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

Thomaser, A.M.

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.

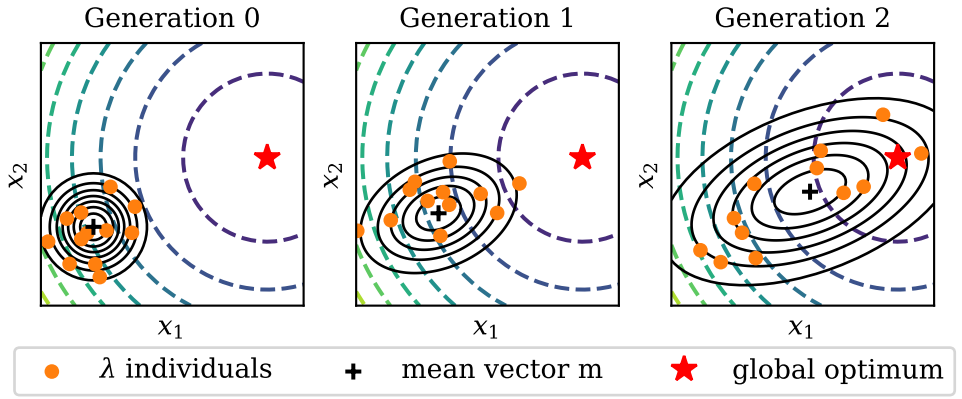

Figure 2.1: Visualization of a CMA-ES run with a population of 10 offspring on the two-dimensional sphere function for the first three generations.

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).

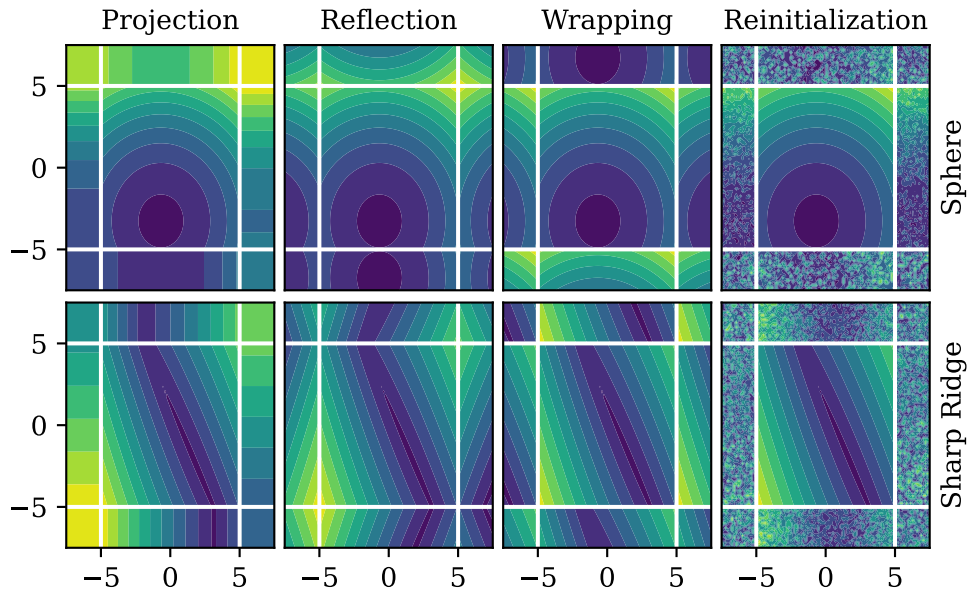

Figure 2.2: Landscape modification outside the feasible region $\mathcal{X} \in [-5, 5]^2$ for the two-dimensional sphere function and the sharp ridge function after the application of the four repair methods: projection, reflection, wrapping and reinitialization.

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).

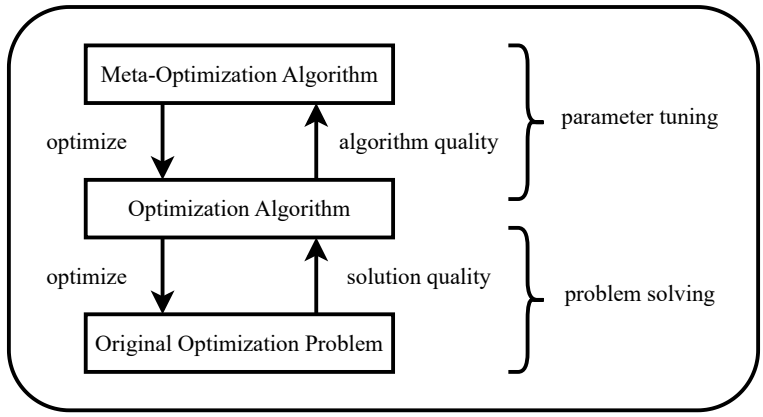

Figure 2.3: Schematic representation of the two optimization problems: (1) problem-solving, where an optimization algorithm seeks an optimal solution to the original optimization problem, and (2) parameter tuning, where a meta-optimization algorithm optimizes the parameters of the primary optimization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

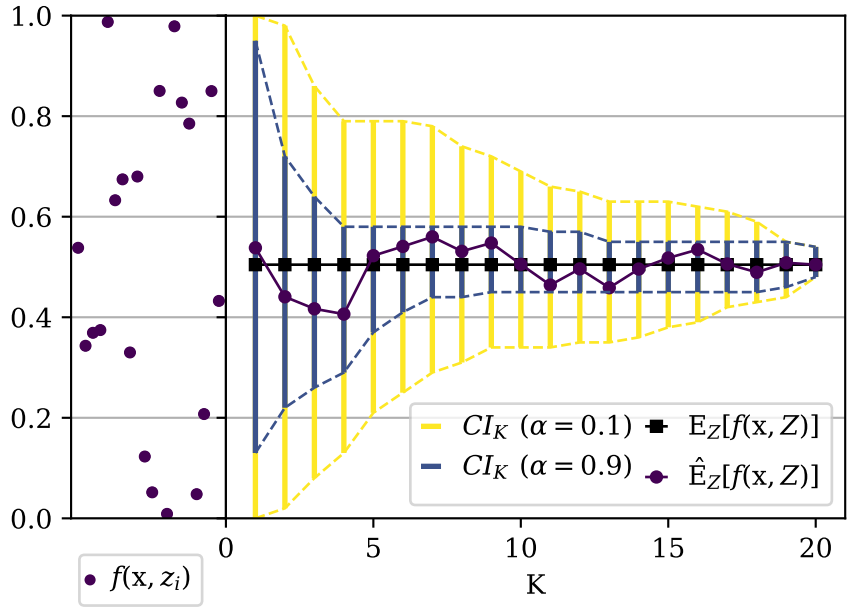

Figure 2.4: Left: The $|\Omega_Z| = 20$ uniformly distributed values $\{f(\mathbf{x}, z_1), \ldots, f(\mathbf{x}, z_{|\Omega_Z|})\}$. Right: Illustration of two CISs, each consisting of a CI after $K = 1, \ldots, 20$ evaluations for the significance levels $\alpha \in \{0.1, 0.9\}$. Each CI_K is constructed 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 $K = 20$.

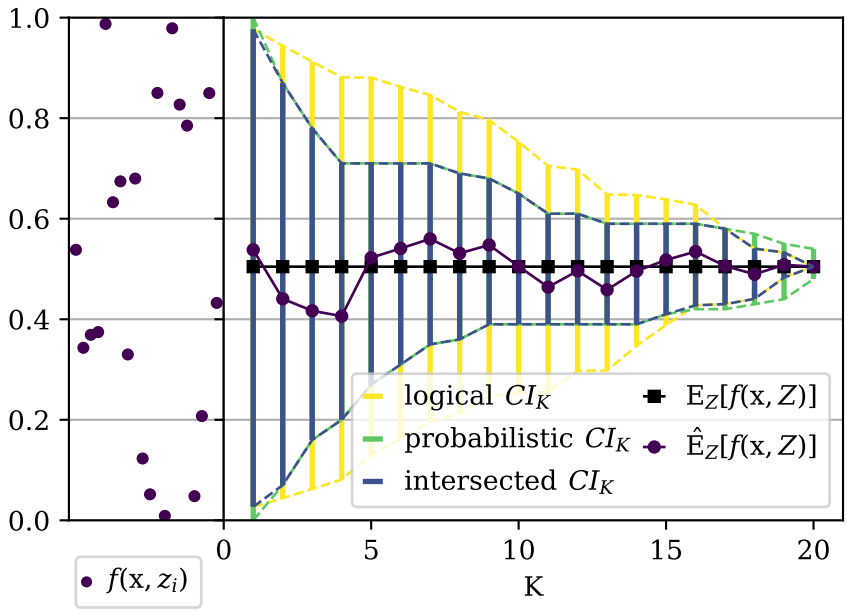

Figure 2.5: Left: The $|\Omega_Z| = 20$ uniformly distributed values $\{f(\mathbf{x}, z_1), \ldots, f(\mathbf{x}, z_{|\Omega_Z|})\}$. Right: Illustration of the resulting probabilistic CIS (Equation 2.29) and logical CIS (Equation 2.33), as well as the intersection, which is also a valid CIS. Each CI_K 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:

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

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.

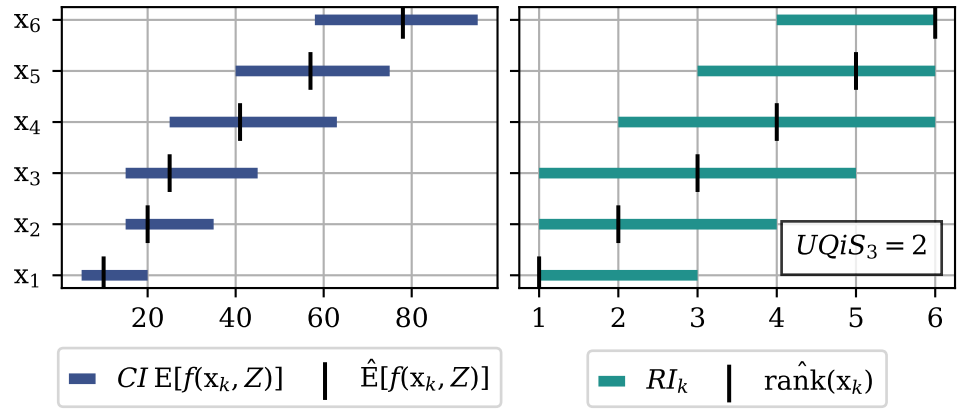

Figure 2.6: *Left*: CI for the true mean and sample mean for each individual in $\{x_1, \dots, x_6\}$. *Right*: Resulting RIs and sample ranks. The UQiS of the top-3 individuals $UQiS_3$ equals 2.