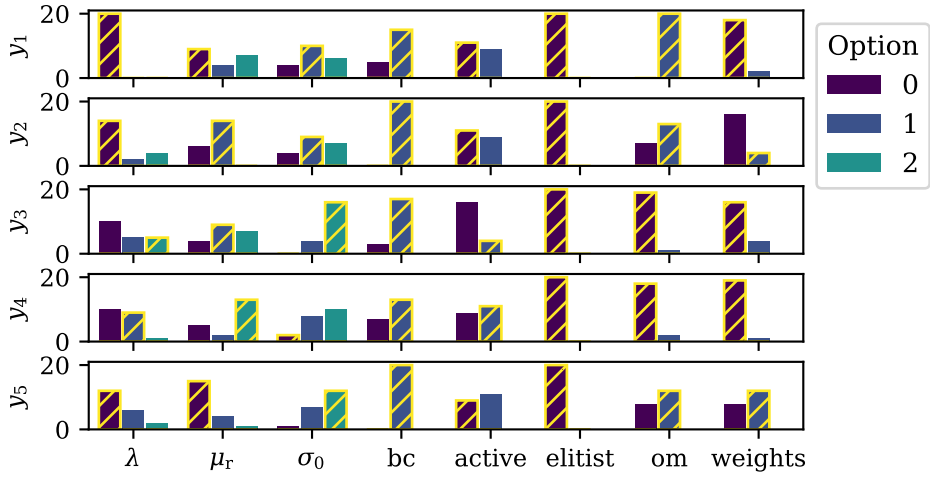

Figure 4.10: Frequency of occurrence of each parameter option given in Table 4.1 for every parameter within the best 20 configurations for each real-world problem y_i. Options corresponding to the best-performing configuration are highlighted with yellow hatching.

Different parameter configurations perform best on each real-world problem. However, there are discernible patterns and similarities in the frequency of occurrence for some parameter options within the best 20 parameter configurations. Notably, for all five real-world problems, the elitist option 0 is consistently chosen, which corresponds to the standard (μ, λ) selection strategy in CMA-ES. Similarly, the box constraint handling method of the best configuration is consistently option 1 reflection which is also more frequently observed than its counterpart, option 0 projection. The default weights option appears more often in the top 20 configurations for four out of the five problems. In contrast, the use of the active update is roughly as frequent as not using it across the problems, with the exception of y_3 where option 0 is more frequent. However, the best configuration for y_3 includes the active update, suggesting that its

importance for high-performing configurations may not be as significant. Orthogonal mirrored sampling option 1 is part of the best configuration and is frequently within the top 20 for y_1, y_2 and y_5, while option 0 is favored for y_3 and y_4.

Regarding the population size, option 0, which represents the smallest number of 6 offspring, and for the initial standard deviation, options 1 and 2, which represent larger values of 4 and 6 compared to the default value of 2, seem to offer some performance improvement across all problems. For the selection ratio, no clear pattern emerges across the five real-world problems, and the preference for each problem is also ambiguous. The only exceptions are on y_5, where the default option is more frequent, and on y_4, where option 2 is more frequent within the top 20 configurations. Interestingly, for y_2, option 2 never occurs.

Therefore, overall, for the five real-world problems, the (μ, λ) selection strategy, reflection, and the default weights option lead to good performance and should be used. Using active leads to no significant improvement. Also, a small number of offspring and an increased initial standard deviation should be used. Mirror orthogonal sampling should only be used for y_1, y_2 and y_5.

Figure 4.11 illustrates the frequency of occurrence of each option for every parameter within the best 20 parameter configurations on the ten most similar artificial functions to each of the five real-world problems.

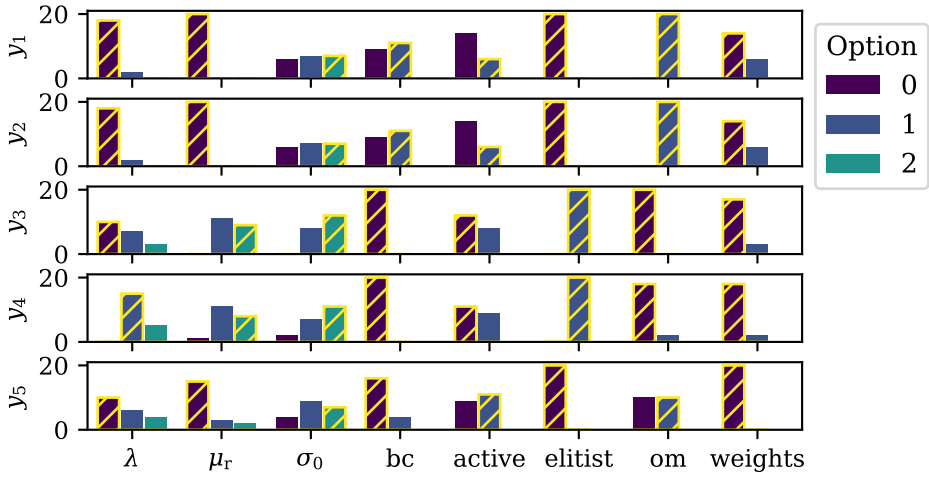

Figure 4.11: Frequency of occurrence of each parameter option given in Table 4.1 for every parameter within the best 20 configurations on the ten most similar artificial functions to each real-world problem y_i. Options corresponding to the best-performing configuration are highlighted with yellow hatching.

Patterns also emerge on the artificial functions, some of which are very similar to those observed on the real-world problems. The default weights option and the orthogonal mirrored sampling option exhibit patterns that almost perfectly resemble those seen on the real-world problems. For the active update, the step size, the number of offspring and the selection ratio, the patterns are similar for the majority of cases.

However, there is a clear difference in how often the box constraint handling method and the elitist option occur. On artificial functions similar to y_3, y_4 and y_5, the projection method is more effective, and the $(\mu + \lambda)$ selection strategy always occurs within the top 20 configurations on artificial functions similar to y_3 and y_4. In fact, the $(\mu + \lambda)$ selection strategy can be identified as the reason for the poor performance on the real-world problems. When configurations are limited to those with a (μ, λ) selection strategy coupled with reflection, the best parameter configuration significantly enhances performance on the real-world problems. For y_3, the improvement is the most pronounced, with a leap from -35% to 21%. For y_4, the increase is from -6.6% to -0.4%, and for y_5, a slight improvement from 1.6% to 1.8% is observed (Figure 4.12). Thus, the artificial functions similar to y_3 and y_4 exhibit properties where the $(\mu + \lambda)$ selection strategy can be beneficial.

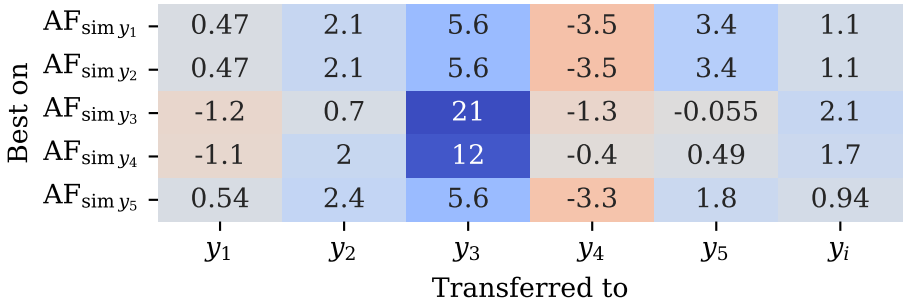

Figure 4.12: Relative improvement of the AUC to the default configuration when transferred to the five real-world problems and across all y_i. The performance of the best configuration, limited to those with a (μ, λ) selection strategy coupled with reflection, across the ten most similar artificial functions $\mathrm{AF}_{sim\ y_i}$ to each of the five real-world problems y_i are listed.

Similarly, on the BBOB functions that are similar to each of the five real-world problems, the $(\mu + \lambda)$ selection strategy is often within the top 20 parameter configurations (Figure 4.13). Furthermore, the parameter options that perform well on the real-world problems differ from the pattern within the similar BBOB functions, except for the weights option and the box constraint handling. The difference in performance is also evident in the decrease observed when applying the best parameter configurations from the similar BBOB functions to the real-world problems.

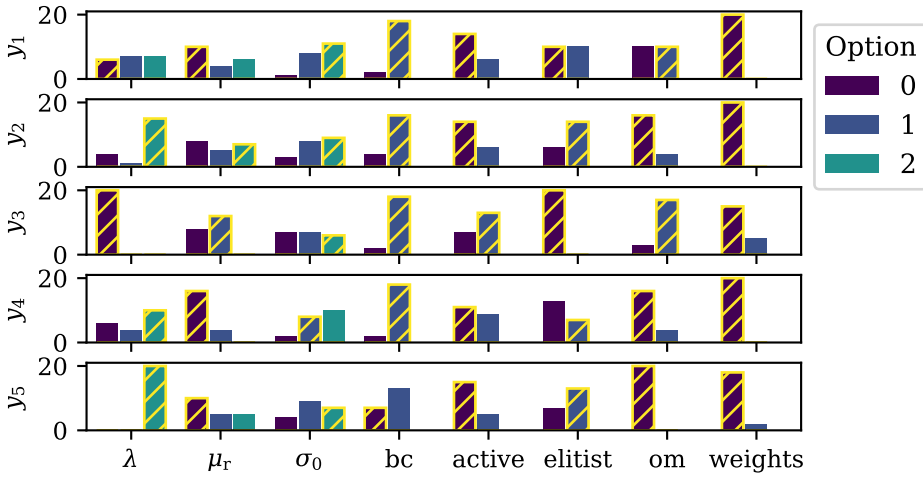

Figure 4.13: Frequency of occurrence of each parameter option given in Table 4.1 for every parameter within the best 20 configurations on the ten most similar BBOB functions to each real-world problem y_i. Options corresponding to the best-performing configuration are highlighted with yellow hatching.

4.3.3 Summary

Significant opportunities for performance improvement are revealed by the conducted study. For each of the five real-world problems, a different parameter configuration achieves the best performance. However, clear patterns have emerged across all five real-world problems. The (μ, λ) selection strategy, reflection and the default weights option lead overall to an enhanced performance. The active update option appears to have a negligible impact on the results and can be retained in its default setting. Moreover, employing a smaller number of offspring and a larger initial standard deviation is recommended. The use of mirrored orthogonal sampling is particularly beneficial for the real-world problems y_1, y_2 and y_5.

However, these patterns do not always translate to similar artificial functions. The $(\mu + \lambda)$ selection strategy, while advantageous for artificial functions similar to y_3 and y_4, results in a significant drop in performance when applied to their real-world counterparts. Similarly, transferring the box constraint handling method from the artificial functions similar to y_3, y_4 and y_5 results in a performance drop. Yet, when these two parameter options are set beforehand, the similar functions always provide a configuration that is better than the default configuration in four out of five cases. Only for y_4 is the performance 0.4% worse.

The application of parameter configurations derived from BBOB functions that are similar to the real-world problems is advantageous for y_1 and y_5, albeit marginally. However, for the other problems, such an approach is detrimental because the patterns of parameter options that frequently occur are very dissimilar. Consequently, transferring parameters from BBOB functions to real-world problems is generally not recommended. An exception is observed for y_3, where similar BBOB functions, often instances of the Weierstrass function, provide a valuable reference for tuning.

The objective function landscapes of the real-world problems exhibit multimodality because of noise. This characteristic is reflected in the similar functions by an internal sine function. Especially the $(\mu + \lambda)$ selection strategy is not recommended for noisy functions. Given this knowledge, a $(\mu + \lambda)$ selection strategy that is effective on artificial functions should be approached with caution when applied to the real-world problems. The discrepancy in how these artificial functions mimic noise-induced multimodality suggests that the $(\mu + \lambda)$ selection strategy should be disregarded entirely for the real-world applications.

4.4 Meta-Optimization

Following the previous Section 4.3, this section aims to expand the number of CMA-ES parameters considered for tuning to unlock additional potential for performance improvement. Especially the learning rates of CMA-ES have a significant impact on the performance of CMA-ES. With the expansion of CMA-ES parameters, the use of a meta-optimization algorithm becomes necessary.

Since each real-world problem has a unique optimal parameter configuration, it is necessary to perform similarity quantification and tuning for each specific case. Practically, however, it is not feasible to perform a new analysis for each variation just to determine the best parameter setting for a particular vehicle setting (Table 3.1). Therefore, the objective is to identify a robust parameter configuration that enhances the performance across all five real-world problems.

The experimental setup details are provided in Section 4.4.1. The best configurations identified for the real-world problems are compared with the optimal configurations derived from the most similar artificial functions and the most similar BBOB functions (Section 4.4.2).

4.4.1 Experimental Setup

The brute force search revealed that the $(\mu+\lambda)$ selection strategy does not yield beneficial results when applied to the five two-dimensional real-world problems (Section 4.3). However, this strategy did show some utility when employed on artificial functions similar to the real-world problems. Therefore, in the upcoming meta-optimization process, the $(\mu + \lambda)$ selection strategy is excluded from the set of parameters to be tuned. Table 4.2 lists the CMA-ES parameters selected for consideration and their corresponding value ranges.

Table 4.2: CMA-ES parameters with value space considered in the meta-optimization.

Parameter	Description	Space
c_1	Learning rate rank-one update	$]0, 1]$
c_c	Learning rate adaption of C	$]0, 1]$
c_μ	Learning rate rank-μ update	$]0, 1]$
c_σ	Learning rate step size control	$]0, 1[$
λ	Number of offspring	$\{6, 9, ..., 18\}$
μ_r	Selection ratio $\mu_\mathrm{r} = \frac{\mu}{\lambda}$	$\{\frac{1}{6}, \frac{2}{6}, ..., \frac{5}{6}\}$
σ_0	Initial standard deviation	$\{2, 3, \ldots, 6\}$
bc	Box constraint handling	{projection, reflection, wrapping, reinitialization}
Active	Covariance matrix update	{on, off}
Orthogonal	Orthogonal sampling	{on, off}
Mirrored	Mirrored sampling	{on, off}
Weights	Option for recombination	{logarithmic, equal}
Restart	Restart strategy	{IPOP, BIPOP}

The tuning process is performed separately on three distinct sets: the five real-world problems, the 24 distinct BBOB functions and the 44 distinct artificial functions that resemble the real-world problems. The performance metric used is consistent with the one employed in the brute force search (Section 4.3.1). However, performance is assessed across all functions within each set. Therefore, the number of runs to assess the performance of a single parameter configuration during meta-optimization is reduced from 200 runs to 100 for the real-world problems and to 20 for both the artificial and BBOB functions.

CMA-ESwM is selected as the meta-optimization algorithm (Section 2.4.3). The meta-optimization evaluation budget is set to 2 500 for a single meta-optimization run. To ensure statistically significant results, the meta-optimization is repeated five times for each set. This study utilizes Optuna's implementation of CMA-ESwM [2].

4.4.2 Results

The brute force search assessed 864 parameter configurations, and the optimal configuration identified improves the performance across all five real-world problems by 2.9% compared to the default parameter configuration (Figure 4.9). Further improvements are observed when in addition, the learning rates of CMA-ES parameters are considered. The performance gains of the meta-optimization directly on the real-world problems range from at least 3% to 4.4% across the five conducted runs. The average improvement is 3.6% across all runs.

However, in practical scenarios, tuning would not be performed directly on real-world problems. Therefore, the results obtained serve as a benchmark for the best-case scenario. Tuning references are used as surrogate problems for this purpose. When the artificial functions most similar to the real-world problems are used as tuning references, the average improvement gain of the five optimal configurations on the real-world problems is 1.1%, with the best configuration achieving a 1.7% increase and the least effective configuration showing a 0.7% improvement. Using the most similar BBOB functions as tuning references results in an average improvement of 1.9% across the five configurations, with results ranging from 1.7% to 2.5%. All three sets show significant variation in improvement.

In contrast to the previous study, using artificial functions as tuning references leads to inferior outcomes compared to using BBOB functions. The optimal configuration derived from the artificial functions performs similarly to the least effective out of the five meta-optimization runs using the BBOB functions as a tuning reference.

Figure 4.14 shows the values of the learning rates of the optimal configurations from the meta-optimization runs and the recommended default values within CMA-ES. Across all five meta-optimization runs on the real-world problems, a slight increase in c_μ and a decrease in c_1 are found to be optimal. The learning rate for the rank-one update of the covariance matrix c_1 utilizes information on correlations between generations by exploiting the evolution path. The rank-μ update efficiently incorporates information from the entire population. Reducing c_1 and increasing c_μ suggests that the covariance matrix updates should be more influenced by information from the current population rather than from past generations via the evolution path to improve the performance of CMA-ES on the real-world problems. The similar functions capture this phenomenon but do not reflect the exact magnitude. The increase from the default value is excessive for c_μ, while the decrease from the default value is inadequate for c_1 compared to the optimal values.

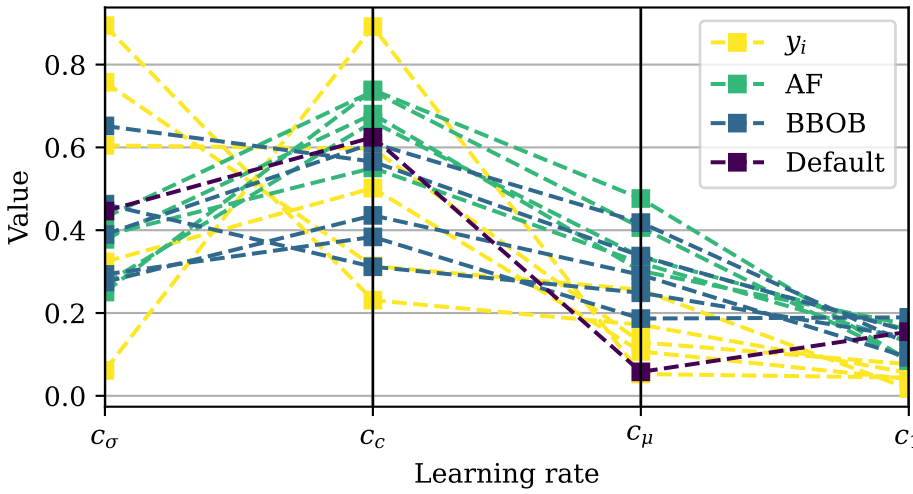

Figure 4.14: Values of the learning rates of the CMA-ES default configuration and the best configurations from five meta-optimization run on the five real-world problems y_i, the most similar artificial functions (AF) and the most similar BBOB functions.

In contrast, the optimal values for c_σ and c_c for the five real-world problems are widely dispersed throughout the feasible range of values. The values obtained from the artificial functions and the BBOB functions also exhibit significant variation. However, each configuration yields superior results compared to the default configuration. Since there is no discernible underlying trend for c_c and c_σ, the default values appear to be appropriate.

The optimal choices for the remaining parameters on real-world problems align with those identified by the brute force search (Section 4.3). A smaller number of offspring, a selection ratio of 0.5 and an increased initial standard deviation are deemed optimal. Furthermore, equal weights, no orthogonal sampling and reflection are among the best options. These clear trends are similarly captured by the artificial functions similar to the real-world problems and by the BBOB functions, with the exception of the box constraint handling method. The meta-optimization algorithm consistently selects projection for the artificial functions, while for the BBOB functions, reinitialization is chosen two out of five times instead of reflection.

4.5 Tuning for Higher Dimensional Problems

This section builds upon the method outlined in Section 4.4 and extends the analysis to higher-dimensional spaces by replacing the five two-dimensional real-world problems with five instances of the ten-dimensional Büche-Rastrigin function from the BBOB benchmark suite.

The experimental setup remains consistent with that of the previous section, with specifics provided in Section 4.5.1. The performance of the identified best configurations for the BBOB instances is then evaluated against the optimal configurations derived from the most similar artificial functions (Section 4.5.2).

4.5.1 Experimental Setup

To obtain tuning references for the five instances of the ten-dimensional Büche-Rastrigin function from the BBOB benchmark suite, a large set of artificial functions is generated, and the ELA features of these problems are computed for similarity quantification, similar to Section 4.2. The differences are explained below.

A set of 100 000 ten-dimensional artificial functions is generated. Again five instances of each artificial function are considered, as well as their negations, resulting in a total of 1 000 000 unique artificial functions. The input domain for all functions is $\mathbf{x} \in [-5, 5]^{10}$. A Sobol' sequence design [96, 121] with 2 000 samples is employed for the ELA feature calculation. The feature computation was successfully completed for 76.5% of the randomly generated artificial functions. A PCA, with a cumulative variance threshold above 0.999, reduces the feature space to 27 dimensions. For each BBOB problem, the ten most similar artificial functions in terms of their distance in feature space (Equation 4.1) are selected. This results in a set of 41 distinct functions.

The meta-optimization process adheres to the same parameter space as introduced in Section 4.4. Only the number of offspring is adjusted to accommodate the higher dimensionality. Table 4.2 lists the CMA-ES parameters and their corresponding value ranges. The new considered numbers of offspring are $\{10, 16, 22, \ldots, 40\}$. The tuning process is executed separately for the five BBOB functions and the 41 artificial functions. To evaluate the performance of each parameter configuration, 20 runs are conducted for each function. CMA-ESwM (Section 2.4.3) is selected as the meta-optimization algorithm with an evaluation budget exceeding 2 000 for each run. To ensure statistically significant results, the meta-optimization is repeated five times for each set.

4.5.2 Results

Figure 4.15 visualizes the position of the functions based on the two primary principal components derived from the PCA. The five BBOB functions are observed to be clustered within a relatively confined region of this principal component space, showing less dispersion compared to the five two-dimensional real-world problems illustrated in Figure 4.3. The randomly generated ten-dimensional artificial functions span a broad area, encompassing the space where the five instances of the Büche-Rastrigin function are situated. Thus, for ten-dimensional optimization problems similar artificial functions can also be generated to serve as tuning references.

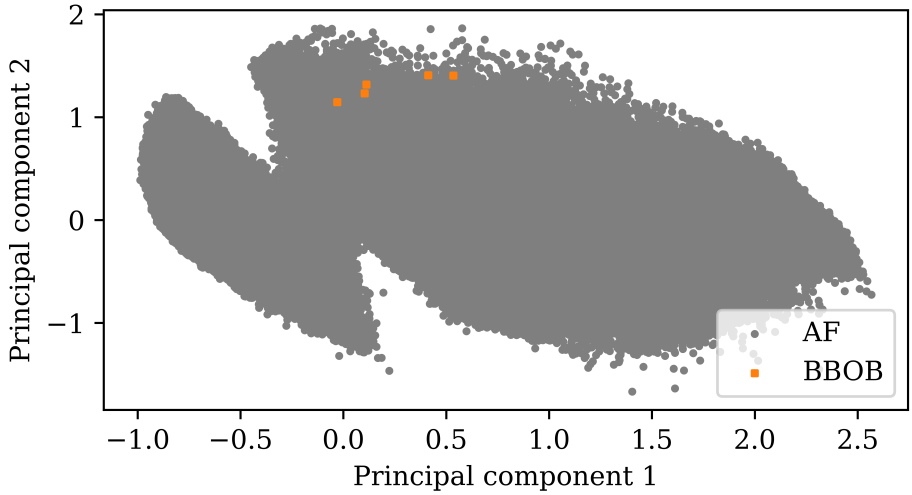

Figure 4.15: Positions defined by the two main PCA components of the 50 ELA features for each of the ten-dimensional artificial function (AF) and the five instances of the ten-dimensional Büche-Rastrigin function from the BBOB benchmark suite.

Transferring the best parameter configurations, tuned on the 41 most similar artificial functions, to the five BBOB problems yields performance improvements over the default CMA-ES parameter configuration, with gains ranging from at least 1.6% to 1.8% across the five meta-optimization runs. The average improvement is 1.7% across all five runs. In contrast, direct tuning on the BBOB problems results in performance gains between 1.7% and 2.4%, with an overall average of 2.1% across the five runs. Therefore, the proposed method attains approximately 81% of the total possible improvement for the five instances of the ten-dimensional Büche-Rastrigin function.

Figure 4.16 presents the learning rates from the optimal configurations obtained in the meta-optimization runs compared to the recommended default settings for CMA-ES. The meta-optimization runs performed on the five BBOB problems consistently identify a decrease in the learning rates for c_c and c_1 as optimal. Conversely, the optimal values for c_σ, and to a lesser extent c_μ, show a wide dispersion over the allowed range of values. This variation is also captured by the similar artificial functions, although with a smaller range for c_σ.

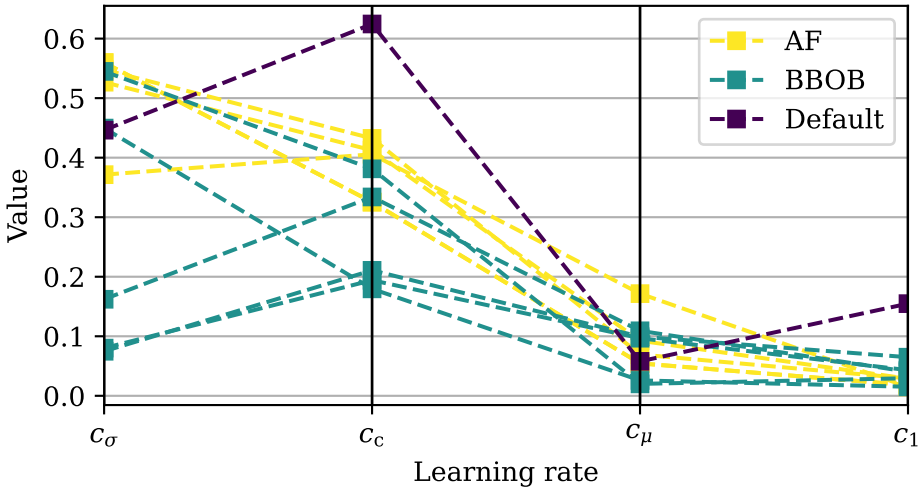

Figure 4.16: Values of the learning rates of the CMA-ES default configuration and the best configurations from five meta-optimization runs on the five instances of the ten-dimensional Büche-Rastrigin function from the BBOB benchmark suite and the most similar artificial functions (AF).

The optimal settings for the other parameters show congruence between the similar artificial functions and the BBOB problems. For example, both show that a higher number of offspring, about 22 to 26, is preferable to the default of 10, and an initial step size between 3.0 and 6.0 is more effective than the default of 2.0. The selection ratio for both sets varies between 0.33 and 0.66. Moreover, the active update of the covariance matrix is consistently chosen by the meta-optimization algorithm, while orthogonal sampling is not. For the remaining parameters, no definitive pattern emerges from the analysis of either the BBOB problems or the similar artificial functions.

4.6 Comparison of Meta-Optimization Algorithms

The tuning of CMA-ES parameters represents a meta-optimization task (Figure 2.3). In Sections 4.4 and 4.5, a specific optimization algorithm is utilized for this purpose. This section presents a comparative analysis of various meta-optimization algorithms that can also be applied to this task.

The objective of meta-optimization is to identify the optimal set of parameter values that enhance the performance of the optimization algorithm in solving the original optimization problem. The optimization of CMA-ES parameters can be viewed as a mixed-integer optimization problem. This means tuning both continuous CMA-ES parameters and various combinations of discrete parameter values and CMA-ES variants.

Several meta-algorithms have been developed. These algorithms efficiently navigate through both continuous and discrete parameter spaces, balancing exploration and exploitation to converge on a robust set of parameters that yield optimal performance for the given optimization tasks.

The Sequential Model-based Algorithm Configuration (SMAC) [62] is a well-known algorithm for parameter tuning. It uses a sequential model-based optimization (SMBO) strategy that integrates Bayesian optimization with random forest regression models to predict the performance of parameter configurations. SMAC is mainly applied in machine learning for algorithm configuration, feature selection and the search for optimal deep neural network architectures [35, 82].

Another notable SMBO algorithm is the Tree-structured Parzen Estimator (TPE) [16]. This algorithm uses a methodology based on tree-structured density estimation to identify optimal parameter settings efficiently. Employing TPE to tune CMA-ES parameters improved the performance on various benchmark optimization problems [142].

Additionally, CMA-ES itself can be used as a meta-algorithm. To address the challenge of mixed-integer optimization, the margin extension [46] for integer handling within CMA-ES can be utilized (Section 2.4.3).

The focus of this study is to investigate the efficacy of CMA-ESwM as a meta-algorithm in the context of CMA-ES parameter tuning. To this end, a series of experiments on several benchmark optimization problems are conducted. The performance of CMA-ESwM is then compared to that of SMAC, TPE and random search.

4.6.1 Experimental Setup

The effectiveness of CMA-ES is evaluated using four benchmark functions from the BBOB set (Section 2.6.1): BBOB_1, BBOB_4, BBOB_{20} and BBOB_{21}. Functions BBOB_1 and BBOB_4 are separable with a global structure, while BBOB_{20} and BBOB_{21} lack a global structure and are non-separable. BBOB_1 is unimodal, whereas BBOB_4, BBOB_{20} and BBOB_{21} are multimodal. These functions are considered in two dimensions to reduce computational effort.

Each run of CMA-ES is allocated a maximum of 400 evaluations for BBOB_1 and 2 000 for BBOB_4, BBOB_{20} and BBOB_{21}. The lower budget for BBOB_1 is due to its unimodal nature, which generally requires fewer evaluations for optimization. Four instances of each BBOB function are tested, with 25 runs per instance, resulting in a total of 100 runs for each function. The AUC is calculated from these runs to evaluate the effectiveness of a CMA-ES configuration.

Table 4.3 provides an overview of the parameters and variants of CMA-ES chosen for tuning in this study. The four learning rates c_1, c_c, c_μ and c_σ are continuous variables, the number of offspring λ is an integer, and the remaining parameters are categorical. Thus, tuning these CMA-ES parameters represents a mixed-integer optimization problem.

Table 4.3: CMA-ES parameter space for the meta-optimization.

Parameter	Description	Variants and Parameters
c_1	Learning rate rank-one update	$]0, 1]$
c_c	Learning rate adaption of C	$]0, 1]$
c_μ	Learning rate rank-μ update	$]0, 1]$
c_σ	Learning rate step size control	$]0, 1[$
λ	Number of offspring	{4,6,..,20}
μ_r	Selection ratio $\mu_\text{r} = \frac{\mu}{\lambda}$	{0.3, 0.5, 0.7}
σ_0	Initial standard deviation	{0.2, 0.4, 0.6, 0.8}
bc	Box constraint handling	{projection, reflection, wrapping, uniform reinitialization, normal reinitialization}
Active	Covariance matrix update	{on, off}
Elitism	Selection strategy	$\{(\mu, \lambda), (\mu + \lambda)\}$
Orthogonal	Orthogonal sampling	{on, off}
Mirrored	Mirrored sampling	{on, off}
Threshold	Mutation vector threshold [100]	{on, off}
Weights	Option for recombination	{logarithmic, equal, α-decay}
Restart	Restart strategy	{IPOP, BIPOP}

This study utilizes for the meta-optimization algorithms the SMAC3 implementation [82], as well as Optuna's CMA-ES sampler, TPE sampler and Random sampler [2], each with their default parameters. The evaluation budget for the meta-algorithm is 3 000, and 50 full parameter tuning runs are performed on each BBOB function for each meta-algorithm.

4.6.2 Results

Figure 4.17 shows the median performance of CMA-ES parameter configurations during the parameter tuning across 50 runs for each meta-algorithm considered, on the four BBOB functions BBOB_1, BBOB_4, BBOB_{20} and BBOB_{21}, which serve as the original optimization problems. The objective for the meta-optimization algorithm is to maximize the AUC.

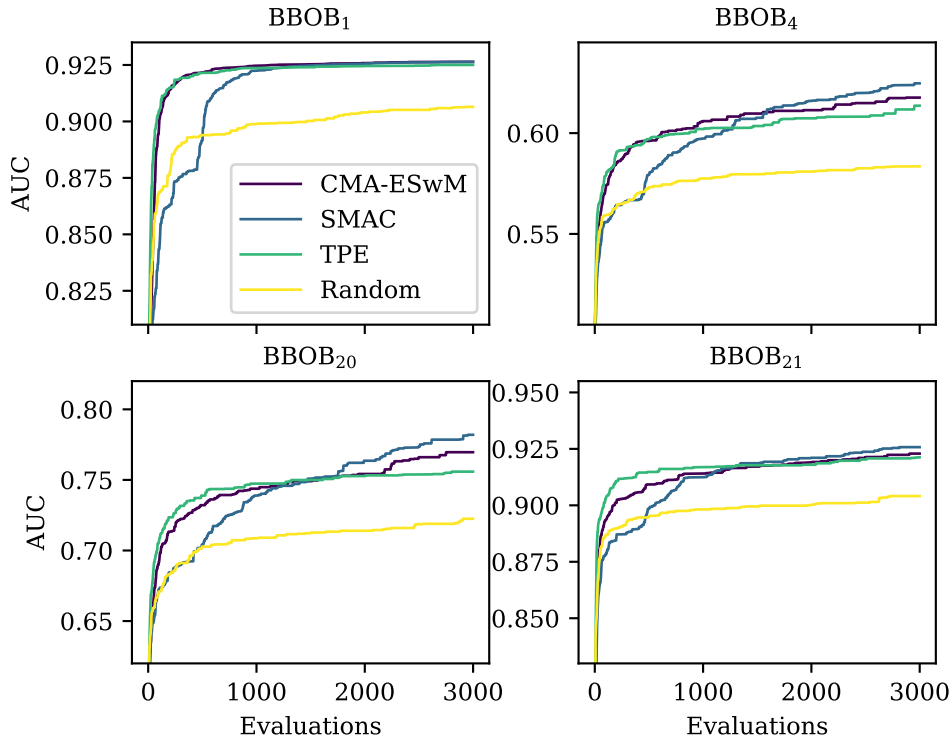

Figure 4.17: Median AUC values over evaluations of 50 runs for the four meta-optimization algorithms considered for tuning CMA-ES parameters on the four two-dimensional BBOB functions $\text{BBOB}_1, \text{BBOB}_4, \text{BBOB}_{20}, \text{BBOB}_{21}$.

For all four BBOB functions, the majority of performance improvements in CMA-ES parameters occur within the first 1 000 evaluations. CMA-ESwM and TPE show comparable performance over the course of evaluations, with TPE slightly performing better in the early stages (up to 1 000 evaluations), while CMA-ESwM tends to perform better thereafter. During the initial 1 000 evaluations, SMAC may appear to be slower in discovering high-quality solutions when compared to other algorithms. However, the performance improves over time. In the end, SMAC achieves comparable or slightly better results compared to the meta-algorithms mentioned above. In contrast, the progress of the random search significantly declines after 500 evaluations.

To verify the efficacy of the best configuration identified by a meta-algorithm, the configurations are subjected to 50 additional runs on the BBOB functions for validation purposes (Figure 4.18).

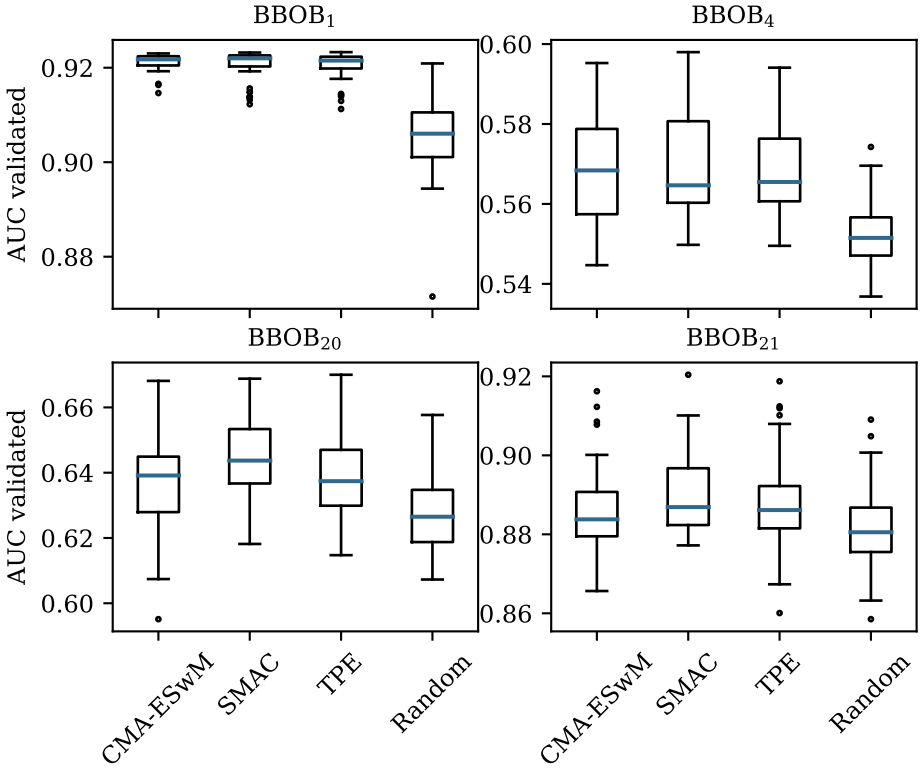

Figure 4.18: Boxplot of the validated AUC values of the best CMA-ES configurations found by the different meta-algorithms on each of the four BBOB functions considered. For each meta-algorithm and BBOB function, 50 parameter tuning runs were performed. Each configuration found in this process is, in turn, validated by 50 validation runs.

CMA-ESwM can compete with state-of-the-art algorithms like SMAC and TPE as a meta-optimization algorithm for tuning the parameters of CMA-ES. The overlap in the AUC values of the solutions identified by each meta-algorithm is notably greater than the disparities in their median values across all problems. Nevertheless, all three algorithms outperform random search. Consequently, for the selection of the meta-algorithm, additional factors, such as the wall-clock time, are required.

In terms of wall-clock time, CMA-ESwM proves to be the fastest of the trio on average. This is due to its ability to parallelize the evaluation of the population within a single generation. Consequently, evaluating a new configuration in random search incurs minimal computational overhead but still requires approximately 50% more time than CMA-ESwM to finalize a parameter tuning run when evaluations are conducted sequentially. In contrast, both SMAC and TPE take roughly two to three times longer than CMA-ESwM. The longer wall-clock time of these algorithms is not only due to their sequential evaluation processes but also to the additional internal computations and model training required. These computations remain time-consuming despite potential increases in parallelization.

In a nutshell, CMA-ES outperforms SMAC and TPE in terms of wall clock time. The reason for that is the inherent efficiency and parallelization capabilities of CMA-ES. However, a simple random search as a meta-optimization algorithm can identify very good parameter configurations. Especially when the meta-optimization can be executed fully parallel, the use of random search can significantly reduce the wall-clock time required.

4.7 Conclusion

This chapter presents a method for tuning CMA-ES parameters using a meta-optimization approach (Section 4.1). The method relies on computationally inexpensive functions similar to the original optimization problems. These similar functions are employed as tuning references. The study generates a set of artificial functions that augment the BBOB function set, thereby offering a broad array of optimization landscapes. ELA features are employed to measure the similarity between different landscapes and identify functions that closely resemble the landscapes of five two-dimensional real-world problems. Based on quantitative measures and visual comparisons, these artificial functions reflect the real-world landscapes more accurately than the BBOB functions (Section 4.2).

Through a brute force approach (Section 4.3), the patterns between the best parameter configurations on the similar functions and their real-world counterparts can be analyzed. This conducted study revealed that certain landscape properties of these similar functions remain elusive. In particular, the choice of the selection strategy within CMA-ES and the box constraint handling method for real-world applications should not rely on the performance results obtained from these similar functions.

The results from the brute-force approach described in Section 4.3 highlight the uniqueness of each real-world problem landscape. Each problem requires a tailored parameter configuration for an optimal performance of the optimization algorithm. However, due to the computational effort required to identify similar functions, tuning the parameters for each real-world problem instance is not feasible. Therefore, a single parameter configuration that outperforms the default across all five problems is identified (Section 4.4). Moreover, the tuning method is also successfully applied to the ten-dimensional Büche-Rastrigin function from the BBOB benchmark suite (Section 4.5).

Furthermore, the use of CMA-ESwM as a meta-optimization algorithm is a cost-effective strategy for parameter tuning. This approach delivers competitive results compared to established algorithms for parameter tuning, such as SMAC and TPE (Section 4.6).

Tuning the parameters of CMA-ES to specific low-level probabilities of real-world problem instances can lead to significant improvements. However, this procedure is computationally expensive. Moreover, the identified configuration is only optimal for that particular instance. An alternative is to identify an optimal general-purpose configuration that solves a wide range of problem instances of one problem class. This means tuning the algorithm to more high-level probabilities prevalent across all problem instances. Thus, a well-performing parameter configuration can be identified computationally efficiently for many problems simultaneously.

Chapter 5

Handling Discretization

Optimization problems can be characterized by whether the input variables are continuous or discrete (Section 2.1). However, in practice, the distinction between continuous and discrete variables is not as binary as it may first appear. For instance, the optimization of industrial designs must contend with the physical limits of precision, effectively introducing a level of discretization to variables that are theoretically continuous. Additionally, the computational representation of continuous variables in the form of floating-point numbers possesses finite precision.

This chapter examines the impact of variable discretization on the performance of CMA-ES. The presented results build upon [124]. A method for discretizing continuous functions is introduced (Section 5.1) and a comprehensive study is conducted to assess the effects (Section 5.3). The performance of various CMA-ES variants are compared alongside with an EA (Section 5.2) designed for discrete optimization problems. The results are presented in Section 5.4 followed by a short conclusion (Section 5.5).

5.1 Function Discretization

To investigate the impact of discretization on optimization algorithm performance a test function is required. Any continuous optimization problem can be transformed into a discrete optimization problem by limiting the set of feasible values and rounding continuous values to this finite set. However, such a transformation can significantly change the search landscape. The location of the global optimum does not remain constant in the general case.

The proposed method only considers problems for which the location of the global optimum $\mathbf{x}^*$ is known. This ensures that the location of the optimum can be adjusted after the discretization if necessary. The discretization process begins by creating a grid with a specified number of levels n_{levels} between a lower bound l and an upper bound u for each dimension d_i, resulting in evenly spaced numbers over the interval, with a distance Δ between two discrete values:

$$G_{d_i} = l + j \cdot \underbrace{\frac{u-l}{n_{\text{levels}}-1}}_{\Delta}, \quad \text{for } j = 0, 1, \ldots, n_{\text{levels}} - 1. \tag{5.1}$$

Continuous values are then rounded to the nearest feasible value on the grid. This procedure creates a modified landscape in which each discrete value lies in the center of a plateau of identical fitness. To ensure that the global optimum $\mathbf{x}^*$ is included in the discretized space, a subsequent translation is applied. Since the bounds remain fixed, this translation can cause values adjacent to the bounds to shift outside the feasible domain. To address this, all out-of-bounds values are clipped to the nearest bound. Additionally, the plateaus near the bounds may be smaller because values on one side are rounded to the nearest grid point within the bounds, while values on the other side fall outside the bounds. Figure 5.1 illustrates the discretization of an ellipsoid function in one dimension for different numbers of levels n_{levels}.

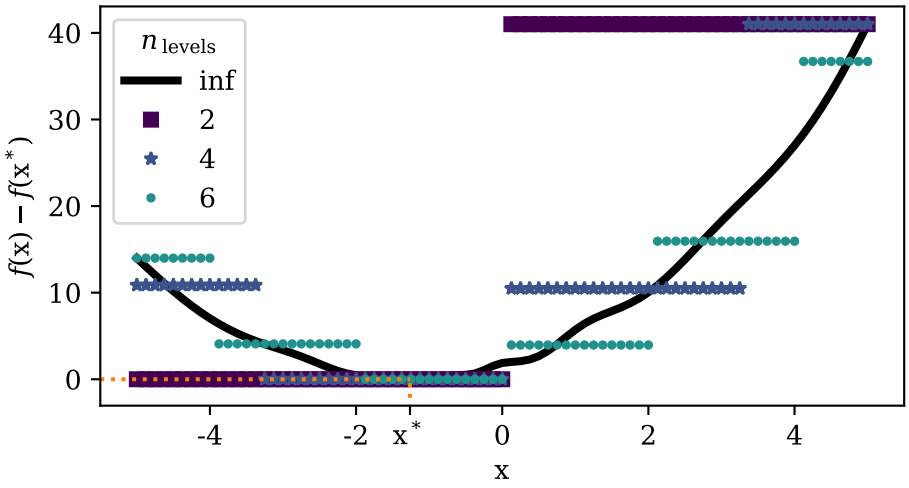

Figure 5.1: Discretized one-dimensional ellipsoid function $f(\mathbf{x})$, with $\mathbf{x} \in [-5, 5]$ for the different numbers of levels $n_{\text{levels}} \in \{\text{inf}, 2, 4, 6\}$ with the global optimum at $\mathbf{x}^*$.

Discretization alters the original landscape of an optimization problem and can thereby change its key characteristics. Figure 5.2 illustrates the effect of discretizing a two-dimensional Rosenbrock function (BBOB function F8 (Table 2.1)) for different numbers of levels. The primary challenge for optimization algorithms operating on the continuous Rosenbrock function is the need to constantly adjust search directions to navigate along a narrow, curved ridge toward the optimum.

The application of the discretization method introduces additional complexity by creating multiple local optima. The original unimodal problem is transformed into a multimodal one. These artificially induced local optima can mislead an optimization algorithm and complicate the search process. The emergence of multimodality in an originally unimodal problem due to discretization has also been reported by Tušar et al. [130]. However, this effect does not occur if the original problem is axis-parallel (such as the BBOB sphere function f_1 or ellipsoid function f_2), where one of the neighboring discrete values always improves the objective function.

Reducing the number of levels n_{levels} to a small number, such as five levels, results again in a unimodal landscape. However, this simplification comes at the cost of losing the characteristic ridge feature of the Rosenbrock function. Thus, maintaining the intricate features of a complex landscape without introducing multimodality presents a significant challenge. Conversely, if there are too few levels, these defining features are inevitably lost.

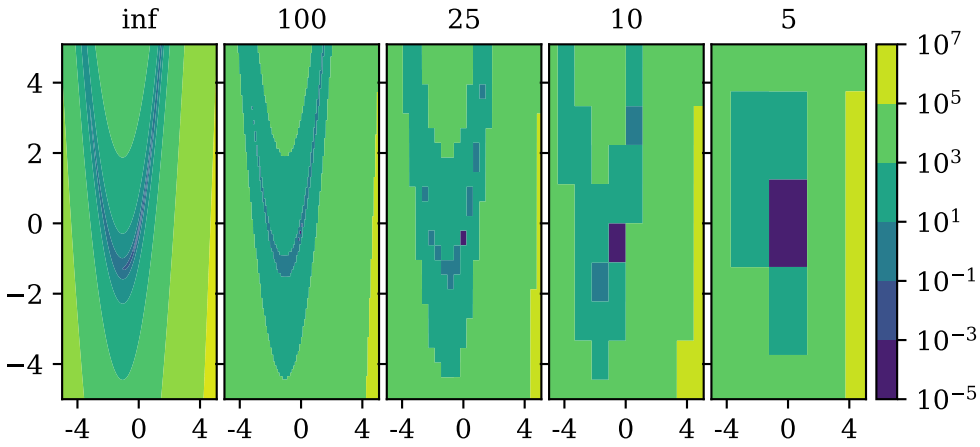

Figure 5.2: Discretized two-dimensional Rosenbrock function $f(\mathbf{x})$, with $\mathbf{x} \in [-5, 5]^2$ for the different numbers of levels $n_{\text{levels}} \in \{\text{inf}, 100, 25, 10, 5\}$. Several local optima can be created due to the discretization, clearly observable for $n_{\text{levels}} = 25$.

5.2 EA for Integer Programming

A benchmark algorithm for solving discrete optimization problems is needed to compare and assess the effects and the extent of the impact of variable discretization on the performance of CMA-ES. The Evolutionary Algorithm for Integer Programming (int-EA), devised by Rudolph in 1994 [110], represents such a specialized approach for solving optimization problems within integer domains and is described in the following.

The int-EA is characterized by employing a mutation distribution that is designed for integer search spaces, deviating from traditional continuous space evolutionary strategies. Theoretical analysis has revealed the maximum entropy mutation distribution for unbounded integer domains to be [110]:

$$p_k = \frac{p}{2-p}(1-p)^{|k|}, \tag{5.2}$$

where p_k denotes the probability of the integer k being selected. This distribution is symmetric around zero. Therefore, smaller mutations are favored, while larger variations are still possible. To generate random numbers that conform to this specialized distribution, a technique can be employed that involves the difference of two independent random variables. These variables, G_1 and G_2, are parameterized by:

$$p = 1 - m/((1+m)^{1/2} + 1), \tag{5.3}$$

where m is the deviation parameter that directly controls the mutation variance of the int-EA [110]. The deviation parameter m within int-EA is analogous to the mutation strength σ in traditional ES algorithms and is encoded as a strategy parameter, similar to σ, allowing the int-EA to self-adapt its mutation rate over time. Structurally, the int-EA closely follows the general framework of the (μ, λ)-ES, with nearly identical core mechanics.

To enable the int-EA to operate on the same discretized function, the discrete set of values is encoded as an integer space $\mathbf{z} \in [0, n_{\text{levels}}]^d \subseteq \mathbb{Z}^d$. These values are then transformed back to the original space using the formula $\mathbf{x} = l + \mathbf{z} \cdot \frac{u-l}{n_{\text{levels}}}$, where l and u denote the lower and upper bounds of the original search space, respectively. This encoding and transformation process allows both integer-based and continuous solvers to be assessed on the same discretized version of the original objective function $f(\mathbf{x})$. This facilitates a direct comparison between, e.g., the int-EA and CMA-ES.

5.3 Experimental Setup

The experimental setup is designed to study the effects of discretization on the performance of different CMA-ES variants and to compare them with the int-EA using the proposed discretization approach (Section 5.1). The BBOB sphere function f_1 and the ellipsoid function f_2 serve as the original continuous optimization problems. These problems are chosen because they are unimodal and also separable, which ensures that the discretization does not introduce multimodality. Thus, there are no local optima where the optimization algorithm can get stuck in both continuous and discretized domains.

The performance metric chosen for this study is the success rate, which measures the proportion of runs that solve the problem within a given budget. The BBOB framework defines a problem as "solved" if the distance to the optimum in the objective space, defined as $\delta_{f^*} = f(\mathbf{x}) - f(\mathbf{x}^*)$, is less than 10^{-8}.

For the experimental analysis, the first ten instances of the BBOB problems f_1 and f_2 are utilized, with 20 independent optimization runs conducted for each instance, resulting in a total of 200 runs per setting. The experiments cover a range of different numbers of levels, with $n_{\text{levels}} \in \{2, 3, 5, 10, 10^2, 10^3, 10^4, \text{inf}\}$, across dimensions $d \in \{2, 5, 10, 20, 40\}$. Each run is allocated a budget of 2 000 function evaluations per dimension, amounting to 40 settings per BBOB problem and algorithm.

The used configuration of the int-EA is for the population parameters $\mu = 30$ and $\lambda = 100$, which showed good performance. The initial values for the deviation parameter m_i mirror the initialization of σ_i in traditional ES, providing a comparable baseline for the evolutionary process. This configuration ensures that the int-EA is well-equipped to navigate the integer search space.

5.4 Results

To investigate the impact of discretization on the performance of the standard (μ, λ)-CMA-ES without integer handling, the algorithm is applied to a discretized five-dimensional sphere function with a spectrum of different numbers of levels n_{levels}. Figure 5.3 shows the evolution of the distance to the optimal function value δ_{f^*} across successive generations for 100 independent runs for each of the five considered number of levels $n_{\text{levels}} \in \{2, 10, 10^3, \text{inf}\}$.

In the absence of discretization ($n_{\text{levels}} = \text{inf}$), CMA-ES consistently solved the sphere function across all 100 runs. The distance to the optimal function value δ_{f^*} consistently declines as the algorithm progresses through generations. Upon introducing discretization (Equation 5.1), two notable patterns emerged (Figure 5.3).

First, with a finite number of levels, the runs diverged. A fraction of the runs rapidly achieves a δ_{f^*} value below 10^{-8}, indicating successful optimization, while others experienced stagnation. This divergence is attributed to the size of the region surrounding the optimum, which is of the size Δ^d. When the algorithm reaches this optimal region, the distance to the optimal function value δ_{f^*} directly falls to zero.

Second, if the optimal plateau is not reached quickly, the CMA-ES can become trapped on a suboptimal plateau, progressively reducing the step size. This lowers the probability of reaching another, potentially superior plateau. The likelihood of convergence to the optimum decreases. As the number of levels increases, these effects diminish. Nevertheless, for 1 000 levels some runs of CMA-ES still exhibits stagnation on the discretized five-dimensional sphere function.

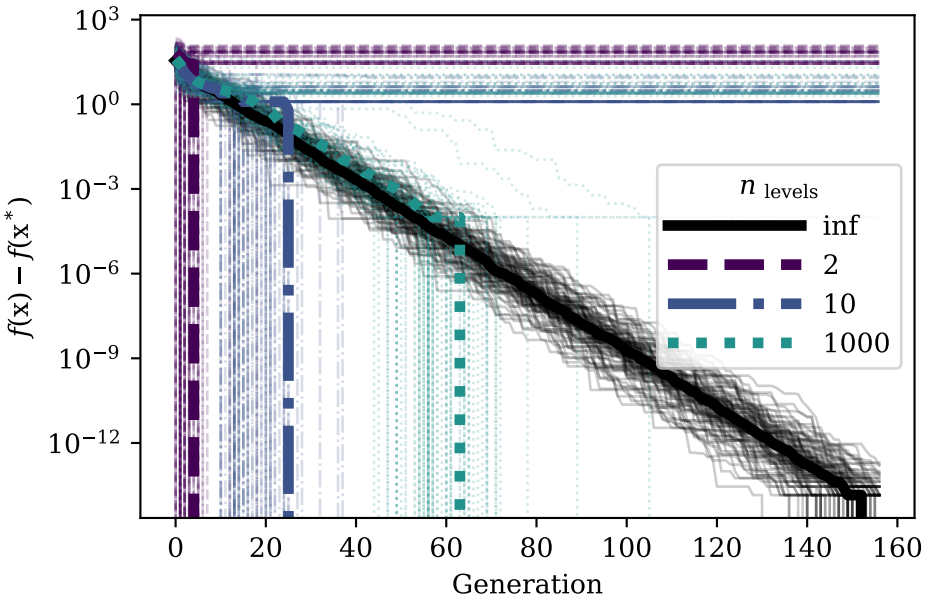

Figure 5.3: Distance to the optimal function value δ_{f^*} over generations of single CMA-ES runs on a discretized five-dimensional sphere function with a spectrum of different numbers of levels $n_{\text{levels}} \in \{2, 10, 10^3, \text{inf}\}$ for a total of 100 runs and the median (bold line) for each number of levels.

5.4.1 Success Rates

The discretization not only slows convergence but causes stagnation, stopping the convergence progress of the standard (μ, λ)-CMA-ES entirely. Therefore, the rate of successful runs emerges as a good metric for evaluating the impact of discretization on the performance of CMA-ES. The analysis commences with a comparison between the standard CMA-ES, the CMA-ESwM and the int-EA. Figure 5.4 shows the results for different numbers of levels and dimensions (Section 5.3). For comparison, in the continuous case, where $n_{\text{levels}} = \text{inf}$, all runs of CMA-ES on the two considered BBOB problems f_1 and f_2 successfully converge within the prescribed budget across all dimensions, achieving a success rate of 1.0.

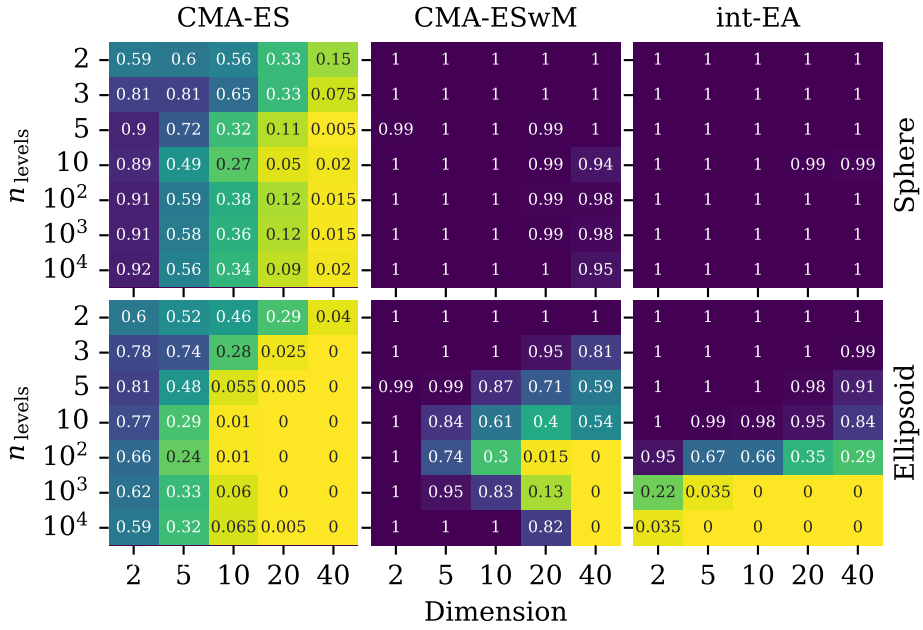

Figure 5.4: Success rates for CMA-ES, CMA-ESwM and int-EA on the sphere function (top row) and ellipsoid function (bottom row), across various dimensions (x-axis) and discretization levels n_{levels} (y-axis). Each algorithm is given a budget of 2 000 evaluations per dimension, with 200 runs conducted for each setting. The target function value is set to 10^{-8}.

For all the considered algorithms, the discretized ellipsoid function is more difficult to solve than the discretized sphere function. The sphere function is isotropic, which facilitates a more straightforward optimization process even after discretization. In contrast, the ellipsoid function is anisotropic. The variable scale varies along different axes.

The step size adaptation mechanism of CMA-ES is sensitive to the scale of the search space. The various scales along different dimensions of the ellipsoid function mean that the step size needs to be carefully balanced in order to progress. This balancing becomes increasingly complex within a discretized search landscape. Here, the step size needs to be sufficiently large to escape the plateaus in some dimensions while simultaneously being adequately restrained to permit precise, incremental advancements in others. The interplay between these requirements can significantly hinder the optimization process.

Adding a certain amount of discretization considerably increases the difficulties faced by the standard (μ, λ)-CMA-ES. The higher the dimension, the more the discretization has a negative impact on the performance of CMA-ES (Figure 5.4 left column). However, already in the two-dimensional setting, the discretization leads to a failure rate of at least around 10% for the sphere function and around at least 20% for the ellipsoid function, indicating that CMA-ES stagnates. In higher dimensions, such as 20 and particularly 40, the success rate decreases further, with almost no runs finding the solution. The extension with the margin improves the performance significantly and outperforms the standard CMA-ES on all problem settings (Figure 5.4 center column). For the sphere function, the CMA-ESwM nearly always solves the problem, regardless of the setting. However, for the ellipsoid function, CMA-ESwM struggles particularly with a moderate number of levels. The complexity of the problem amplifies with increasing dimensionality. A small number of levels simplifies the optimization task due to the enlarged region containing the optimal solution. Conversely, a large number of levels diminishes the size of the plateaus with identical function values, reducing the risk of stagnation. These effects, however, are mitigated as the dimensionality rises. For example, in the 20-dimensional space with 100 levels, the success rate decreases to a mere 0.13 on the ellipsoid function.

When applied to the sphere function, the int-EA performs comparably to the CMA-ESwM, successfully resolving the problem nearly every time, regardless of the discretization (Figure 5.4 right column). However, in the case of the ellipsoid function, particularly at moderate numbers of levels and in higher dimensions, the int-EA achieves a higher success rate than the CMA-ESwM. In settings with discretization levels above 1 000, the ability of the int-EA to find the solution diminishes significantly. Consequently, while there are scenarios where the CMA-ESwM performs worse than the int-EA, for cases involving a large number of levels, the CMA-ESwM emerges as the preferred algorithm. For a small number of levels, both int-EA and CMA-ESwM exhibit comparable performance.

To enhance the capability of the standard CMA-ES in addressing the challenges posed by discretization, three modifications are considered. A BBOB problem is defined within the search space $\mathbf{x} \in [-5, 5]^d$. Especially in higher dimensions, the probability of the initial population being generated entirely outside of this space increases. Therefore, the repair method projection is employed for box constraint handling (Section 2.4.2). This significantly improves the performance of CMA-ES (Figure 5.5 left column) by preventing stagnation beyond the given lower and upper bounds. With the box-constraint handling, on the sphere function for 5 and higher numbers of levels, the success rate approaches 1.0 across the considered dimensions, with the exception of the higher dimensions 20 and 40, where the success rate drops to approximately 0.9 and 0.7, respectively. The performance is still suboptimal for a very small number of levels, such as two and three. For the ellipsoid function, CMA-ES with box constraint handling continues to face challenges, particularly in solving problems with a moderate number of levels.

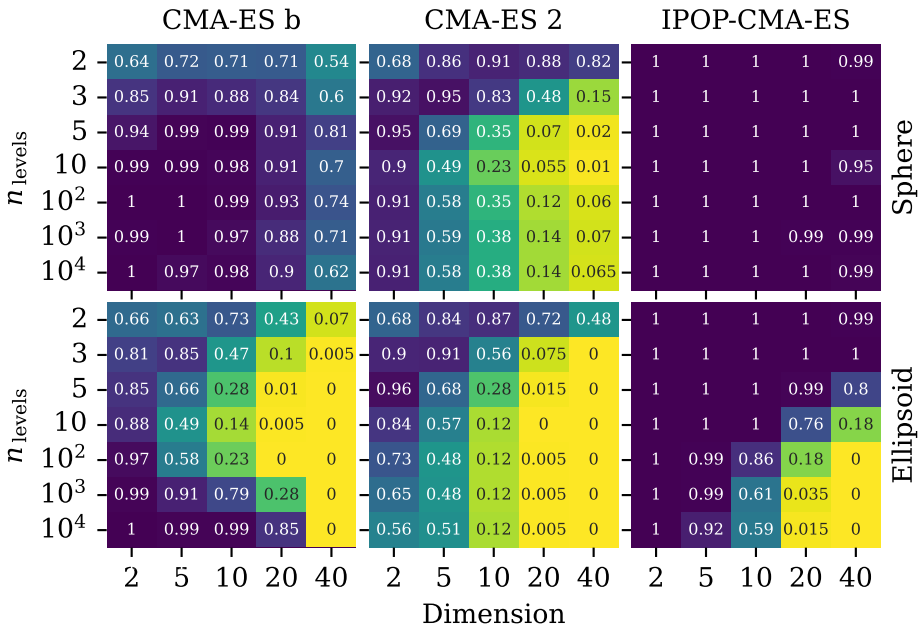

Figure 5.5: Success rates for CMA-ES with box-constraint handling (CMA-ES b), CMA-ES with a doubled default population size (CMA-ES 2) and CMA-ES with an IPOP restart strategy (IPOP-CMA-ES) on the sphere function (top row) and ellipsoid function (bottom row), across various dimensions (x-axis) and discretization levels n_{levels} (y-axis). Each algorithm is given a budget of 2 000 evaluations per dimension, with 200 runs conducted for each setting. The target function value is set to 10^{-8}.

Doubling the default population size of CMA-ES generally improves performance across all considered settings (Figure 5.5 center column). The primary advantage of a larger population is the increased probability of sampling individuals outside of the plateaus created by discretization. This enhancement enables CMA-ES to escape plateaus, thus mitigating the risk of stagnation more effectively. However, despite these improvements, stagnation of the optimization progress does occur in some runs across all settings, with a higher incidence in scenarios involving higher dimensions and larger numbers of levels.

The adoption of a restart strategy with an increasing population size affords CMA-ES the opportunity for additional runs in the event of early stagnation, provided that the evaluation budget has not been fully expended. This approach achieves a success rate approaching 1.0 in the majority of settings considered (Figure 5.5 right column). However, while a restart is potentially beneficial, a restart does not inherently preclude the possibility of subsequent stagnation. Notably, in the case of the 40-dimensional ellipsoid function and 100 or more levels, the majority of runs still fail to find a solution and continue to stagnate. Therefore, employing a restart strategy should be considered a measure of last resort.

To address the low success rates of the CMA-ESwM on the ellipsoid function with a moderate number of levels, both an increased population size and a restart strategy are applied to the CMA-ESwM (Figure 5.6). At first, the combination of a larger population size and the margin extension (CMA-ESwM) synergistically enhances the capacity of CMA-ES to escape the plateaus. As a result, doubling the default population size of the CMA-ESwM leads to a significant increase in success rates, particularly for the more challenging settings in 20 dimensions with 5 or more levels. However, particularly for 10 levels, the int-EA still slightly outperforms CMA-ESwM (Figure 5.6 left column).

The addition of a restart strategy significantly improves the performance of CMA-ESwM, which is now able to solve all problem settings except in cases with 100 or more levels in 20 and 40 dimensions, where the success rate does not reach 1.0 (Figure 5.6 center column). When the restart strategy is combined with the increased population size, CMA-ESwM consistently finds the solution across all considered settings, with the sole exception of the three settings with 100 or more levels in 40 dimensions on the ellipsoid function (Figure 5.6 right column). It is important to note that in these particular settings, the int-EA does not exhibit superior performance either (Figure 5.4 right column).

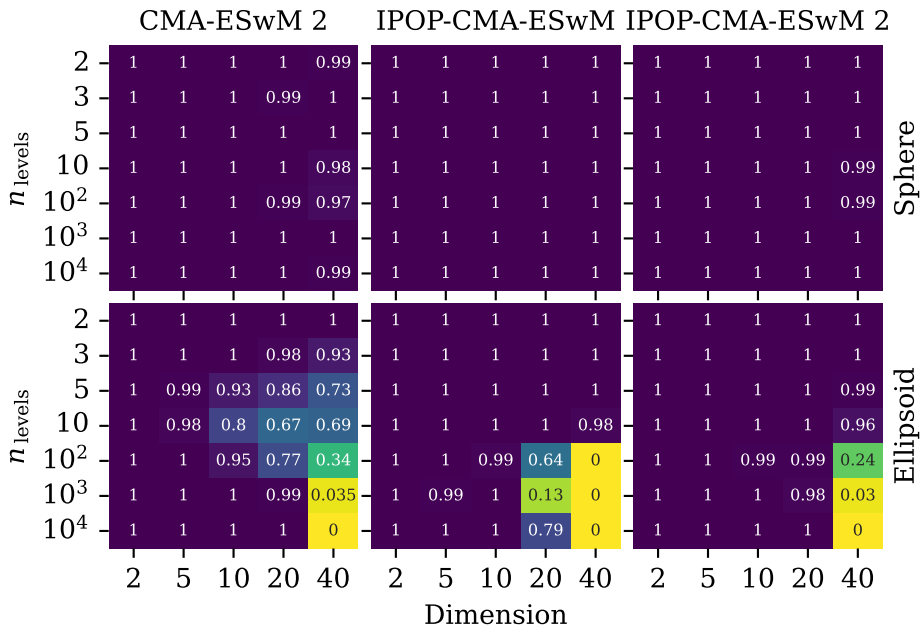

Figure 5.6: Success rates for CMA-ESwM with a doubled default population size (CMA-ESwM 2), CMA-ESwM with an IPOP restart strategy (IPOP-CMA-ESwM) and CMA-ESwM with an IPOP restart strategy and a doubled default population size (IPOP-CMA-ESwM 2) on the sphere function (top row) and ellipsoid function (bottom row), across various dimensions (x-axis) and discretization levels n_{levels} (y-axis). Each algorithm is given a budget of 2 000 evaluations per dimension, with 200 runs conducted for each setting. The target function value is set to 10^{-8}.

5.4.2 ECDF

So far, the success rate has only been considered after a predefined budget of evaluations. To assess the convergence performance over time, success rates can be illustrated with the ECDF over the complete evaluation process. The performance of IPOP-CMA-ESwM with a doubled population size on the discretized 10-dimensional ellipsoid function is compared with the standard CMA-ES on the continuous 10-dimensional ellipsoid function (Figure 5.7).

The ECDF for the IPOP-CMA-ESwM on the discretized ellipsoid function closely resembles that of the CMA-ES on the continuous ellipsoid function in shape. All runs achieve the target threshold of 10^{-8} within the allocated budget of 20 000 evaluations, and practically no runs stagnated. However, the ECDF curves reveal differences in the number of evaluations required to attain specific success rates.

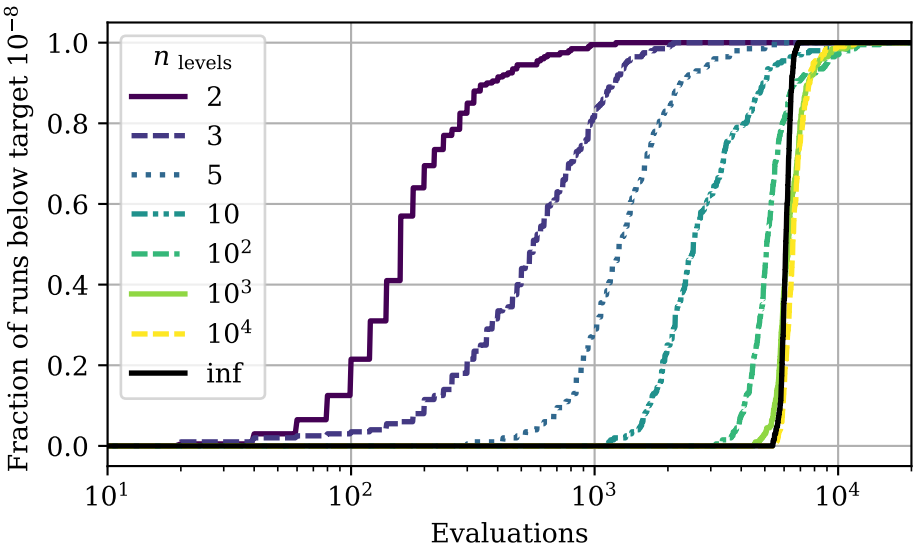

Figure 5.7: Empirical Cumulative Distribution Function for the IPOP-CMA-ESwM with a doubled default population size on the 10 dimensional ellipsoid function for $n_{\text{levels}} \in \{2, 3, 5, 10, 10^2, 10^3, 10^4\}$ and the standard CMA-ES for $n_{\text{levels}} \in \{\text{inf}\}$. The target function value is set to 10^{-8} and 200 runs are conducted for each setting.

For settings with 10 or fewer levels, the number of evaluations required to achieve a given success rate decreases with fewer levels and is consistently lower than that required for the standard CMA-ES on the continuous function. Therefore, CMA-ESwM seems to benefit from discretization, transforming the negative effects of discretization on standard CMA-ES performance into a positive outcome when the margin extension is incorporated into CMA-ES.

In the setting with 100 levels, only for the last 10 percent, as the success rate approaches 1.0, the number of evaluations required by the IPOP-CMA-ESwM exceeds that of the standard CMA-ES on the continuous function. This increase in required evaluations can be attributed to the need for a restart to achieve a success rate of 1.0. Without such a restart, CMA-ESwM achieves only a success rate of 0.95 (Figure 5.6 left column).

For 1 000 and 10 000 levels, the ECDF curves closely align with the continuous case. However, particularly towards the latter stages of optimization, CMA-ESwM requires a slightly higher number of evaluations to achieve a success rate of 1.0 across all runs compared to the CMA-ES on the continuous function. This observation means that discretization can slow down convergence somewhat but does not prevent the

algorithm from being successful. Remarkably, for both the 1 000 and 10 000 levels, a success rate of 1.0 is achieved with a reduced number of evaluations relative to the setting with 100 levels, which implies that a higher number of levels aids in facilitating convergence. Additionally, the absence of stagnation in individual runs indicates that a restart strategy is not required for these settings to attain a success rate of 1.0 (Figure 5.6 left column).

For the 20-dimensional discretized ellipsoid function, the ECDF curves show a pattern similar to those observed in the 10-dimensional settings (Figure 5.8). Notably, for 10 levels, the ECDF curve displays a slight inflection at a success rate of 0.4, suggesting that restarts have been conducted. In the absence of restarts, the success rate for CMA-ESwM is 0.67 (Figure 5.6 left column). For 100 levels, the success rate without restarts increases to 0.77, the inflection in the ECDF curve occurs later, at a higher success rate. This suggests a slower convergence speed compared CMA-ES on the continuous ellipsoid function. Similarly, at 1 000 and 10 000 levels, the convergence speed is slightly slower than that observed in the 10-dimensional case when benchmarked against CMA-ES on the continuous ellipsoid function.

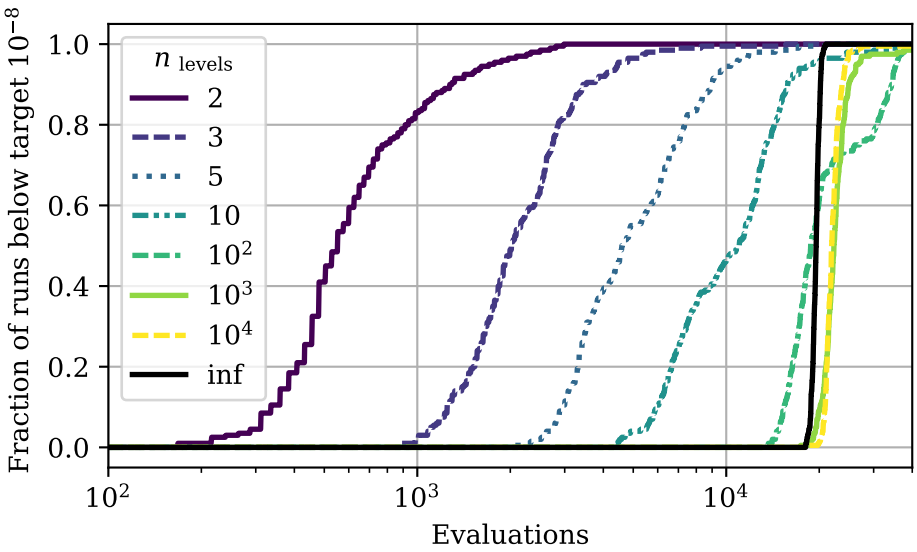

Figure 5.8: Empirical Cumulative Distribution Function for the IPOP-CMA-ESwM with a doubled default population size on the 20 dimensional ellipsoid function for $n_{\text{levels}} \in \{2, 3, 5, 10, 10^2, 10^3, 10^4\}$ and the standard CMA-ES for $n_{\text{levels}} \in \{\text{inf}\}$. The target function value is set to 10^{-8} and 200 runs are conducted for each setting.

5.5 Conclusion

This chapter introduced a method for discretizing continuous optimization problems by creating a grid with a specified number of discrete levels per dimension. The discretized problems are then used to assess the impact of different levels of discretization on the performance of an optimization algorithm. The analysis showed that the discretized ellipsoid function is more difficult to solve than the discretized sphere function.

The standard CMA-ES struggles with discretized spaces. This issue is addressed by employing CMA-ESwM for integer handling. This variant improves the performance on the sphere function and on the ellipsoid function across all levels.

The int-EA, here used as a benchmark algorithm, outperforms CMA-ESwM on the ellipsoid function for moderate numbers of discrete levels and particularly in higher dimensions. However, for higher numbers of levels, the int-EA cannot compete with the CMA-ESwM on the ellipsoid function across all considered dimensions. The combination of CMA-ESwM with the IPOP restart strategy and an increased initial population size outperforms the int-EA in all considered settings.

The convergence speed of the modified CMA-ESwM compared to the standard CMA-ES on the continuous ellipsoid function is not significantly impeded across the majority of considered settings. In fact, the discretization with small numbers of levels simplifies the difficulty of the problem, and convergence of the CMA-ESwM is accelerated. Nonetheless, for 100 levels, the convergence speed of CMA-ESwM is, on average, slightly reduced towards the end of the optimization process because of the need for restarts, particularly in higher dimensions such as 20.

In the 40-dimensional setting with a higher number of levels, neither the IPOP-CMA-ESwM with an increased population size nor int-EA managed to solve these settings effectively. Therefore, to solve these settings, another strategy or optimization algorithm is required. However, up to 20 dimensions, CMA-ESwM proved to be a robust optimization algorithm for wide numbers of discrete levels, from binary optimization problems with just two levels to settings with over 10 000 levels.

In conclusion, the results from this chapter reveal the capability of CMA-ESwM to address discretized optimization problems across diverse dimensions and numbers of discrete levels. The use of a restart strategy and an increased population size is highly recommended.

Chapter 6

Uncertainty Quantification

Optimization algorithms, such as CMA-ES, are designed to minimize an objective function by selecting the top-μ individuals from a population (Section 2.4). However, due to measurement errors, external disturbances or the inherent stochastic nature of the system under consideration, real-world optimization problems often exhibit noise or non-deterministic characteristics. This means that evaluating the objective with identical inputs can yield different results. Consequently, selecting the top-μ individuals based solely on a single evaluation of the objective function per individual can lead to inaccuracies. Section 2.7 provides a comprehensive overview of uncertainty quantification and allocation strategies to increase the certainty of this selection process.

This chapter introduces a novel Dynamic Allocation (DA) methodology that integrates uncertainty quantification with CMA-ES [50, 51] to mitigate the uncertainty in selecting the top-μ individuals in each generation (Section 6.2). The uncertainty quantification method utilized is based on the recently introduced concepts of Confidence Interval Sequences (CISs) (Section 2.7.3) and Rank Intervals (RIs) (Section 2.7.4). First, an artificial objective function characterized by inherent noise is developed to facilitate the design and testing of the proposed methodology (Section 6.1). The analysis evaluates the performance of this methodology and compares it to Static Allocation (SA) schemes within the CMA-ES algorithm. Finally, the methodology is applied to a real-world problem that exhibits noise, demonstrating its practical utility (Section 6.3).

6.1 Noisy Test Function

The investigation and analysis of various DA methods with minimal computational resources require test functions that are inexpensive to evaluate. To this end, a normalized ellipsoid function is defined, characterized by the dimensionality d and a distinct weight w_j for each dimension:

$$f(\mathbf{x}) = \frac{\sum_j^d w_j \, x_j^2}{\sum_j^d w_j}, \quad \text{with } w_j \in \{1, 2, \ldots, d\}. \tag{6.1}$$

To introduce asymmetry into the landscape, each x_j is transformed according to the following equation:

$$x_j = \begin{cases} x_j, & \text{if } x_j \leq 0 \\ (1 + p_{\text{skew}}) x_j, & \text{if } x_j > 0 \end{cases}, \tag{6.2}$$

where p_{skew} governs the degree of skewness in the objective landscape. The landscape reverts to that of the original ellipsoid function for $p_{\text{skew}} = 0$.

The black-box nature of many real-world optimization problems necessitates minimizing assumptions about the distribution of objective function values across repeated evaluations. Therefore, this thesis adopts a frequentist perspective. Each function evaluation corresponds to a specific scenario z_i, which is randomly sampled without replacement from the set of all possible scenarios Ω_Z. After evaluating all possible scenarios, the sample mean is identical to the true mean of the objective function (Equation 2.23).

For a given input $\mathbf{x}_k$, a set of $|\Omega_Z|$ values is sampled from a normal distribution with standard deviation σ_Z representing the scenarios z_i. After being sampled, this set of $|\Omega_Z|$ values remains fixed and is unique to each input $\mathbf{x}_k$. Furthermore, the mean across all $|\Omega_Z|$ values is subtracted from each individual value:

$$n(\mathbf{x}_k, z_i) := z_i - \frac{1}{|\Omega_Z|} \sum_{i=1}^{|\Omega_Z|} z_i, \quad \text{where } \Omega_Z := \left\{ z_1, \ldots, z_{|\Omega_Z|} \;\middle|\; z_i \overset{s}{\sim} \sigma_Z \, \mathcal{N}(0, \mathbf{I}) \right\}, \tag{6.3}$$

This ensures that the sample mean over all possible scenarios Ω_Z is equal to the true mean, which is always zero for each input $\mathbf{x}_k$:

$$\mathrm{E}_Z \left[n(\mathbf{x}_k, Z) \right] = \hat{\mathrm{E}}_Z \left[n(\mathbf{x}_k, Z) \right] = \frac{1}{|\Omega_Z|} \sum_{i=1}^{|\Omega_Z|} n(\mathbf{x}_k, Z) = 0. \tag{6.4}$$

Equation 6.3 defines the values used to model both additive and proportional noise within the test function. The parameter r_{noise} governs the balance between these two noise types. Consequently, for a given input $\mathbf{x}_k$ and scenario z_i, the noisy test function $f(\mathbf{x}_k, z_i)$ can be expressed as follows:

$$f(\mathbf{x}_k, z_i) := f(\mathbf{x}_k) \cdot (1 + r_{\text{noise}} \cdot n(z_i) + (1 - r_{\text{noise}}) \cdot n(z_i). \tag{6.5}$$

The test function is with no noise original unimodal. The multimodality results only from the noise introduced. Therefore, if an optimization algorithm fails to locate the global optimum, the failure cannot be attributed to local optima inherent to the original function. When all possible scenarios are sampled, the noisy test function converges to the original function.

The noisy test function employed in this chapter is defined with $p_{\text{skew}} = 0.5$, $r_{\text{noise}} = 0.7$ and $|\Omega_Z| = 50$. Figure 6.1 illustrates the one-dimensional noisy test function with $\sigma_Z = 0.5$ and different numbers of evaluated scenarios.

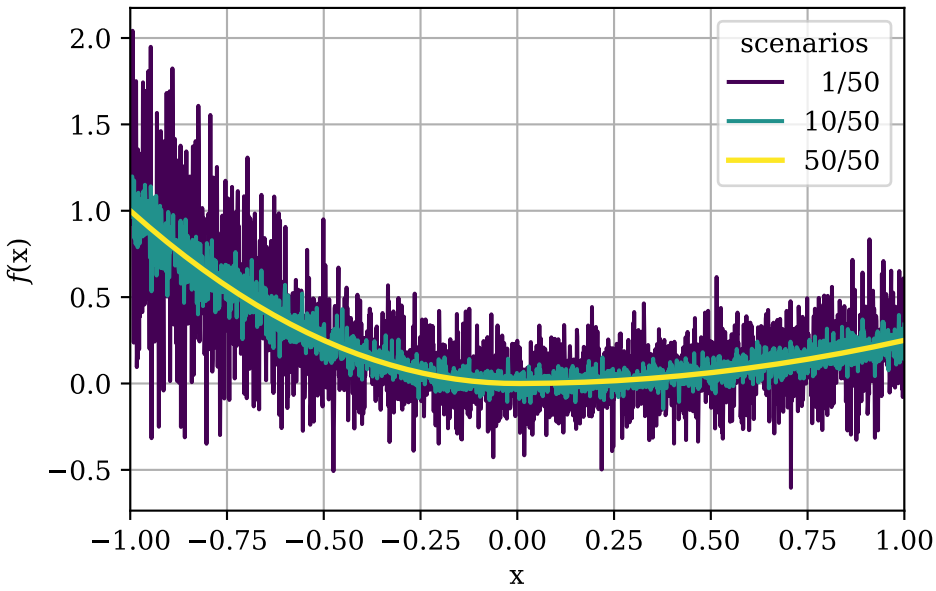

Figure 6.1: Mean values across 1, 10 and 50 evaluated scenarios on the noisy test function (Equation 6.5). Due to Equation 6.4, the noisy test function is equivalent to the original function as defined by Equation 6.1, when all 50 possible scenarios are evaluated. The degree of skewness p_{skew} is set to 0.5 (Equation 6.2). The standard deviation σ_Z of the normally distributed noise is set to 0.5 (Equation 6.3) and split to proportional and additional noise with the ratio $r_{\text{noise}} = 0.7$.

6.2 Methodology

The goal of the proposed methodology is to reliably identify the top-μ individuals within a population of λ individuals while minimizing the number of evaluations required. To achieve this, a DA methodology is proposed.

Algorithm 2 outlines the process of the proposed DA methodology. First, each individual $\mathbf{x}_k$ is evaluated on the objective function $f(\mathbf{x}_k, z_i)$ for n_0 scenarios z_i, which are sampled WoR from the set of all possible scenarios Ω_Z. Scenarios are continuously allocated WoR to individuals until either the uncertainty in the selection of the top-μ individuals, quantified by the $UQiS_\mu$ value (Equation 2.41), falls below a predefined threshold u_{thr} or the maximum number of evaluations N is reached. In each iteration, the CISs CIS_k and RIs RI_k of each individual $\mathbf{x}_k$ are computed based on the until then determined K_k objective function values $f(\mathbf{x}_k, z_i)$ of each individual $\mathbf{x}_k$. In order to apply the CISs introduced in Section 2.7.3, the objective function values are min-max scaled to $[0, 1]$ with the provided lower bounds lb and upper bounds ub. The $UQiS_\mu$ value is computed from the RIs. If the $UQiS_\mu$ value is not below the threshold u_{thr}, a DA policy $\pi_{\text{top-}\mu}$ allocates n_{allocate} more scenarios to the individuals. Each individual $\mathbf{x}_k$ is then evaluated with the scenarios z_i allocated to that individual, and K_k is adjusted accordingly. Finally, the sample mean across the K_k evaluated scenarios $\hat{\mathrm{E}}_Z\left[f(\mathbf{x}_k, Z)\right]$ is returned for each individual.

Algorithm 2 Proposed DA methodology to minimize UQiS.

Require: $\lambda, \{\mathbf{x}_k\}_{k=1}^{\lambda}, \mu, |\Omega_Z|, n_{\text{allocate}}, u_{\text{thr}}, \alpha, lb, ub$
  $n_0 \leftarrow 1$
  $UQiS_\mu \leftarrow \inf$
  Evaluate all individuals $\{\mathbf{x}_k\}_{k=1}^{\lambda}$ for n_0 scenarios z_k ▷ sampled WoR from Ω_Z
  $n \leftarrow \lambda \cdot n_0$
  $N \leftarrow \lambda \cdot |\Omega_Z|$
  while $n \leq N - n_{\text{allocate}}$ and $UQiS_\mu > u_{\text{thr}}$ **do**
    Compute CIS_k ▷ according to Equation 2.29 and 2.33
    Compute RI_k and $UQiS_\mu$ ▷ according to Equation 2.39 and 2.41
    if $UQiS_\mu > u_{\text{thr}}$ **then**
      Allocate n_{allocate} scenarios ▷ according to DA policy $\pi_{\text{top-}\mu}$ (Sections 6.2.1)
      Evaluate individuals with allocated scenarios
      Adjust K_k ▷ according to the number of allocated scenarios to $\mathbf{x}_k$
      $n \leftarrow \sum_{k=1}^{\lambda} K_k$ ▷ total number of allocated scenarios
    end if
  end while
  return $\hat{\mathrm{E}}_Z\left[f(\mathbf{x}_k, Z)\right] = \frac{1}{K_k} \sum_{i=1}^{K_k} f(\mathbf{x}_k, z_i)$

6.2.1 Dynamic Allocation Policy

A DA policy proposed by Ellmaier [32] for the top-μ selection task is utilized. This DA policy, denoted as $\pi_{\text{top-}\mu}$, allocates additional scenarios to specific individuals based on the computed RIs RI_k of each individual $\mathbf{x}_k$. For the top-μ selection task, only those individuals need to be considered that, according to their RI, are neither clearly among the top-μ nor the bottom-(λ-μ) subset.

$$\Lambda_\mu := \left\{ k \in \{1, \ldots, \lambda\} \;\middle|\; RI_k\{1, \ldots, \mu\} \text{ and } RI_k\{\mu+1, \ldots, \lambda\} \right\}, \tag{6.6}$$

consist of the indices of these individuals. The excess of the RI RI_k of individual $\mathbf{x}_k$, denoted as ξ_k, is defined as follows:

$$\xi_k := \begin{cases} \max(RI_k) - \mu, & \text{if } \hat{\text{rank}}_{\text{frq}}(\mathbf{x}_k) \leq \mu \\ \mu + 1 - \min(RI_k), & \text{else} \end{cases}. \tag{6.7}$$

Figure 6.2 exemplarily shows the excesses of the RI ξ_k for the selection of the top-3 individuals from a population of six individuals. According to their RIs all individuals can be among the top-3 individuals. Thus, the $UQiS_3$ equals three, and selecting the top-3 individuals is ambiguous. If all the excesses of the RIs ξ_k are zero, the $UQiS_\mu$ value is zero, and the selection of the top-μ individuals is unambiguous.

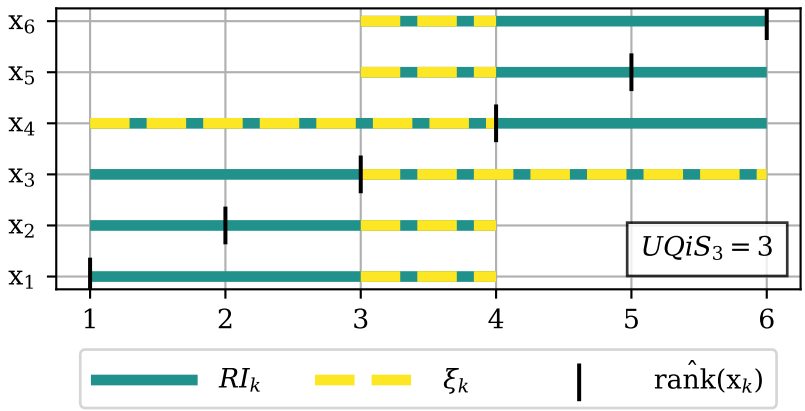

Figure 6.2: Given the RIs RI_k for each individual in $\{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$ the excess of the RIs ξ_k computed according to Equation 6.7 is illustrated for the selection of the top-3 individuals. According to the RIs all individuals can be among the top-3 individuals, and therefore the $UQiS_3$ equals three.

Algorithm 3 outlines the process of the proposed DA policy $\pi_{\text{top-}\mu}$. In each iteration, the policy can allocate n_{allocate} additional scenarios among the $|\Lambda_\mu|$ individuals that play a pivotal role in differentiating the top-μ individuals from the rest of the population according to Equation 6.6. The policy prioritizes individuals based on the excesses of their RIs ξ_k. This assumes that the magnitude of this excess is indicative of the uncertainty in the selection process. Then, the policy allocates additional objective values to these individuals to thereby narrow the width of their CIs and, consequently, their RIs.

To distribute more than one scenario per iteration in proportion to the magnitude of the excesses of the RIs, the policy employs the Imperiali method. The Imperiali method [39] originally designed to allocate parliamentary seats to political parties based on their proportion of votes, is repurposed here to allocate scenarios. With the Imperiali method, an integer number of scenarios $K_{\text{allocated},i}$ is allocated to each of the $|\Lambda_\mu|$ individuals not clearly assigned to either the top-μ or bottom-(λ-μ) subset. Thereby, the sum of the allocated scenarios $K_{\text{allocated},i}$ to the individuals is equal to the total number of allocated scenarios:

$$n_{\text{allocate}} = \sum_{k}^{\lambda} K_{\text{allocated},k}. \tag{6.8}$$

The allocation is based on the proportional excess of the RI RI_k of individual $\mathbf{x}_k$ in percent of the total excess of the RIs of the $|\Lambda_\mu|$ individuals, denoted as $\xi_{\text{percent},k}$:

$$\xi_{\text{percent},k} := \frac{100 \cdot \xi_k}{\sum_{j \in \Lambda_\mu} \xi_j}. \tag{6.9}$$

Algorithm 3 DA policy $\pi_{\text{top-}\mu}$ to allocate n_{allocate} additional scenarios [32].

Require: $\lambda, \{RI_k\}_{k=1}^{\lambda}, \mu, n_{\text{allocate}}$
Determine Λ_μ ▷ according to Equation 6.6
Compute ξ_k ▷ according to Equation 6.7
Compute $\xi_{\text{percent},k}$ ▷ according to Equation 6.9
$\{K_{\text{allocated},k}\}_{k=1}^{\lambda} \leftarrow \text{Imperiali}(\text{totalseats} = n_{\text{allocate}}, \text{percentages} = \xi_{\text{percent},k})$
return $\{K_{\text{allocated},k}\}_{k=1}^{\lambda}$

If the excess of the RIs of two individuals is equal, the width of their CIs is considered as a secondary attribute for the allocation. Apart from the proportional excess of the RIs, other metrics based on, for example, the excess of the CIs can be used as an alternative.

6.2.2 Top-μ Selection and Ranking

The mechanism of the DA methodology in selecting the top-μ individuals from a population, as outlined in Algorithm 2, is exemplarily demonstrated by utilizing the noisy test function defined in Section 6.1 with a dimensionality of two and a standard deviation σ_Z of 0.5. A population is generated by randomly sampling six individuals $\{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$ from a uniform distribution within the input space $[-1, 1]^2$. In each iteration, six additional scenarios can be allocated to the individuals, with a total limit of 50 scenarios per individual. The allocation is stopped when the UQiS of the top-μ individuals $UQiS_\mu$ is zero. The significance level α for the computation of the CISs is set to 0.3. Furthermore, for the uncertainty quantification, the lower bound lb and upper bound ub are set to the true minimum and maximum possible objective function values, respectively, by evaluating all possible scenarios for each individual in advance. The provision of broader lower and upper bounds does not affect the core principles of the qualitative findings that emerge from the results presented with the true lower and upper bounds. Table 6.1 summarizes the chosen hyperparameters of Algorithm 2 for the top-μ selection task and the utilized noisy test function.

Table 6.1: Chosen hyperparameters of the utilized noisy test function (Section 6.1) and of the DA (Algorithm 2) for the top-μ selection task.

Component	Parameter Description	Parameter Value
Noisy Test Function	Dimensionality	2
	Standard deviation σ_Z	0.5
	Degree of skewness p_{skew}	0.5
	Ratio proportional noise r_{noise}	0.7
	Possible scenarios Ω_Z	$\{z_1, \ldots, z_{50}\}$
DA	Population X	$\{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$
	Significance level α	0.3
	UQiS of top-μ individuals	μ
	UQiS threshold u_{thr}	0
	Allocations per iteration n_{allocate}	6
	Lower bound lb	$\min_{\mathbf{x}_k \in X, z_i \in \Omega_Z}(f(\mathbf{x}_k, z_i))$
	Upper bound ub	$\max_{\mathbf{x}_k \in X, z_i \in \Omega_Z}(f(\mathbf{x}_k, z_i))$

First, the task of selecting the top-1 individual out of the population of the six individuals $\{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$ is examined. Figure 6.3 illustrates the state after the final iteration of the DA by Algorithm 2, showcasing the determined objective function values $f(\mathbf{x}_k, z_i)$, along with the computed CIs CI_k, RIs RI_k and the resultant UQiS of the top-1 individual $UQiS_1$.

A total of 114 scenarios are allocated across all individuals. The top-1 individual $\mathbf{x}_1$ is evaluated on all 50 scenarios, which narrows the CI for this individual to a width of zero. The number of scenarios allocated to an individual K_k decreases along with the rank of the individual. This allocation pattern arises because the employed DA policy $\pi_{\text{top-}\mu}$ (Algorithm 3) stops allocating additional scenarios to an individual once this individual can be definitively assigned to the top-1 or bottom-5 subset based on the RIs RI_k. This trend is especially noticeable for the individuals in the lower ranks, which can be assigned to the bottom-5 subset with only a few scenarios, resulting in comparatively large CIs. However, in the final iteration, the CIs of the individuals in the bottom-5 subset are tight enough to avoid overlapping with the CI of the top-1 individual. Thus, the UQiS of the top-1 individual $UQiS_1$ is zero, and the selection of the top-1 individual is clear.

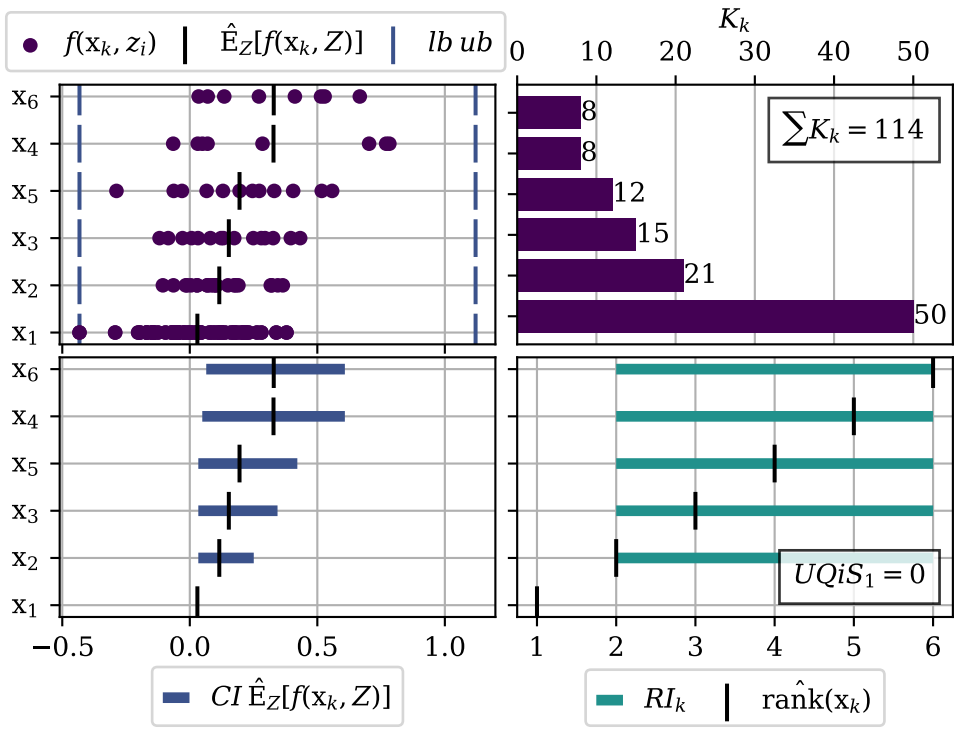

Figure 6.3: State after the final iteration of the DA by Algorithm 2 with hyperparameters of Table 6.1 for the selection of the top-1 individual from the population $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$. *Top Left*: Determined objective function values $f(\mathbf{x}_k, z_i)$, sample means and given bounds. *Top Right*: Number of allocated scenarios K_k sampled WoR from the 50 possible scenarios. *Bottom Left*: Computed CIs from the determined objective function values and sample means. *Bottom Right*: Resulting RIs, sample ranks and UQiS of the top-1 individual $UQiS_1$.

Subsequently, the task of selecting the top-3 individuals from the population of the six individuals $\{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$ is considered. Figure 6.4 illustrates the final state. A total of 180 scenarios are allocated across the population, requiring 66 additional evaluations compared to selecting only the top-1 individual. Moreover, a significant shift in the allocation pattern is observed. The number of scenarios allocated to individuals in the middle ranks is increased. This is necessary to narrow the CIs for an unambiguous assignment of these individuals to either the top-3 or bottom-3 subsets. In comparison, the individuals at the extreme ranks need fewer allocations for an unambiguous assignment. In the final iteration, the RIs of individuals in the top-3 subset do not overlap with those in the bottom-3 subset. Therefore, the selection of the top-3 individuals is unambiguous, and the UQiS of the top-3 individuals $UQiS_3$ is zero.

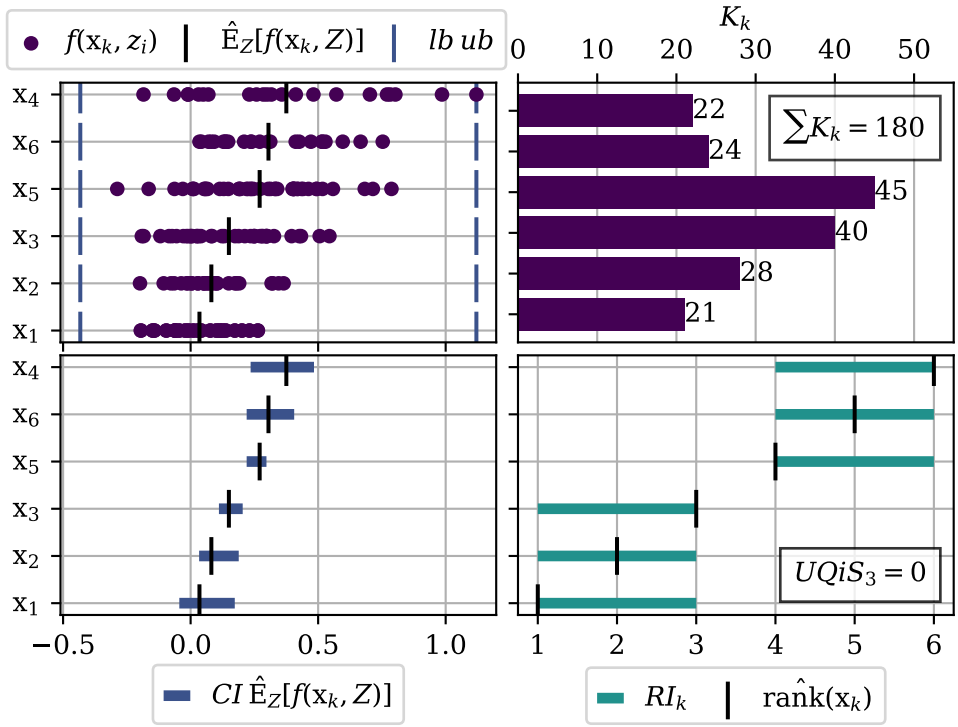

Figure 6.4: State after the final iteration of the DA by Algorithm 2 with hyperparameters of Table 6.1 for the selection of the top-3 individual from the population $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$. *Top Left*: Determined objective function values $f(\mathbf{x}_k, z_i)$, sample means and given bounds. *Top Right*: Number of allocated scenarios K_k sampled WoR from the 50 possible scenarios. *Bottom Left*: Computed CIs from the determined objective function values and sample means. *Bottom Right*: Resulting RIs, sample ranks and UQiS of the top-3 individual $UQiS_3$.

In each generation of CMA-ES, the top-μ individuals from a population are selected. Additionally, in the subsequent recombination step, the individuals are weighted based on their rank (Equation 2.5). Therefore, besides selecting the top-μ individuals, the ranking within these top-μ individuals is required. To achieve this, the DA by Algorithm 2 is conducted several times, sequentially increasing the selection of the top-μ individuals from 1 to 3. If $UQiS_1$ is zero, then $UQiS_2$ is used to allocate additional scenarios until it is zero. Once $UQiS_3$ is also zero, a total of 204 scenarios are allocated (Figure 6.5). As the top three individuals have similar objective function values, all three are evaluated on all 50 possible scenarios. The remaining individuals are assigned to the bottom three sets with fewer allocated scenarios.

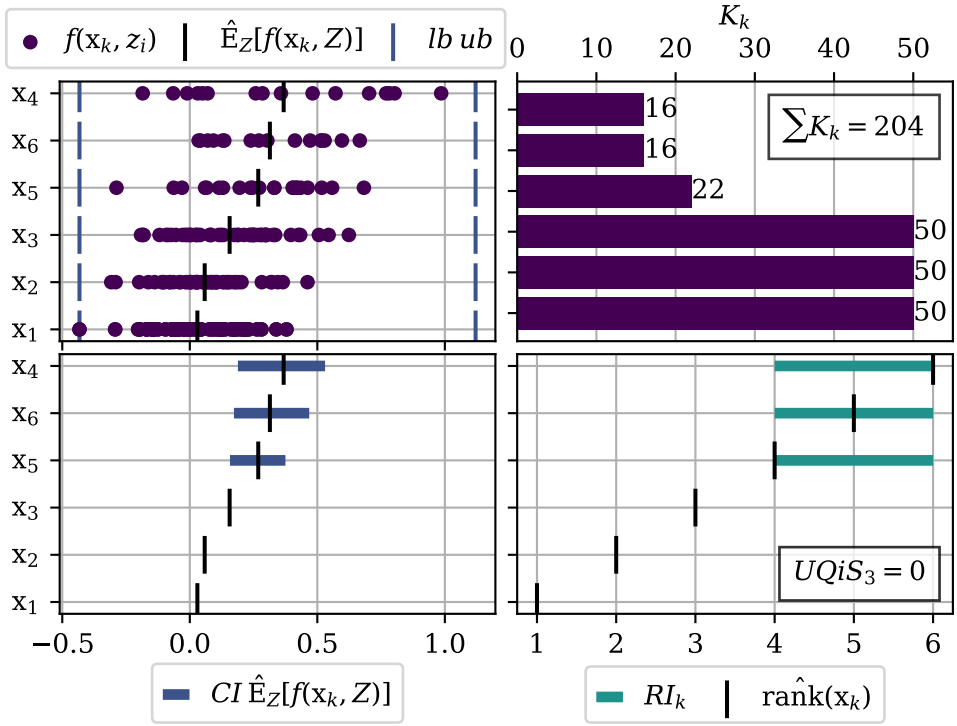

Figure 6.5: State after the final iteration of the DA by Algorithm 2 with hyperparameters of Table 6.1 for the ranking of the top-3 individual from the population $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$. *Top Left*: Determined objective function values $f(\mathbf{x}_k, z_i)$, sample means and given bounds. *Top Right*: Number of allocated scenarios K_k sampled WoR from the 50 possible scenarios. *Bottom Left*: Computed CIs from the determined objective function values and sample means. *Bottom Right*: Resulting RIs, sample ranks and UQiS of the top-3 individual $UQiS_3$.

6.2.3 CMA-ES and Bound Adaption for UQiS

The DA methodology (Algorithm 2) can be combined with CMA-ES to identify the top-μ individuals within each generation g of CMA-ES. However, since CMA-ES minimizes the objective function, the values of the objective function are likely to decrease as the generations progress. Moreover, the amount of noise may change during the optimization process. Therefore, using static lower and upper bounds for the UQiS throughout the entire optimization procedure is suboptimal.

To address this issue, a bounds adaptation mechanism is introduced. This mechanism adjusts the bounds for the UQiS dynamically after each generation based on the minimum and maximum objective function values from the last n_{gen} generations, denoted as $y_{\text{min},n_{\text{gen}}}$ and $y_{\text{max},n_{\text{gen}}}$. To account for potential future increases or decreases in the bounds, the range between the determined minimum and maximum values, scaled by a factor b_{factor}, is added to the current minimum and maximum values, respectively:

$$\begin{aligned} ub^{(g+1)} &= y_{\text{max},n_{\text{gen}}} + b_{\text{factor}} \cdot (y_{\text{max},n_{\text{gen}}} - y_{\text{min},n_{\text{gen}}}), \\ lb^{(g+1)} &= y_{\text{min},n_{\text{gen}}} + b_{\text{factor}} \cdot (y_{\text{max},n_{\text{gen}}} - y_{\text{min},n_{\text{gen}}}), \end{aligned} \tag{6.10}$$

Figure 6.6 illustrates the bound adaption with $b_{\text{factor}} = 0.25$ and $n_{\text{gen}} = 3$ during a CMA-ES run on the noisy test function with $\sigma_Z = 0.5$ and $r_{\text{noise}} = 0.9$.

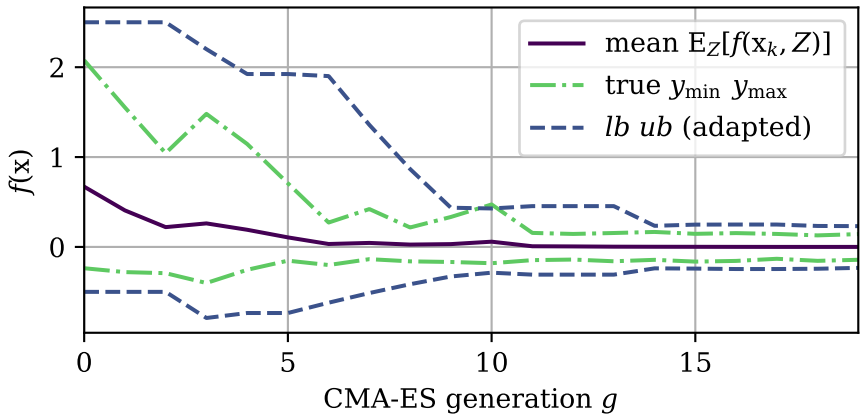

Figure 6.6: Adaption of the lower bound lb and the upper bound ub according to Equation 6.10 with $b_{\text{factor}} = 0.25$ and $n_{\text{gen}} = 3$ in each generation g of a CMA-ES run on the noisy test function (Section 6.1) with $\sigma_Z = 0.5$ and $r_{\text{noise}} = 0.9$. In addition, for the DA methodology, unknown true minimum y_{min} and maximum y_{max} objective function values are provided along with the true mean objective function value across the CMA-ES population $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$.

The significance level α is set to 0.99, and the UQiS threshold u_{thr} is set to 1. This configuration results in fewer scenarios being allocated for selecting the top-μ individuals, which resembles a more complex case for bound adaptation due to the less information available from past generations. Starting with an initial lower bound of -0.5 and an upper bound of 2.5, the first adaptation is conducted after three generations. During the 20 generations of CMA-ES, the mean objective function value and thus also the proportional noise component decreases, which results in a continuous adjustment of the bounds.

For each generation g, the bounds are considered correctly adapted if the lower bound $lb^{(g)}$ is less than the true minimum objective function value $y_{min}^{(g)}$ and the upper bound $ub^{(g)}$ is greater than the true maximum objective function value $y_{min}^{(g)}$. Over the course of 20 generations, the bounds are correct in 95% of the cases (Figure 6.6). Only in generation 10 is the upper bound slightly too low compared to the true maximum objective function value. This violates the assumptions for the UQiS, which requires that all objective function values fall within the given lower and upper bounds. Increasing the factor b_{factor} leads to looser bounds and a higher percentage of correct bounds. However, looser bounds also result in larger CIs, necessitating more function evaluations to unambiguously select the top-μ individuals. To investigate this trade-off, an experiment is conducted with varying the values of the factor b_{factor}. Table 6.2 summarizes the chosen hyperparameters, resulting in 30 configurations. For each configuration, 100 CMA-ES runs are conducted for 20 generations on the noisy test function, defined in Section 6.1.

Table 6.2: Chosen hyperparameters of the utilized noisy test function (Section 6.1) and of the DA (Algorithm 2) for the analysis of the bound adaption mechanism.

Component	Parameter Description	Parameter Values
Noisy Test Function	Dimensionality	2
	Standard deviation σ_Z	$\{0.1, 0.25, 0.5\}$
	Degree of skewness p_{skew}	0.5
	Ratio proportional noise r_{noise}	0.9
	Possible scenarios Ω_Z	$\{z_1, \ldots, z_{50}\}$
DA	Population X	$\{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$
	Significance level α	0.99
	UQiS of top-μ individuals	$\{1, 3\}$
	UQiS threshold u_{thr}	1
	Allocations per iteration n_{allocate}	6
	Initial lower bound lb	-0.5
	Initial upper bound ub	2.5
	Bound adaption factor b_{factor}	$\{0.0, 0.1, 0.25, 0.5, 1.0\}$

Figure 6.7 illustrates the relationship between the ratio of correct bounds and the ratio of evaluated scenarios, averaged across 200 CMA-ES runs, for various combinations of standard deviation σ_Z and factor b_{factor} values. As expected, a higher factor, which corresponds to looser bounds, results in a higher percentage of correct bounds. With a factor of zero, slightly more than half of the bounds are correct. Increasing the factor to 0.1 leads to approximately 90% of the bounds being correct. However, as the factor increases, the increase in accuracy decreases while the number of scenarios evaluated continues to increase. Therefore, for the subsequent experiments, the factor b_{factor} is set to 0.1 as a compromise between accuracy and computational efficiency.

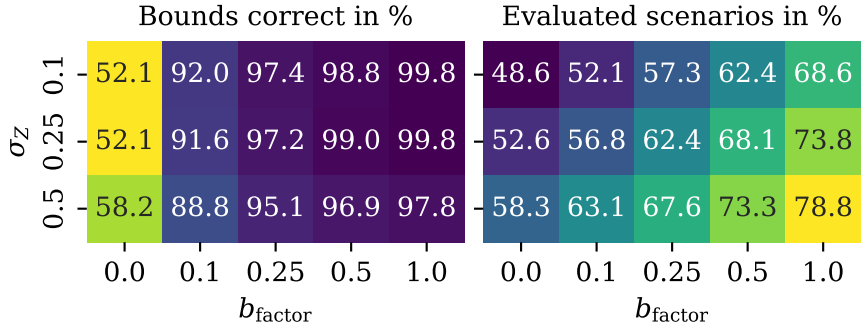

Figure 6.7: The ratio of correct bounds in percent (*left*) and the ratio of evaluated scenarios in percent (*right*) on average across 200 CMA-ES runs for different combinations of the standard deviations σ_Z and factors b_{factor} for the bound adaption according to Equation 6.10.

6.2.4 CMA-ES Convergence

The DA methodology (Algorithm 3) can be employed to determine the ranking of the top-μ individuals within a CMA-ES population (Section 6.2.2). The magnitude of uncertainty in the ranking can be primarily controlled by the user through two parameters: the significance level α and the threshold for the UQiS u_{thr}. With higher levels of uncertainty, the probability of an incorrect ranking increases.

In this section, the effect of increased uncertainty in the ranking on the convergence of CMA-ES is investigated for $\mu = 1$ and $\mu = 3$. The significance level α is varied first, followed by the threshold u_{thr}. Finally, the DA methodology is compared to SA. Table 6.3 summarizes the chosen hyperparameters. For each configuration, 100 CMA-ES runs are conducted over 50 generations on the noisy test function, defined in Section 6.1.

Table 6.3: Chosen hyperparameters of the utilized noisy test function (Section 6.1) and of the DA (Algorithm 2) for the analysis of the convergence of CMA-ES.

Component	Parameter Description	Parameter Values
Noisy Test Function	Dimensionality	2
	Standard deviation σ_Z	$\{0.1, 0.25, 0.5\}$
	Degree of skewness p_{skew}	0.5
	Ratio proportional noise r_{noise}	0.7
	Possible scenarios Ω_Z	$\{z_1, \ldots, z_{50}\}$
DA	Population X	$\{\mathbf{x}_1, \ldots, \mathbf{x}_6\}$
	Allocations per iteration n_{allocate}	6
	Initial lower and upper bound	-0.5, 2.5

Three values for the significance level $\alpha \in \{0.3, 0.6, 0.9\}$ are considered (Figure 6.8). Higher values of α correspond to increased uncertainty, which leads to larger distances to the optimal objective function value Δf^* and, therefore, slower convergence of CMA-ES. This trend is observed across all three standard deviations. The DA methodology adapts to the different noise levels for a given α by adjusting the number of evaluations. For higher values of α, fewer evaluations are required to achieve an unambiguous ranking due to the resulting tighter CIs. Determining an unambiguous ranking for $\mu = 3$ requires more evaluation than for $\mu = 1$.

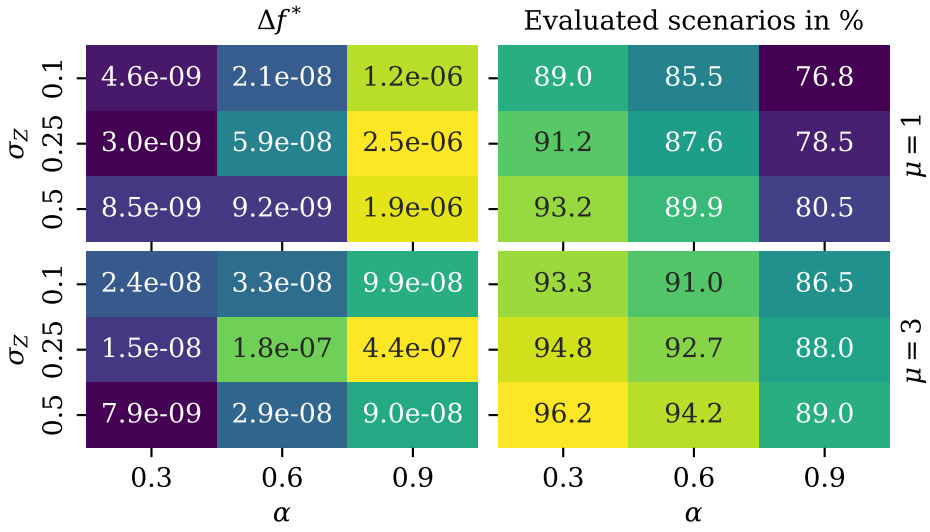

Figure 6.8: Distance to the optimal objective function value Δf^* after 50 generations of CMA-ES and the ratio of evaluated scenarios on average across 100 runs for $\mu = 1$ (first row) and $\mu = 3$ (second row). Different configurations of the significance level $\alpha \in \{0.3, 0.6, 0.9\}$ (Equation 2.26) and the standard deviation σ_Z of the noisy test function are considered.

To examine the convergence of CMA-ES across generations, the configuration with $\mu = 1$ and $\sigma_Z = 0.5$ is analyzed in detail. Figure 6.9 presents the distance to the optimal objective function value Δf^*, the ratio of evaluated scenarios and the ratio of correct ranking (RCR) of the top-μ individuals, denoted as RCR_μ, across 100 CMA-ES runs for each generation g. For $\alpha = 0.9$, the decrease in the distance to the optimal objective function value slows down after 10 generations compared to $\alpha = 0.3$ and $\alpha = 0.6$. Simultaneously, the RCR falls below 0.9. With $\alpha = 0.6$, the RCR remains above 0.9, and the convergence of CMA-ES is not noticeably impacted, even though fewer scenarios are evaluated compared to the case with $\alpha = 0.3$. CMA-ES can internally compensate for the incorrect rankings to some extent.

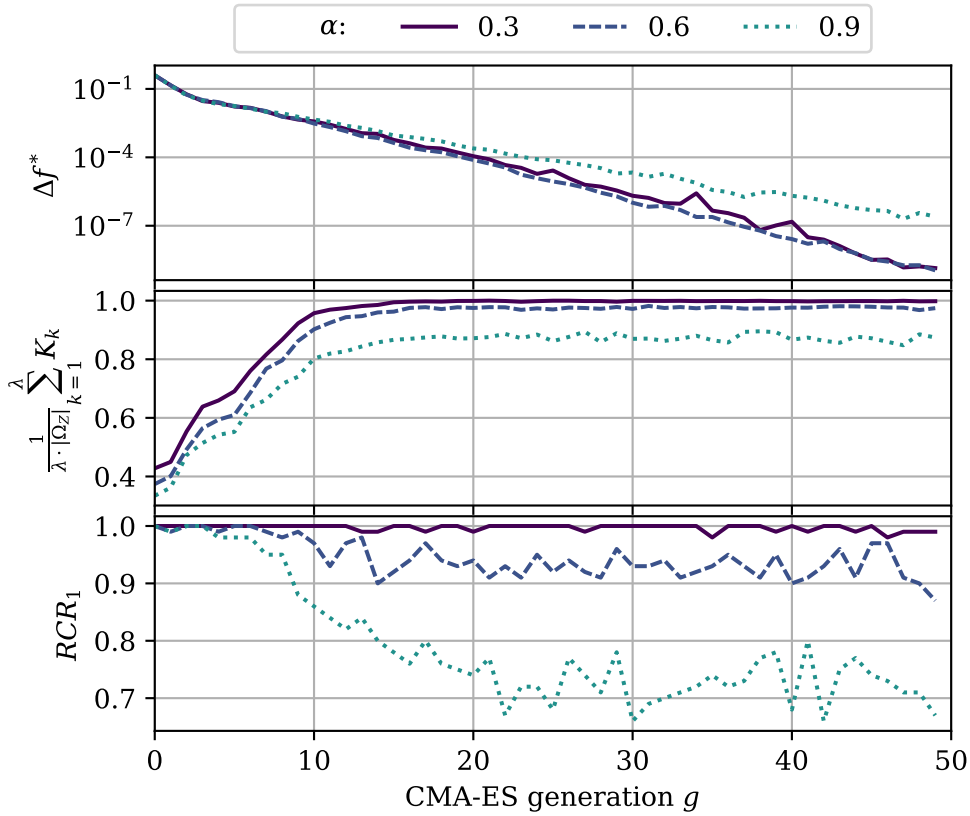

Figure 6.9: Mean distance to the optimal objective function value Δf^*, the ratio of evaluated scenarios and RCR of the top-1 individual RCR_1 for each CMA-ES generation g across 100 runs. Different values of the significance level $\alpha \in \{0.3, 0.6, 0.9\}$ (Equation 2.26) for the DA (Algorithm 2) are considered. The standard deviation of the noisy test function is set to 0.5.

However, there is no significant difference in convergence among the different considered significance levels α until the first five to ten generations. The RCR is also 1.0 for all three significance levels. Therefore, a significance level of 0.3 is too conservative in the beginning and leads to unnecessary evaluations. Conversely, a significance level of 0.9 is too high after the beginning, as not enough scenarios are allocated, and the RCR falls to 0.7.

Subsequently, the UQiS threshold u_{thr} is varied between zero and three, with the significance level α set to 0.3. The DA methodology allocates scenarios until the UQiS does not exceed the set threshold. Therefore, for thresholds greater than zero, the DA methodology allocates fewer scenarios, and the resulting ranking includes $\mu + u_{\text{thr}}$ individuals within the top-μ set. Thus, higher thresholds result in a higher uncertainty in the ranking of the top-μ individuals, leading to poorer convergence across all three standard deviations (Figure 6.10). For $\mu = 1$ and $\mu = 3$, the distance to the optimal objective function value after 50 generations of CMA-ES is similar for the different considered thresholds. However, for $\mu = 3$, more scenarios are required.

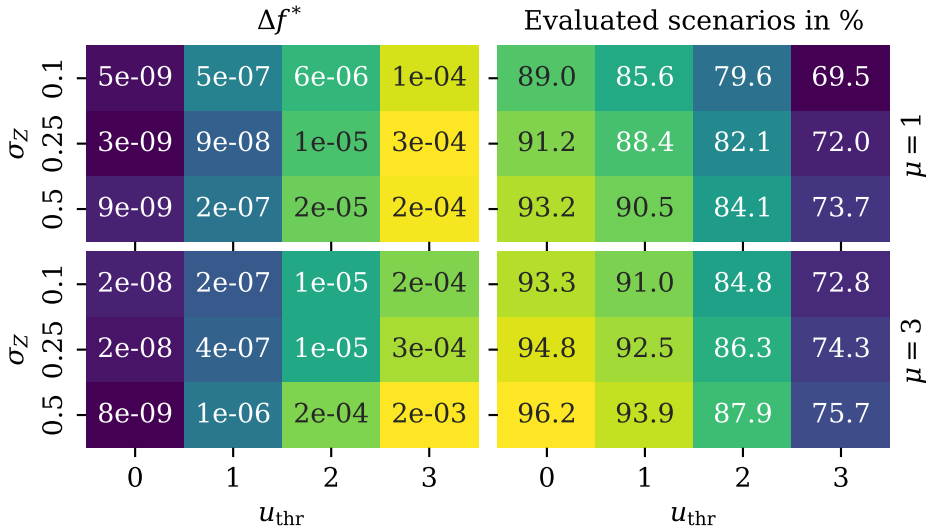

Figure 6.10: Distance to the optimal objective function value Δf^* after 50 generations of CMA-ES and the ratio of evaluated scenarios on average across 100 runs for $\mu = 1$ (first row) and $\mu = 3$ (second row). Different configurations of the UQiS threshold $u_{\text{thr}} \in \{0, 1, 2, 3\}$ and the standard deviation $\sigma_Z \in \{0.1, 0.25, 0.5\}$ of the noisy test function are considered.

For the configuration with $\mu = 1$ and $\sigma_Z = 0.5$, Figure 6.11 presents the distance to the optimal objective function value Δf^*, the ratio of evaluated scenarios and the ratio of correct ranking (RCR) of the top-μ individuals across 50 generations of CMA-ES for the different considered thresholds $u_{\text{thr}} \in \{0, 1, 2, 3\}$. With higher thresholds, the convergence slows down with further generations. After a continuous increase in the number of allocated scenarios in the first ten generations, the allocation stagnates. With a threshold of one, almost all possible scenarios are allocated. However, the RCR is only around 0.8 due to the two possible individuals within the top-1 individual according to the UQiS. Thus, the threshold needs to be zero to receive a correct ranking according to the UQiS.

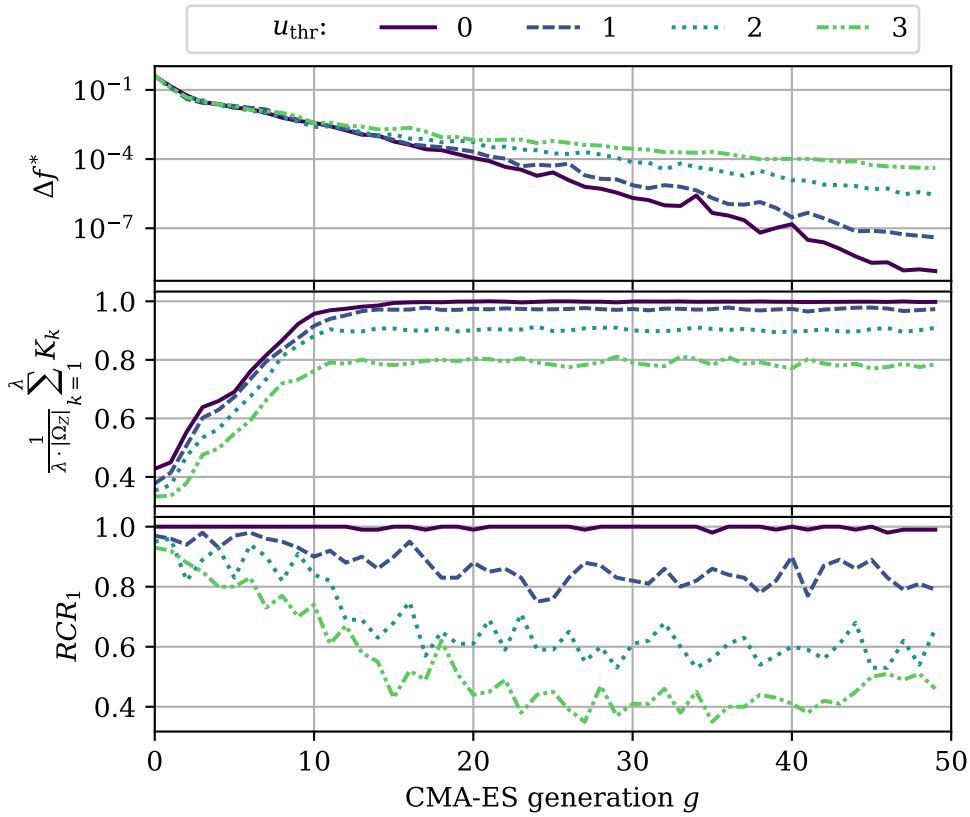

Figure 6.11: Mean distance to the optimal objective function value Δf^*, the ratio of evaluated scenarios and RCR of the top-1 individual RCR_1 for each CMA-ES generation g across 100 runs. Different values of the UQiS threshold $u_{\text{thr}} \in \{0, 1, 2, 3\}$ for the DA (Algorithm 2) are considered. The standard deviation of the noisy test function is set to 0.5.

However, to reduce the number of evaluations, the threshold can be set greater than zero in the beginning. This will not significantly impact the RCR and convergence. The reason for this is that in the beginning, the objective function values of the individuals overlap less, resulting in fewer rank changes as more scenarios are allocated. Therefore, the initial ranking is often already correct. Furthermore, selecting the second-best individual can also lead to significant improvement, and the CMA-ES population evolves even with an incorrect ranking provided.

Finally, the DA is compared to SA with different numbers of allocated scenarios $K_k \in \{45, 40, 30, 20, 5\}$ per individual considered (Figure 6.12).

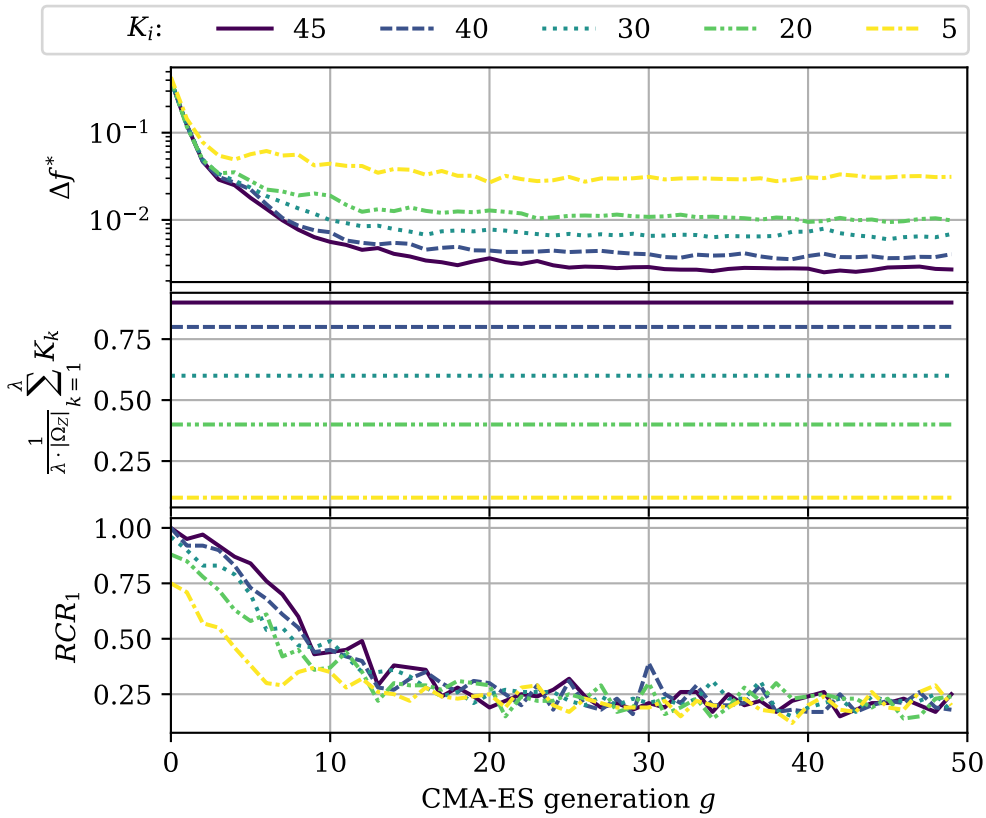

Figure 6.12: Mean distance to the optimal objective function value Δf^*, ratio of evaluated scenarios and RCR of the top-1 individual RCR_1 for each CMA-ES generation g across 100 runs. Different numbers of allocated scenarios $K_k \in \{45, 40, 30, 20, 5\}$ per individual for the SA are considered. The standard deviation of the noisy test function is set to 0.5.

Due to the SA, the number of allocated scenarios is constant across all generations. Thus, no adaptation to the current uncertainty in the ranking is conducted. Concerning convergence of CMA-ES, in the beginning, only five allocated scenarios to each individual are enough for progress, even with an RCR of below 0.75. However, the RCR of the other SA with more scenarios also falls below 0.5 within the first ten generations. Here, two clear drawbacks of the SA are present. To reach efficiently a constant high RCR, in the beginning, fewer evaluations are needed, and with further progress, more and more evaluations are needed for a correct ranking. Thus, a DA is required. A further drawback of SA is the allocation of the same number of scenarios to each individual, which wastes evaluations on the last ranked individuals with no information gain for ranking the top-1 individual.

6.2.5 Conclusion

The DA methodology (Algorithm 2) introduced in this work, which is based on uncertainty quantification, effectively addresses the drawbacks associated with SA. By employing a policy (Algorithm 3), scenarios are dynamically allocated to individuals within a population, thereby reducing the uncertainty in the ranking of these individuals. This methodology enables the ranking or selection of the top-μ individuals with a high degree of confidence. The DA methodology distributes function evaluations more effectively to the individuals than SA. However, the uncertainty quantification method requires the user to provide upper and lower bounds for the objective function values, which are a priori unknown and must be estimated.

The combination of the DA methodology and CMA-ES is not without its imperfections. Initially, the DA tends to allocate too many scenarios to achieve a correct ranking, which is unnecessary for the early stages of CMA-ES. The algorithm is capable of evolving effectively even with a ranking that is not perfectly accurate. Therefore, allowing for higher uncertainty, for example, by setting a higher threshold in the beginning, can reduce the number of allocated scenarios. Furthermore, no information about the uncertainty from past generations is used, the uncertainty quantification starts anew for each population. Only the bounds are adjusted based on information from previous generations. Using more information from past generations could be a valuable direction for future research.

6.3 Application to Real-World Problems

The optimization of parameters in vehicle dynamics control systems, such as ABS and ARP, is a challenging task due to the inherent noise. The objective function value for a given parameter configuration is determined by the sample mean across several repeated maneuvers, with each maneuver considered a distinct scenario.

In this section, the uncertainty quantification and DA methodology is applied to the two-dimensional real-world problem y_1, which involves a partially loaded vehicle with high-performance tires (Section 3.4). For each of the 10 101 possible parameter configurations, 30 braking maneuvers are simulated, and the resulting braking distances are obtained.

The objective function for the real-world problem y_1 returns the distance of the braking distance in meters to the global optimal average braking distance across all 30 considered scenarios. Figure 6.13 illustrates these distances, which are averaged over various numbers of randomly selected scenarios. The first input parameter x_1 is varied, while the second input parameter x_2 is held at its optimal value.

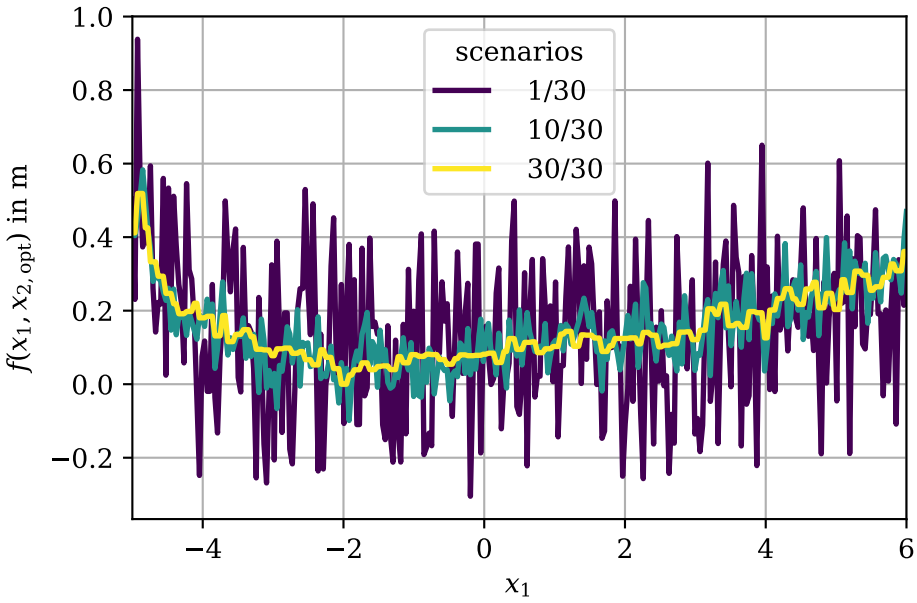

Figure 6.13: Distance of the braking distance in meters to the global optimal average braking distance across different numbers of randomly selected scenarios for the two-dimensional real-world problem y_1 (Table 3.1). The first input parameter x_1 is varied, while the second input parameter x_2 is held at its optimal value.

The dataset for the two-dimensional real-world problem y_1 consists of 303 030 objective function values, which maps the objective function landscape completely. Evaluating the objective function by using the dataset is inexpensive and eliminates the need for further costly simulations. This allows to analyze the performance and convergence of CMA-ES in combination with the DA methodology on the real-world problem, similar to Section 6.2.4. For the analysis, CMA-ES is configured with a population size of six individuals. The DA methodology allocates scenarios during each generation of CMA-ES until the UQiS of the top-1 individual reaches zero. For each significance level $\alpha \in \{0.3, 0.6, 0.9\}$, 100 CMA-ES runs of 50 generations each are performed. The initial lower and upper bounds of the objective function values are set between -0.3 and 1.2 m. Figure 6.14 presents the results.

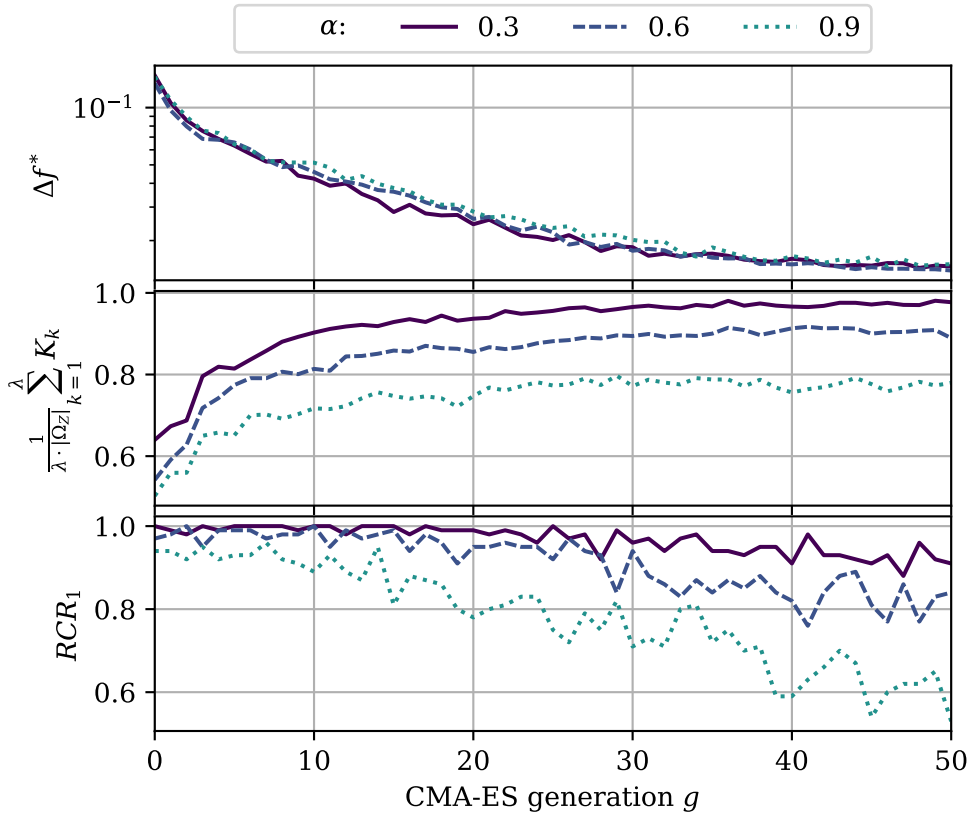

Figure 6.14: Mean distance to the optimal objective function value Δf^*, the ratio of evaluated scenarios and RCR of the top-1 individual RCR_1 for each CMA-ES generation g across 100 runs on the two-dimensional real-world problem y_1 (Table 3.1). Different values of the significance level $\alpha \in \{0.3, 0.6, 0.9\}$ (Equation 2.26) for the DA (Algorithm 2) are considered.

Across the 50 generations of CMA-ES, the distance to the optimal objective function value decreases consistently for all three significance levels $\alpha \in \{0.3, 0.6, 0.9\}$ considered, with no discernible differences in convergence. However, the ratio of evaluated scenarios is lower for higher significance levels across all 50 generations. Thus, on the real-world problem, a higher significance level for the uncertainty quantification is more efficient when applied in combination with CMA-ES. The lower RCR results from higher significance levels and decreases further across generations. This does not negatively impact the convergence of CMA-ES and is similar to the results on the noisy test function in Section 6.2.4. The DA methodology demonstrates that evaluations can be significantly reduced compared to evaluating all individuals across all scenarios. Specifically, with significance levels of 0.3 and 0.9, reductions of 8% and 26%, respectively, are achieved.

Regardless of the selected significance level, not every CMA-ES run reaches the optimal value of the objective function. On average, the deviation from the optimal value is approximately 2 cm. The reason for this is the presence of noise, even if all possible scenarios are evaluated. This noise contributes to a multimodal objective function landscape and can cause CMA-ES to converge to a local optimum instead of the global optimum. Two potential strategies can be employed to address this challenge: increasing the population size or the number of possible scenarios. First, a larger population size enhances the exploratory capabilities of the optimization algorithm. Thus, the optimization algorithm is more likely to escape a local optima. Second, a larger number of scenarios reduces the impact of noise and smoothes the objective function landscape. This simplifies the identification of the true global optimum for the optimization algorithm.

The RCR is with a significance level of 0.3 above 0.9 across all 50 generations. Thus, a significance level of 0.3 is effective for selecting the best parameter configuration with high probability, regardless of the generation. In the last generations of CMA-ES, almost all scenarios must be allocated to achieve correct ranking. Thus, even low levels of noise can complicate the differentiation between individuals. This could serve as a termination criterion or a sign to increase the number of scenarios.

Combining the DA methodology with CMA-ES can reduce the required evaluations by up to 26% in a real-world setting. However, the precision of uncertainty quantification for CMA-ES may be excessive, as the algorithm can progress even with some incorrect rankings, suggesting potential for further efficiency gains. Nonetheless, the DA methodology is adept at ranking individuals correctly and efficiently within a population.

The DA methodology and associated uncertainty quantification are not restricted to simulated objective function values but can also be applied to real-world vehicle measurements on a test track. Uncertainty quantification can provide information to reduce the number of maneuvers required. Furthermore, this information can guide engineers when choosing between parameter configurations of comparable quality.

Chapter 7

Conclusion and Outlook

This thesis focuses on designing and tuning optimization algorithms that efficiently navigate through the complicated problem landscapes typical of real-world challenges in the design of vehicle dynamics control system parameters. Methods are presented for analyzing and improving the performance and applicability of optimization algorithms in solving computationally expensive mixed-integer black-box problems. Seven research questions posed in Section 1.2 are addressed in this thesis. The answers to these questions are summarized below.

In Chapter 3, the first research question is addressed by objectifying the desired behavior of the two control systems ABS and ARP. For example, the performance of the ABS parameters can be assessed through the braking distance, which enables the translation of the complex engineering problem into a quantifiable mathematical objective function. Defining the objective function of an engineering problem is a basic requirement for the application of optimization algorithms. Moreover, this objective function serves as the basis for subsequent algorithmic developments. To circumvent the computational costs of vehicle dynamics simulations, a comprehensive dataset is created using the workflow described in Section 3.3. Once created, this dataset serves as a complete replacement for the simulation with very low evaluation costs. Multiple optimization runs to obtain robust statistical results when benchmarking optimization algorithms on the real-world problem can be conducted time-efficiently.

Chapter 4 stands as a cornerstone of this thesis, addressing the second research question of how to tune optimization algorithm parameters for specific computationally expensive real-world optimization problems. The research explored the creation and use of computationally inexpensive surrogate problems to tune the parameters of an

optimization algorithm. The similarity of optimization problems is quantified based on ELA features. This allows for precise calibration of algorithm parameters in a more computationally tractable but similar environment. The validation of the presented method through rigorous experimentation shows that the performance of CMA-ES can be significantly improved by using surrogate problems with similar properties to the original optimization landscape.

Furthermore, Chapter 4 contributes to answering the third research question, whether a universal parameter configuration can suffice for a class of related real-world problems or whether each instance requires a tailored approach. The research has shown that while a singular, well-tuned parameter configuration can indeed yield superior performance across different instances within a problem class, such as the design of vehicle dynamics control system parameters, there is also merit to the argument that individual problems may benefit from tailored parameter configurations. The meta-optimization framework developed allows for both strategies: tuning for general applicability or problem-specific optimization. The answer to the fourth research question is to define the identification of the optimal parameters of an optimization algorithm for solving a specific optimization problem as a meta-optimization task. The results of a comprehensive experiment show that compared to established algorithms such as SMAC and TPE, the use of CMA-ESwM as a meta-optimization algorithm is a cost-effective strategy for tuning the parameters of CMA-ES.

The parameters of the considered real-world problems are discrete, which raises the fifth research question addressed in Chapter 5: How does this discretization affect the optimization performance of the used optimization algorithm CMA-ES? The chapter presents a comparison and in-depth analysis of different CMA-ES variants and an extension of CMA-ES specifically designed to deal with discrete parameters. A spectrum of discretization levels and problem dimensionality is considered. The extension of CMA-ES, combined with strategic algorithmic adaptations, has demonstrated remarkable robustness in effectively handling the challenges introduced by discretization.

The sixth research question is: How can uncertainty in the objective function be quantified? Therefore, methods for the uncertainty quantification are presented in Section 2.7. Moreover, these methods can be combined with a DA policy described in Chapter 6, enabling the allocation of scenarios to individuals within a population dynamically. The presented methodology ensures efficient ranking with a high degree of confidence, which in turn reduces the number of evaluations required in uncertain landscapes.

The final research question asks about the advantage of combining this policy with CMA-ES. On the real-world problem, the computational resources saved throughout the optimization process are up to 26%. However, the proposed methodology is quite conservative. Thus, there is further potential for improvements by allocating fewer evaluations.

In a nutshell, by using the developed meta-optimization framework and specific extensions for CMA-ES, this thesis provides tailored CMA-ES configurations to solve mixed-integer black-box optimization problems with inherent uncertainty, such as the design of VDCS parameters. Furthermore, the developed meta-optimization framework is also applicable to other optimization algorithms and optimization problems beyond the scope of VDCSs.

From this thesis, several promising avenues for future research emerge. One potential direction is to extend the tuning methodology presented in Chapter 4 to other real-world problem classes. Moreover, this allows for the analysis of the differences and similarities of these problem classes, such as vehicle dynamics, vehicle crash, or even non-vehicle domains, such as economics or finance. Furthermore, a database that stores and uses information from previous tuning and optimization runs can provide a path to enhance the efficiency of the developed method. Finally, an open research question is the extension from single-objective to multi-objective optimization, especially in the context of algorithm tuning.

As highlighted in Chapter 5, future research may address strategies for managing high-dimensional settings with numerous discrete levels. Improving the efficiency of uncertainty quantification in optimization algorithms, such as CMA-ES, as discussed in Chapter 6, also provides fertile ground for research.

The methods and findings presented in this thesis narrow the gap between the application of sophisticated optimization algorithms and computationally expensive black-box problems. These methods provide a basis for future developments and improvements in vehicle design and control as the automotive industry and scientific research continue to evolve.

Bibliography

[1] Ali Ahrari, Saber Elsayed, Ruhul Sarker, Daryl Essam, and Carlos A. Coello Coello. Revisiting Implicit and Explicit Averaging for Noisy Optimization. *IEEE Transactions on Evolutionary Computation*, 27(5):1250–1259, 2023.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-Generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery.

[3] Satyajith Amaran, Nikolaos V. Sahinidis, Bikram Sharda, and Scott J. Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380, 2016.

[4] Martin Andersson, Sunith Bandaru, Amos H.C. Ng, and Anna Syberfeldt. Parameter Tuned CMA-ES on the CEC'15 Expensive Problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 1950–1957, 2015.

[5] Jarosław Arabas, Adam Szczepankiewicz, and Tomasz Wroniak. Experimental Comparison of Methods to Handle Boundary Constraints in Differential Evolution. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, pages 411–420, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[6] Dirk V. Arnold and Hans-Georg Beyer. Local Performance of the $(\mu/\mu_I, \lambda)$-ES in a Noisy Environment. In Worthy N. Martin and William M. Spears, editors, *Foundations of Genetic Algorithms 6*, pages 127–141. Morgan Kaufmann, San Francisco, 2001.

[7] Dirk V. Arnold and Hans-Georg Beyer. A general noise model and its effects on evolution strategy performance. *IEEE Transactions on Evolutionary Computation*, 10(4):380–391, 2006.

[8] Charles Audet and Warren Hare. *Derivative-Free and Blackbox Optimization*. Springer, Cham, 2017.

[9] Anne Auger and Nikolaus Hansen. A Restart CMA Evolution Strategy With Increasing Population Size. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776, 2005.

[10] Anne Auger and Nikolaus Hansen. Performance Evaluation of an Advanced Local Search Evolutionary Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1777–1784, 2005.

[11] Thomas Bäck. Parallel Optimization of Evolutionary Algorithms. In Gerhard Goos, Juris Hartmanis, Jan Leeuwen, Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature — PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 418–427. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[12] Thomas Bäck, Christophe Foussette, and Peter Krause. *Contemporary Evolution Strategies*. Natural Computing Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 1st ed. edition, 2013.

[13] Thomas Bäck, Anna V. Kononova, Bas van Stein, Hao Wang, Kirill A. Antonov, Roman T. Kalkreuth, Jacob de Nobel, Diederick Vermetten, Roy de Winter, and Furong Ye. Evolutionary Algorithms for Parameter Optimization—Thirty Years Later. *Evolutionary Computation*, 31(2):81–122, 2023.

[14] Thomas Bartz-Beielstein, Carola Doerr, Daan van den Berg, Jakob Bossek, Sowmya Chandrasekaran, Tome Eftimov, Andreas Fischbach, Pascal Kerschke, William La Cava, Manuel Lopez-Ibanez, Katherine M. Malan, Jason H. Moore, Boris Naujoks, Patryk Orzechowski, Vanessa Volz, Markus Wagner, and Thomas Weise. Benchmarking in Optimization: Best Practice and Open Issues. Technical report, 2020.

[15] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer. Per Instance Algorithm Configuration of CMA-ES with Limited Budget. In Peter

A. N. Bosman, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM Digital Library, pages 681–688, New York, NY, USA, 2017. Association for Computing Machinery.

[16] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc, 2011.

[17] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution Strategies – A Comprehensive Introduction. *Natural Computing*, 1(1):3–52, 2002.

[18] Rafał Biedrzycki. Handling bound constraints in CMA-ES: An experimental study. *Swarm and Evolutionary Computation*, 52:100627, 2020.

[19] Patrick L. Boyd. NHTSA's NCAP Rollover Resistance Rating System. Technical report, National Highway Traffic Safety Administration, United States, 2004.

[20] Jürgen Branke, Stephen E. Chick, and Christian Schmidt. Selecting a Selection Procedure. *Management Science*, 53(12):1916–1932, 2007.

[21] Dimo Brockhoff, Anne Auger, Nikolaus Hansen, Dirk V. Arnold, and Tim Hohm. Mirrored Sampling and Sequential Selection for Evolution Strategies. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, Lecture Notes in Computer Science, pages 11–21. Springer, Berlin, 2010.

[22] Chun-Hung Chen, S. David Wu, and Liyi Dai. Ordinal comparison of heuristic algorithms using stochastic optimization. *IEEE Transactions on Robotics and Automation*, 15(1):44–56, 1999.

[23] E. Jack Chen. Using ordinal optimization approach to improve efficiency of selection procedures. *Discrete Event Dynamic Systems*, 14(2):153–170, 2004.

[24] Jiecao Chen, Xi Chen, Qin Zhang, and Yuan Zhou. Adaptive Multiple-Arm Identification. Technical report, 2017.

[25] Jennifer N. Dang. Preliminary results analyzing the effectiveness of electronic stability control (ESC) systems. Technical report, US Department of Transportation, National Highway Traffic Safety Administration, 2004.

[26] Jacob de Nobel, Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Bäck. Tuning as a Means of Assessing the Benefits of New Ideas in Interplay with Existing Algorithmic Modules. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, pages 1375–1384, New York, NY, USA, 2021. Association for Computing Machinery.

[27] Isabelle D. Dourson. *KI-basierte virtuelle Vorapplikation in der Fahrdynamikauslegung*. Master Thesis, RWTH Aachen University, Aachen, 2022.

[28] David Eckman. *Reconsidering Ranking-and-Selection Guarantees*. Dissertation, Cornell University Library, 2019.

[29] David J. Eckman and Shane G. Henderson. Posterior-Based Stopping Rules for Bayesian Ranking-and-Selection Procedures. *INFORMS Journal on Computing*, 34(3):1711–1728, 2022.

[30] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[31] A. E. Eiben and James E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, Berlin and Heidelberg, 2 edition, 2015.

[32] Sebastian Ellmaier. *Uncertainty Management for Reduction of Simulation Costs in Virtual Application of Vehicle Dynamics Control*. Master Thesis, Technical University of Munich, Munich, 2023.

[33] Alena Erke. Effects of electronic stability control (ESC) on accidents: a review of empirical evidence. *Accident Analysis & Prevention*, 40(1):167–173, 2008.

[34] Leonard Evans. Antilock Brake Systems and Risk of Different Types of Crashes in Traffic. *Journal of Crash Prevention and Injury Control*, 1(1):5–23, 1999.

[35] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and Robust Automated Machine Learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc, 2015.

[36] J. Michael Fitzpatrick and John J. Grefenstette. Genetic Algorithms in Noisy Environments. *Machine Learning*, 3(2):101–120, 1988.

[37] Christodoulos A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford University Press, 1995.

[38] Michael C. Fu. Optimization via simulation: A review. *Annals of Operations Research*, 53(1):199–247, 1994.

[39] Michael Gallagher. Comparing Proportional Representation Electoral Systems: Quotas, Thresholds, Paradoxes and Majorities. *British Journal of Political Science*, 22(4):469–496, 1992.

[40] Jean Dickinson Gibbons, Ingram Olkin, and Milton Sobel. *Selecting and ordering populations: A new statistical methodology*, volume 26 of *Classics in applied mathematics*. SIAM, Philadelphia, Pa., 1999.

[41] J. P. Gram. Ueber die Entwickelung reeller Funktionen in Reihen mittelst der Methode der kleinsten Quadrate. *Journal für die reine und angewandte Mathematik*, 1883(94):41–73, 1883.

[42] John Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128, 1986.

[43] Matthew Groves and Juergen Branke. Sequential sampling for noisy optimisation with CMA-ES. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 1023–1030, New York, NY, USA, 2018. Association for Computing Machinery.

[44] Dae Woong Ham, Michael Lindon, Martin Tingley, and Iavor Bojinov. Design-Based Confidence Sequences: A General Approach to Risk Mitigation in Online Experimentation. *SSRN Electronic Journal*, 2023.

[45] Ryoki Hamano, Shota Saito, Masahiro Nomura, and Shinichi Shirakawa. Benchmarking CMA-ES with Margin on the Bbob-Mixint Testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, pages 1708–1716, New York, NY, USA, 2022. Association for Computing Machinery.

[46] Ryoki Hamano, Shota Saito, Masahiro Nomura, and Shinichi Shirakawa. CMA-ES with Margin: Lower-Bounding Marginal Probability for Mixed-Integer Black-Box Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, pages 639–647, New York, NY, USA, 2022. Association for Computing Machinery.

[47] Nikolaus Hansen. The CMA Evolution Strategy: A Comparing Review. In Jose A. Lozano, editor, *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer, Berlin and Heidelberg and New York, 2006.

[48] Nikolaus Hansen. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, ACM Conferences, pages 2389–2396, New York, NY, USA, 2009. Association for Computing Machinery.

[49] Nikolaus Hansen. A CMA-ES for Mixed-Integer Nonlinear Optimization: Research Report. Technical Report RR-7751, INRIA, 2011.

[50] Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial. Technical report, 2016.

[51] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github, 2019.

[52] Nikolaus Hansen and Anne Auger. Evolution Strategies and CMA-ES (Covariance Matrix Adaptation). In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO Comp '14, pages 513–534, New York, NY, USA, 2014. Association for Computing Machinery.

[53] Nikolaus Hansen, Anne Auger, Dimo Brockhoff, and Tea Tušar. Anytime Performance Assessment in Blackbox Optimization Benchmarking. *IEEE Transactions on Evolutionary Computation*, 26(6):1293–1305, 2022.

[54] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions: Research Report. Technical Report RR-6829, INRIA, 2009.

[55] Nikolaus Hansen, André S.P. Niederberger, Lino Guzzella, and Petros Koumoutsakos. A Method for Handling Uncertainty in Evolutionary Optimization With an Application to Feedback Control of Combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197, 2009.

[56] Nikolaus Hansen and Andreas Ostermeier. Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In

Proceedings of the IEEE International Conference on Evolutionary Computation, pages 312–317, 1996.

[57] Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[58] Y-C Ho, C. G. Cassandras, C-H Chen, and L. Dai. Ordinal optimisation and simulation. *Journal of the Operational Research Society*, 51(4):490–500, 2000.

[59] L. Jeff Hong, Weiwei Fan, and Jun Luo. Review on ranking and selection: A new perspective. *Frontiers of Engineering Management*, 8(3):321–343, 2021.

[60] Steven R. Howard and Aaditya Ramdas. Sequential estimation of quantiles with applications to A/B testing and best-arm identification. *Bernoulli*, 28(3), 2022.

[61] Susan R. Hunter and Barry L. Nelson. Parallel Ranking and Selection. In Andreas Tolk, John Fowler, Guodong Shao, and Enver Yücesan, editors, *Advances in Modeling and Simulation: Seminal Research from 50 Years of Winter Simulation Conferences*, pages 249–275. Springer International Publishing, Cham, 2017.

[62] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[63] International Organization for Standardization. ISO 21994:2007 - Passenger cars - Stopping distance at straight-line braking with ABS - Open-loop test method, 2007.

[64] International Organization for Standardization. ISO 19365:2016 - Passenger cars - Validation of vehicle dynamic simulation - Sine with dwell stability control testing, 2016.

[65] Grahame A. Jastrebski and Dirk V. Arnold. Improving Evolution Strategies through Active Covariance Matrix Adaptation. In *IEEE International Conference on Evolutionary Computation*, pages 2814–2821, 2006.

[66] Alexandre D. Jesus, Arnaud Liefooghe, Bilel Derbel, and Luis Paquete. Algorithm Selection of Anytime Algorithms. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, pages 850–858, New York, NY, USA, 2020. Association for Computing Machinery.

[67] Haotian Jiang, Jian Li, and Mingda Qiao. Practical Algorithms for Best-K Identification in Multi-Armed Bandits. page arXiv:1705.06894, 2017.

[68] Yaochu Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[69] Ian T. Jolliffe. *Principal Component Analysis*. Springer eBook Collection Mathematics and Statistics. Springer, New York, NY, 1986.

[70] Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. Detecting Funnel Structures by Means of Exploratory Landscape Analysis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ACM Digital Library, pages 265–272, New York, NY, 2015. Association for Computing Machinery.

[71] Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models. In Frank Neumann, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM Digital Library, pages 229–236, New York, NY, USA, 2016. Association for Computing Machinery.

[72] Pascal Kerschke and Heike Trautmann. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation*, 27(1):99–127, 2019.

[73] Pascal Kerschke and Heike Trautmann. Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package Flacco. In Nadja Bauer, Katja Ickstadt, Karsten Lübke, Gero Szepannek, Heike Trautmann, and Maurizio Vichi, editors, *Applications in Statistical Computing*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 93–123. Springer, 2019.

[74] Martin Klein, Tommy Wright, and Jerzy Wieczorek. A Joint Confidence Region for an Overall Ranking of Populations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 69(3):589–606, 2020.

[75] Heinz-Jürgen Koch-Dücker and Ulrich Papert. Antilock braking system (ABS). In Konrad Reif, editor, *Brakes, Brake Control and Driver Assistance Systems*, Bosch professional automotive information, pages 74–93. Springer Vieweg, Wiesbaden, 2014.

[76] Mykel J. Kochenderfer and Tim Allan Wheeler. *Algorithms for Optimization*. The MIT Press, Cambridge and London, 2019.

[77] Anna V. Kononova, David W. Corne, Philippe de Wilde, Vsevolod Shneer, and Fabio Caraffini. Structural bias in population-based algorithms. *Information Sciences*, 298:468–490, 2015.

[78] Friedrich Kost, Jürgen Schuh, Heinz-Jürgen Koch-Dücker, Frank Niewels, Thomas Ehret, Jochen Wagner, Ulrich Papert, Peter Eberspächer, and Frank Heinen. Electronic Stability Program (ESP). In Konrad Reif, editor, *Automotive Mechatronics: Automotive Networking, Driving Stability Systems, Electronics*, pages 378–393. Springer Fachmedien Wiesbaden, Wiesbaden, 2015.

[79] Johannes Willem Kruisselbrink. *Evolution Strategies for Robust Optimization*. Doctoral Thesis, Leiden University, Leiden, 2012.

[80] Arun Kumar Kuchibhotla and Qinqing Zheng. Near-Optimal Confidence Sequences for Bounded Random Variables. Technical report, 2020.

[81] Anders Lie, Claes Tingvall, Maria Krafft, and Anders Kullgren. The effectiveness of electronic stability control (ESC) in reducing real life crashes and injuries. *Traffic injury prevention*, 7(1):38–43, 2006.

[82] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

[83] Fu Xing Long, Bas van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. Learning the Characteristics of Engineering Optimization Problems with Applications in Automotive Crash. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, New York, NY, USA, 2022. Association for Computing Machinery.

[84] Fu Xing Long, Diederick Vermetten, Bas van Stein, and Anna V. Kononova. BBOB Instance Analysis: Landscape Properties and Algorithm Performance

Across Problem Instances. In João Correia, Stephen Smith, and Raneem Qaddoura, editors, *Applications of Evolutionary Computation*, Lecture Notes in Computer Science, pages 380–395, Cham, 2023. Springer Nature Switzerland and Imprint Springer.

[85] Monte Lunacek and Darrell Whitley. The Dispersion Metric and the CMA Evolution Strategy. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, page 477. Association for Computing Machinery, 2006.

[86] Katherine M. Malan. A Survey of Advances in Landscape Analysis for Optimisation. *Algorithms*, 14(2):40, 2021.

[87] Katherine M. Malan and Andries P. Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013.

[88] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory Landscape Analysis. In Pier Luca Lanzi, editor, *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ACM Conferences, pages 829–836, New York, NY, USA, 2011. ACM.

[89] Olaf Mersmann, Mike Preuss, and Heike Trautmann. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, Lecture Notes in Computer Science, pages 73–82. Springer, Berlin, 2010.

[90] Manfred Mitschke and Henning Wallentowitz. *Dynamik der Kraftfahrzeuge*. VDI-Buch. Springer Vieweg, Wiesbaden, 5., überarb. und erg. aufl. edition, 2014.

[91] Mario A. Muñoz, Michael Kirley, and Saman K. Halgamuge. Exploratory Landscape Analysis of Continuous Space Optimization Problems Using Information Content. *IEEE Transactions on Evolutionary Computation*, 19(1):74–87, 2015.

[92] Mario A. Muñoz, Yuan Sun, Michael Kirley, and Saman K. Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245, 2015.

[93] Barry L. Nelson. Foundations of Ranking & Selection for Simulation Optimization. In Zdravko Botev, Alexander Keller, Christiane Lemieux, and Bruno Tuffin, editors, *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*, pages 353–379. Springer International Publishing, Cham, 2022.

[94] Barry L. Nelson and Linda Pei. Simulation Optimization and Sensitivity. In *Foundations and Methods of Stochastic Simulation: A First Course*, pages 231–285. Springer International Publishing, Cham, 2021.

[95] Francesco Orabona and Kwang-Sung Jun. Tight Concentrations and Confidence Sequences From the Regret of Universal Portfolio. *IEEE Transactions on Information Theory*, 70(1):436–455, 2024.

[96] Art B. Owen. Scrambling Sobol' and Niederreiter–Xing Points. *Journal of Complexity*, 14(4):466–489, 1998.

[97] Hans B. Pacejka and Egbert Bakker. The Magic Formula Tyre Model. *Vehicle System Dynamics*, 21(sup001):1–18, 1992.

[98] S. Gopal Krishna Patro and Kishore Kumar Sahu. Normalization: A Preprocessing Stage. Technical report, 2015.

[99] Yijie Peng, Chun-Hung Chen, Edwin K. P. Chong, and Michael C. Fu. A Review of Static and Dynamic Optimization for Ranking and Selection. In *2018 Winter Simulation Conference (WSC)*, pages 1909–1920, 2018.

[100] Alejandro Piad-Morffis, Suilan Estévez-Velarde, Antonio Bolufé-Röhler, James Montgomery, and Stephen Chen. Evolution Strategies with Thresheld Convergence. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2097–2104, 2015.

[101] Erik Pitzer and Michael Affenzeller. A Comprehensive Survey on Fitness Landscape Analysis. In Ryszard Klempous, editor, *Recent Advances in Intelligent Engineering Systems*, volume 378 of *Studies in Computational Intelligence Ser*, pages 161–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[102] Raphael P. Prager. pflacco: The R-package flacco in native Python code, 2023.

[103] Mike Preuss. Improved Topological Niching for Real-Valued Global Optimization. In *Applications of Evolutionary Computation*, volume 7248 of *Lecture Notes in Computer Science*, pages 386–395. Springer, Berlin and Heidelberg, 2012.

[104] Rajesh Rajamani. *Vehicle Dynamics and Control*. Mechanical engineering series. Springer, Boston, MA, 2 edition, 2012.

[105] Pratyusha Rakshit, Amit Konar, and Swagatam Das. Noisy evolutionary optimization algorithms – A comprehensive survey. *Swarm and Evolutionary Computation*, 33:18–45, 2017.

[106] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Dissertation, Technical University of Berlin, 1971.

[107] Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy. In Thomas Bäck, Mike Preuss, André. Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann, editors, *Parallel Problem Solving from Nature - PPSN XVI*, volume 12270 of *Lecture Notes in Computer Science*, pages 139–153. Springer International Publishing, Cham, 2020.

[108] Justin Rising. An Order-Theoretic Perspective on Rank Estimation. 2023.

[109] Sheldon M. Ross. *Introduction to Probability and Statistics for Engineers and Scientists*. Academic press, London, United Kingdom, 6 edition, 2021.

[110] Günter Rudolph. An evolutionary algorithm for integer programming. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature — PPSN III*, pages 139–148, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[111] J. Jon Ryu and Alankrita Bhatt. On Confidence Sequences for Bounded Random Processes via Universal Gambling Strategies. Technical report, 2022.

[112] Francisco J. Samaniego. *A Comparison of the Bayesian and Frequentist Approaches to Estimation*. Springer New York, New York, NY, 2010.

[113] Erhard Schmidt. Zur Theorie der linearen und nichtlinearen Integralgleichungen. *Mathematische Annalen*, 63(4):433–476, 1907.

[114] Dieter Schramm, Manfred Hiller, and Roberto Bardini. *Vehicle Dynamics: Modeling and Simulation*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2 edition, 2018.

[115] Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung*. Dissertation, Technical University of Berlin, 1975.

[116] Shubhanshu Shekhar, Ziyu Xu, Zachary Lipton, Pierre Liang, and Aaditya Ramdas. Risk-limiting financial audits via weighted sampling without replacement. *Uncertainty in Artificial Intelligence*, pages 1932–1941, 2023.

[117] Jaehyeok Shin, Aaditya Ramdas, and Alessandro Rinaldo. Are sample means in multi-armed bandits positively or negatively biased? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc, 2019.

[118] Jaehyeok Shin, Aaditya Ramdas, and Alessandro Rinaldo. On the Bias, Risk, and Consistency of Sample Means in Multi-armed Bandits. *SIAM Journal on Mathematics of Data Science*, 3(4):1278–1300, 2021.

[119] Siemens Digital Industries Software. Tire Simulation & Testing.

[120] Urban Škvorc, Tome Eftimov, and Peter Korošec. Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. *Applied Soft Computing*, 90, 2020.

[121] I. M. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[122] Peter F. Stadler. Fitness landscapes. In Michael Lässig and Angelo Valleriani, editors, *Biological evolution and statistical physics*, volume 585 of *Physics and astronomy online library*, pages 183–204. Springer, Berlin and Heidelberg, 2002.

[123] The MathWorks, Inc. MATLAB and Simulink.

[124] André Thomaser, Jacob de Nobel, Diederick Vermetten, Furong Ye, Thomas Bäck, and Anna V. Kononova. When to Be Discrete: Analyzing Algorithm Performance on Discretized Continuous Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, pages 856–863, New York, NY, USA, 2023. Association for Computing Machinery.

[125] André Thomaser, Anna V. Kononova, Marc-Eric Vogt, and Thomas Bäck. One-Shot Optimization for Vehicle Dynamics Control Systems: Towards Benchmarking and Exploratory Landscape Analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, pages 2036–2045, New York, NY, USA, 2022. Association for Computing Machinery.

[126] André Thomaser, Marc-Eric Vogt, Thomas Bäck, and Anna V. Kononova. Optimizing CMA-ES with CMA-ES. In *Proceedings of the 15th International Joint Conference on Computational Intelligence*, pages 214–221. SCITEPRESS - Science and Technology Publications, 2023.

[127] André Thomaser, Marc-Eric Vogt, Thomas Bäck, and Anna V. Kononova. Real-World Optimization Benchmark from Vehicle Dynamics: Specification of Problems in 2D and Methodology for Transferring (Meta-)Optimized Algorithm Parameters. In *Proceedings of the 15th International Joint Conference on Computational Intelligence*, pages 31–40. SCITEPRESS - Science and Technology Publications, 2023.

[128] André Thomaser, Marc-Eric Vogt, Anna V. Kononova, and Thomas Bäck. Transfer of Multi-objectively Tuned CMA-ES Parameters to a Vehicle Dynamics Problem. In Michael Emmerich, André Deutz, Hao Wang, Anna V. Kononova, Boris Naujoks, Ke Li, Kaisa Miettinen, and Iryna Yevseyeva, editors, *Evolutionary Multi-Criterion Optimization*, pages 546–560, Cham, 2023. Springer Nature Switzerland.

[129] Ye Tian, Shichen Peng, Xingyi Zhang, Tobias Rodemann, Kay Chen Tan, and Yaochu Jin. A Recommender System for Metaheuristic Algorithms for Continuous Optimization Based on Deep Recurrent Neural Networks. *IEEE Transactions on Artificial Intelligence*, 1(1):5–18, 2020.

[130] Tea Tušar, Dimo Brockhoff, and Nikolaus Hansen. Mixed-Integer Benchmark Problems for Single- and Bi-Objective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pages 718–726, New York, NY, USA, 2019. Association for Computing Machinery.

[131] Sander van Rijn, Hao Wang, Matthijs van Leeuwen, and Thomas Bäck. Evolving the Structure of Evolution Strategies. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2016.

[132] Sander van Rijn, Hao Wang, Bas van Stein, and Thomas Bäck. Algorithm Configuration Data Mining for CMA Evolution Strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 737–744, New York, NY, USA, 2017. Association for Computing Machinery.

[133] VIRES Simulationstechnologie GmbH. Opencrg.

[134] Hao Wang, Michael Emmerich, and Thomas Bäck. Mirrored Orthogonal Sampling with Pairwise Selection in Evolution Strategies. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 154–156, New York, NY, USA, 2014. Association for Computing Machinery.

[135] Hongjian Wang and Aaditya Ramdas. Catoni-style confidence sequences for heavy-tailed mean estimation. *Stochastic Processes and their Applications*, 163:168–202, 2023.

[136] Ian Waudby-Smith and Aaditya Ramdas. Estimating means of bounded random variables by betting. Technical report, 2020.

[137] Simon Wessing. Repair Methods for Box Constraints Revisited. In Anna I. Esparcia-Alcázar, editor, *Applications of Evolutionary Computation*, pages 469–478, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[138] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[139] Furong Ye, Carola Doerr, Hao Wang, and Thomas Bäck. Automated Configuration of Genetic Algorithms by Tuning for Anytime Performance. *IEEE Transactions on Evolutionary Computation*, 26(6):1526–1538, 2022.

[140] Gongbo Zhang, Yijie Peng, Jianghua Zhang, and Enlu Zhou. Dynamic Sampling Policy For Subset Selection. In *2021 Winter Simulation Conference (WSC)*, pages 1–12, 2021.

[141] Gongbo Zhang, Yijie Peng, Jianghua Zhang, and Enlu Zhou. Asymptotically Optimal Sampling Policy for Selecting Top-m Alternatives. *INFORMS Journal on Computing*, 35(6):1261–1285, 2023.

[142] Meng Zhao and Jinlong Li. Tuning the hyper-parameters of CMA-ES with tree-structured Parzen estimators. In *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pages 613–618, 2018.

List of Abbreviations

ABS	Antilock Braking System
AF	artificial function
ARP	Active Rollover Protection
AUC	Area Under the Curve
BBOB	Black-Box Optimization Benchmarking
BIPOP	bi-population
CI	Confidence Interval
CIS	Confidence Interval Sequence
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CMA-ESwM	CMA-ES with Margin
CRG	curved regular grid
CS	Confidence Set
CSA	cumulative step length adaptation
CV	characteristic value
DA	Dynamic Allocation
EA	Evolutionary Algorithm
ECDF	Empirical Cumulative Distribution Function
ELA	Exploratory Landscape Analysis
ERT	Expected Running Time
ES	Evolutionary Strategy
ESC	Electronic Stability Control
int-EA	Evolutionary Algorithm for Integer Programming
IPOP	increasing population
PCA	Principal Component Analysis
PCR	Probability of Correct Ranking

PCS	Probability of Correct Selection
RaS	Ranking and Selection
RCR	ratio of correct ranking
RI	Rank Interval
RQ	research question
SA	Static Allocation
SMAC	Sequential Model-based Algorithm Configuration
SMBO	sequential model-based optimization
SR	Success Rate
SWD	Sine with dwell
TPE	Tree-structured Parzen Estimator
UQiR	Uncertainty Quantification in Ranking
UQiRS	Uncertainty Quantification in Ranking and Selection
UQiS	Uncertainty Quantification in Selection
VDCS	vehicle dynamics control system
WoR	Without Replacement

Samenvatting

Deze dissertatie onderzoekt optimalisatiealgoritmes voor het oplossen van complexe en rekenintensieve optimalisatieproblemen uit verschillende domeinen. Als voorbeeld van een realistisch optimalisatieprobleem beschouwen we het ontwerpen van parameters voor voertuig-dynamische regelsystemen. De toenemende concurrentie in de auto-industrie vereist een op maat gemaakte, snelle ontwikkeling van technologisch geavanceerde voertuigen. Daarom worden de computationeel dure state-of-the-art simulatietechnologieën gecombineerd met optimalisatiealgoritmes.

Dit proefschrift richt zich op één centrale vraag: Hoe kan een optimalisatiealgoritme effectief en efficiënt worden ontworpen om computationeel dure problemen in de echte wereld op te lossen? Deze centrale vraag leidt tot een aantal verdere onderzoeksvragen die het hele ontwerpproces omvatten, van het wiskundige definiëren van de problemen als functies die geoptimaliseerd kunnen worden tot het kwantificeren van onzekerheid in optimalisatie.

Een belangrijke praktische bijdrage van dit proefschrift is de creatie van een dataset die instanties van optimalisatieproblemen uit het veld van voertuigdynamica representeert. Deze dataset vermindert de rekeninspanning die gepaard gaat met complexe simulaties en maakt efficiënte benchmarking van algoritmes mogelijk. Dit vergemakkelijkt de ontwikkeling van geavanceerde optimalisatiealgoritmes voor specifieke uitdagingen in de auto-industrie in een rekenkundig efficiënt en methodologisch robuust kader.

Een centraal aspect van het onderzoek is de ontwikkeling van een meta-optimalisatie methodiek voor het afstemmen van de parameters van de Covariance Matrix Adaptation Evolution Strategy (CMA-ES). De voorgestelde methode omvat het genereren van surrogaat-optimalisatieproblemen die de complexiteit van de oorspronkelijke optimalisatieprobleemlandschappen weerspiegelen. Vervolgens worden de gegenereerde surrogaat-optimalisatieproblemen gebruikt als afstemmingsreferenties

voor de meta-optimalisatie. De uitgevoerde experimenten tonen het potentieel van de aanpak aan voor brede toepassing op verschillende optimalisatieproblemen.

Verder worden de effecten van discretisatie op de prestaties van optimalisatiealgoritmes besproken. Een CMA-ES variant die discrete variabelen kan verwerken wordt onderzocht op verschillende discretisatieniveaus en probleemdimensies. Aanbevelingen voor het aanpassen van de parameters worden gegeven om de prestaties te verbeteren.

Daarnaast wordt een dynamische toewijzingsmethode ontwikkeld om dynamisch evaluaties toe te wijzen aan individuen binnen een populatie. Deze methode kwantificeert de onzekerheid in de selectie van de top individuen. Door deze methode te combineren met CMA-ES wordt het aantal vereiste evaluaties aanzienlijk verminderd. Dit verbetert de rekenefficiëntie in onzekere optimalisatieomgevingen.

Tot slot wordt het potentieel voor het toepassen van het meta-optimalisatie framework op verschillende probleemklassen en industrieën besproken. Bovendien kan de efficiëntie nog verder worden verbeterd, bijvoorbeeld door de rekeninspanning voor evaluaties en functieberekeningen te minimaliseren en een database te ontwikkelen om informatie van eerdere optimalisaties te gebruiken. Daarnaast is de uitbreiding naar multi-objectieve optimalisatie een veelbelovend gebied voor toekomstig onderzoek.

Summary

This thesis investigates the design of optimization algorithms for solving complex and computationally expensive optimization problems from various domains. As an example of a real-world optimization problem, the task of designing parameters for vehicle dynamics control systems is considered. The increasing competition in the automotive industry requires the tailored, swift development of technologically sophisticated vehicles. Therefore, the computationally expensive state-of-the-art simulation technologies are combined with optimization algorithms.

The thesis focuses on one central question: How can an optimization algorithm be effectively and efficiently designed to solve computationally expensive real-world problems? This central question leads to a number of further research questions that cover the design process from the mathematical definition of the objective function of real-world problems to the complicated issues of quantifying uncertainty in optimization.

An important practical contribution of this thesis is the creation of a dataset that represents real-world optimization problem instances from the field of vehicle dynamics. This dataset reduces the computational effort associated with complex simulations and enables efficient benchmarking of algorithms. This facilitates the development of advanced optimization algorithms for specific challenges in the automotive industry in a computationally efficient and methodologically robust framework.

A central aspect of the research is the development of a meta-optimization framework for tuning the parameters of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). The proposed method involves the generation of surrogate optimization problems that mirror the complexity of the original optimization problem landscapes. Then, the generated surrogate optimization problems are used as tuning references for the meta-optimization. The experiments conducted demonstrate the potential of the approach for broad application to various optimization problems.

Furthermore, the effects of discretization on the performance of optimization algorithms are discussed. A CMA-ES variant able of handling discrete variables is investigated across different discretization levels and problem dimensions. Recommendations for adjusting the parameters are provided to improve performance.

In addition, a dynamic allocation method is developed to dynamically assign evaluations to individuals within a population. This method quantifies the uncertainty in the selection of the top individuals. Combining this method with CMA-ES significantly reduces the number of evaluations required. This improves computational efficiency in uncertain optimization environments.

Finally, the potential for applying the meta-optimization framework to different problem classes and industries is discussed. Moreover, further efficiency improvements are achievable, for example, by minimizing the computational effort associated with evaluations and feature computations as well as developing a database to leverage information from previous optimizations. In addition, the extension to multi-objective optimization is a promising area for future research.

Curriculum Vitae

André Thomaser was born in Munich, Germany in 1996. He received a Bachelor's degree in Mechanical Engineering in 2018 and a Master's degree in Automotive Engineering in 2020 at the Technical University of Munich (TUM). In 2021, he joined the ProMotion Program at BMW with a focus on state-of-the-art optimization algorithms for the the virtual design of vehicle dynamics control systems. As a PhD student at the Leiden Institute of Advanced Computer Science (LIACS), he concentrated on the development of efficient and robust optimization algorithms, especially Evolutionary Strategies, for real-world optimization problems. His research studies were supervised by Thomas Bäck and Anna Kononova. Since 2024, André is a full-time engineer at BMW, where he continues to work on the development and application of optimization algorithms to design vehicle dynamics control systems.