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.

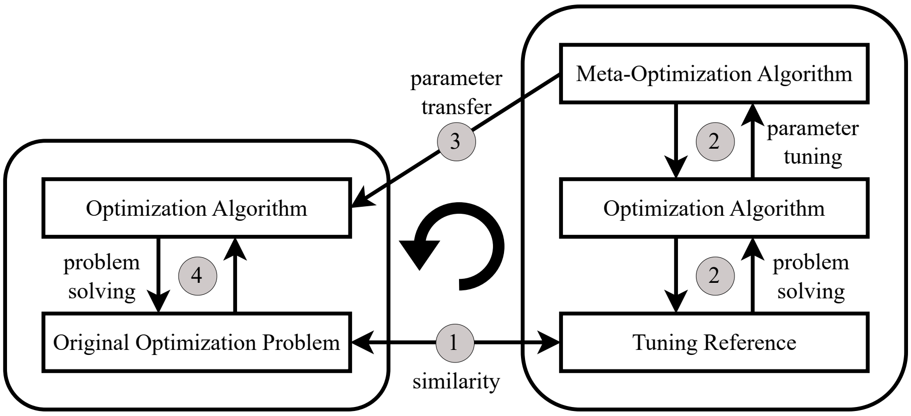


Figure 4.1: Schematic representation of the methodology for parameter tuning by employing computationally inexpensive tuning references similar to the computationally expensive original optimization problem.

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.

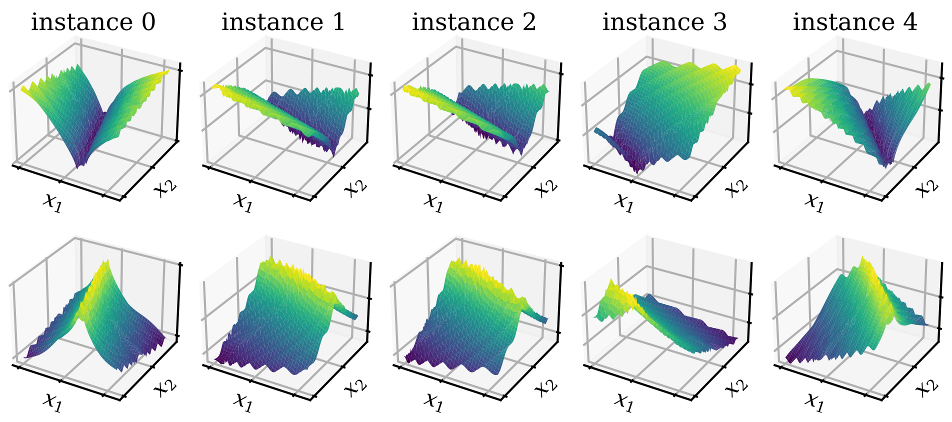


Figure 4.2: Five instances of a two-dimensional randomly generated artificial function (top row) and their negative counterparts (bottom row).

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.

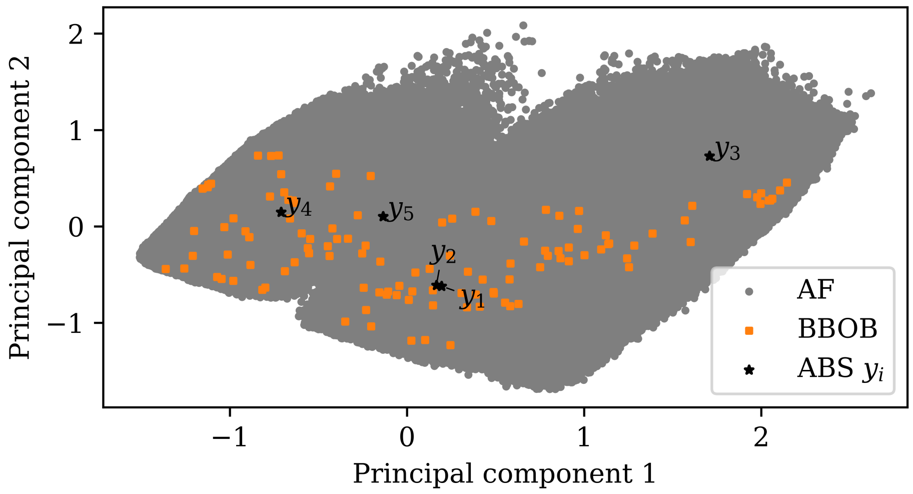
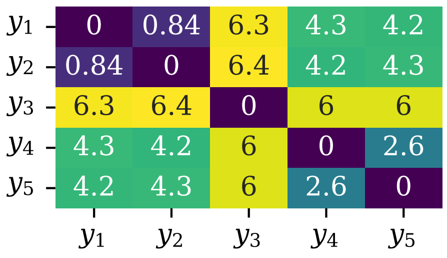


Figure 4.3: Positions defined by the two main PCA components of the 50 ELA features for each artificial function (AF), the BBOB functions and the ABS real-world problems y_i.

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.



	y_1	y_2	y_3	y_4	y_5
y_1	0	0.84	6.3	4.3	4.2
y_2	0.84	0	6.4	4.2	4.3
y_3	6.3	6.4	0	6	6
y_4	4.3	4.2	6	0	2.6
y_5	4.2	4.3	6	2.6	0

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.

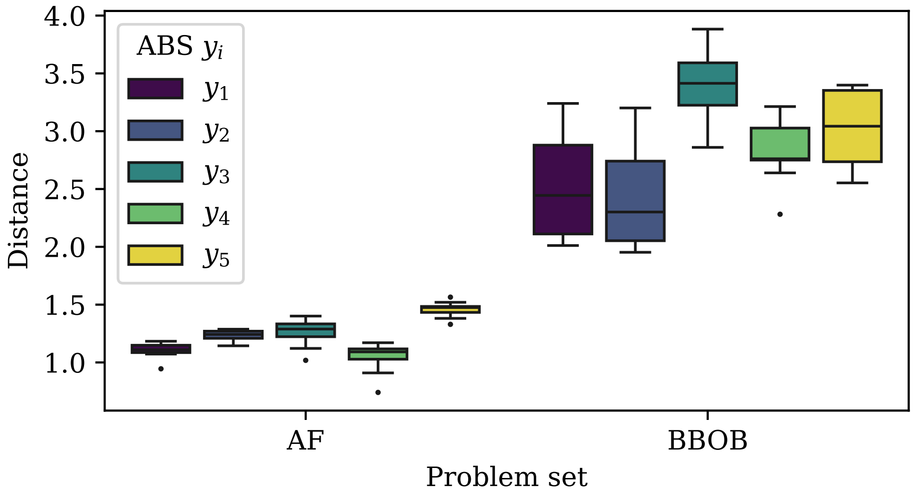


Figure 4.5: Similarity distance between the five real-world problems y_i and their ten most similar artificial functions (AF) and BBOB functions (as defined in Equation 4.1).

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 $AF_{17523,4}$ and $AF_{27980,4}$.

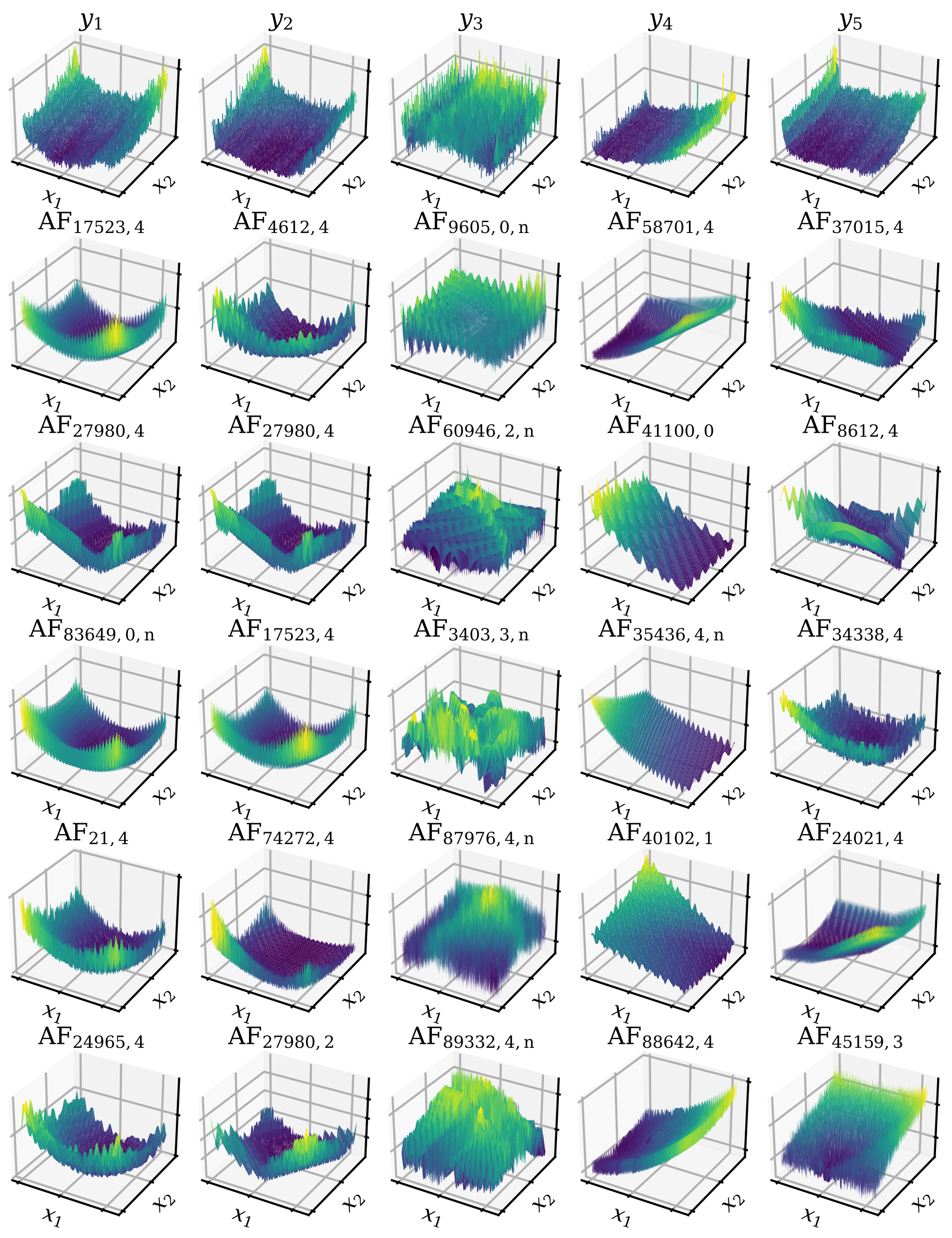


Figure 4.6: The landscapes of the five two-dimensional real-world problems y_i and the five most similar artificial functions (AF) to each problem y_i. The objective is minimization. Thus, dark blue-purple indicates better solutions and yellow worse.

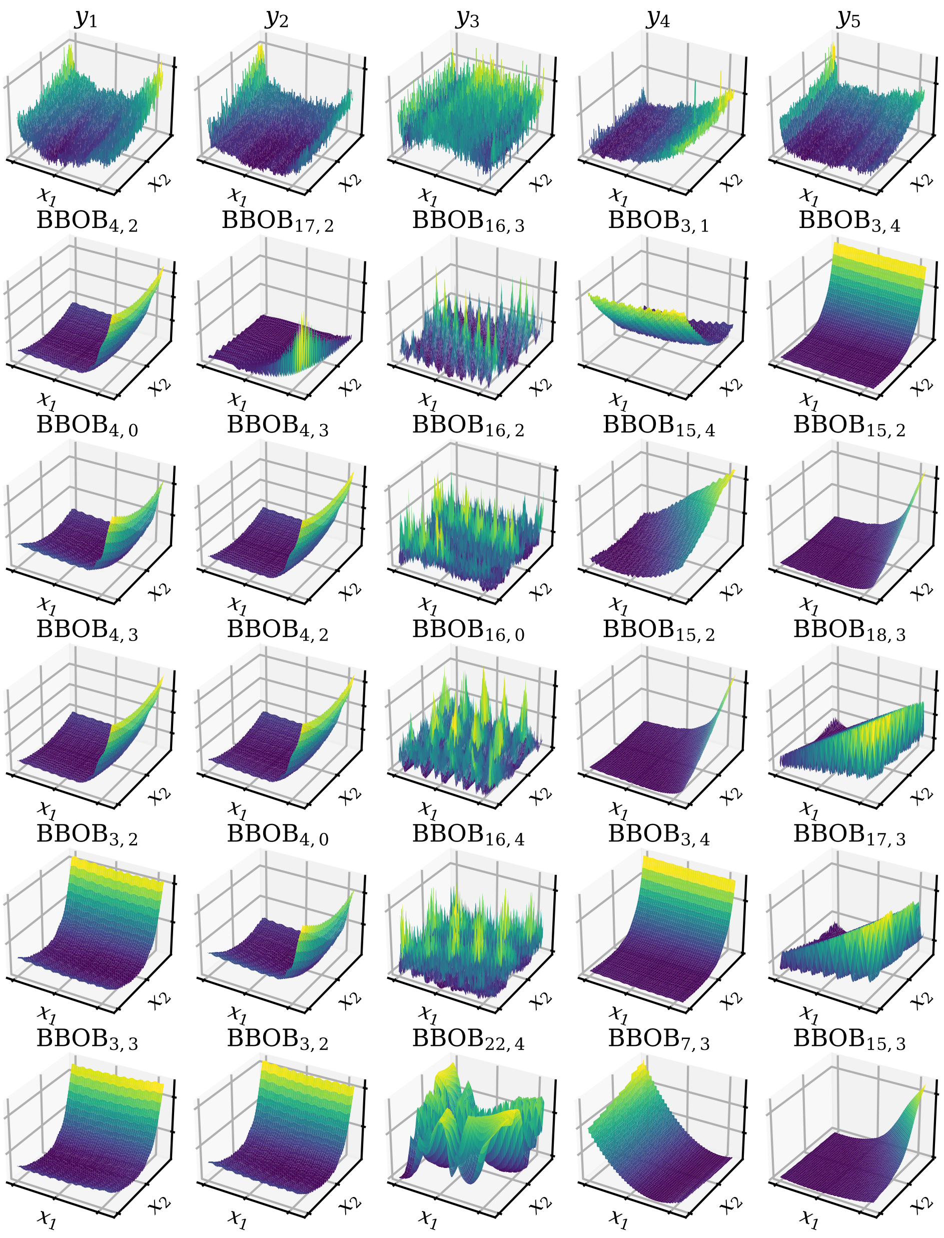


Figure 4.7: The landscapes of the five two-dimensional real-world problems y_i and the five most similar BBOB functions to each problem y_i. The objective is minimization. Thus, dark blue-purple indicates better solutions and yellow worse.

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.

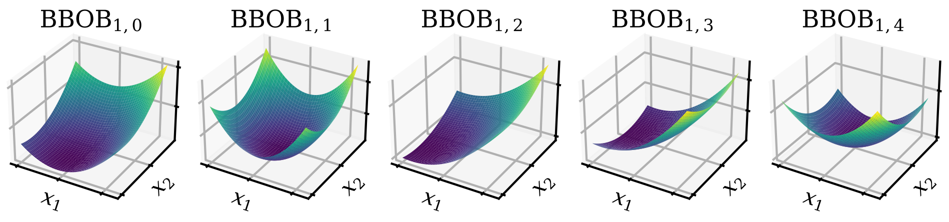


Figure 4.8: The landscapes of the first five instances of the sphere function. The objective is minimization. Thus, dark blue-purple indicates better solutions and yellow worse.

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.

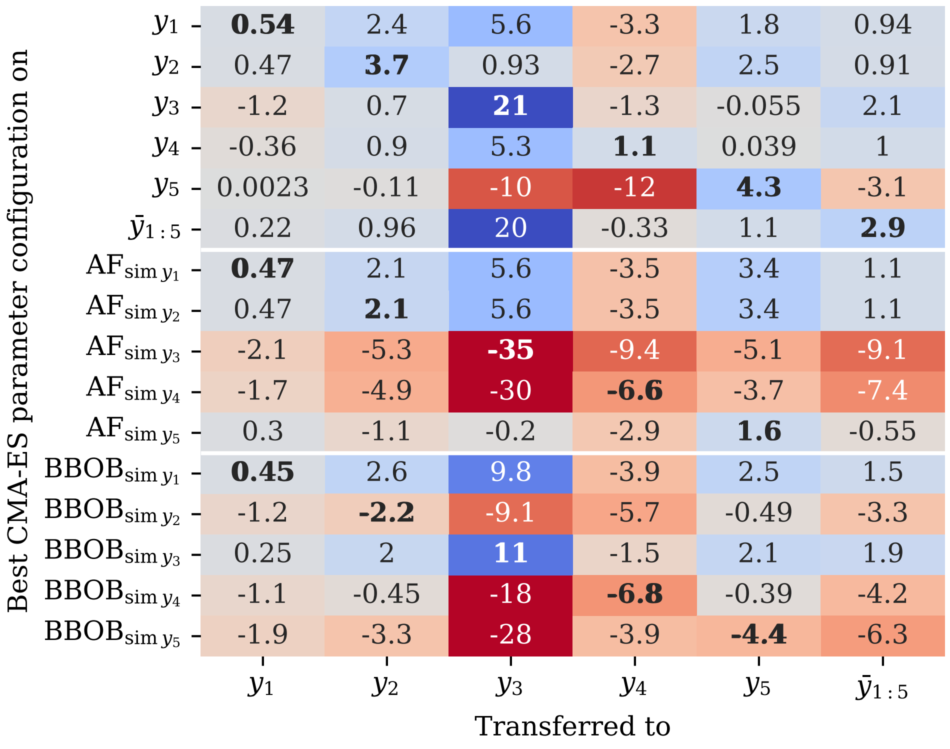
To evaluate the performance of a parameter configuration on the two-dimensional problems, the mean AUC value is calculated over 200 independent runs, each with an evaluation budget of 2 000 evaluations. For the artificial functions, an exhaustive search is conducted using CMA-ES with the default parameter 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.



	y_1	y_2	y_3	y_4	y_5	$\bar{y}_{1:5}$
y_1	**0.54**	2.4	5.6	-3.3	1.8	0.94
y_2	0.47	**3.7**	0.93	-2.7	2.5	0.91
y_3	-1.2	0.7	**21**	-1.3	-0.055	2.1
y_4	-0.36	0.9	5.3	**1.1**	0.039	1
y_5	0.0023	-0.11	-10	-12	**4.3**	-3.1
$\bar{y}_{1:5}$	0.22	0.96	20	-0.33	1.1	**2.9**
$\mathrm{AF}_{\mathrm{sim}\,y_1}$	**0.47**	2.1	5.6	-3.5	3.4	1.1
$\mathrm{AF}_{\mathrm{sim}\,y_2}$	0.47	**2.1**	5.6	-3.5	3.4	1.1
$\mathrm{AF}_{\mathrm{sim}\,y_3}$	-2.1	-5.3	**-35**	-9.4	-5.1	-9.1
$\mathrm{AF}_{\mathrm{sim}\,y_4}$	-1.7	-4.9	-30	**-6.6**	-3.7	-7.4
$\mathrm{AF}_{\mathrm{sim}\,y_5}$	0.3	-1.1	-0.2	-2.9	**1.6**	-0.55
$\mathrm{BBOB}_{\mathrm{sim}\,y_1}$	**0.45**	2.6	9.8	-3.9	2.5	1.5
$\mathrm{BBOB}_{\mathrm{sim}\,y_2}$	-1.2	**-2.2**	-9.1	-5.7	-0.49	-3.3
$\mathrm{BBOB}_{\mathrm{sim}\,y_3}$	0.25	2	**11**	-1.5	2.1	1.9
$\mathrm{BBOB}_{\mathrm{sim}\,y_4}$	-1.1	-0.45	-18	**-6.8**	-0.39	-4.2
$\mathrm{BBOB}_{\mathrm{sim}\,y_5}$	-1.9	-3.3	-28	-3.9	**-4.4**	-6.3

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.

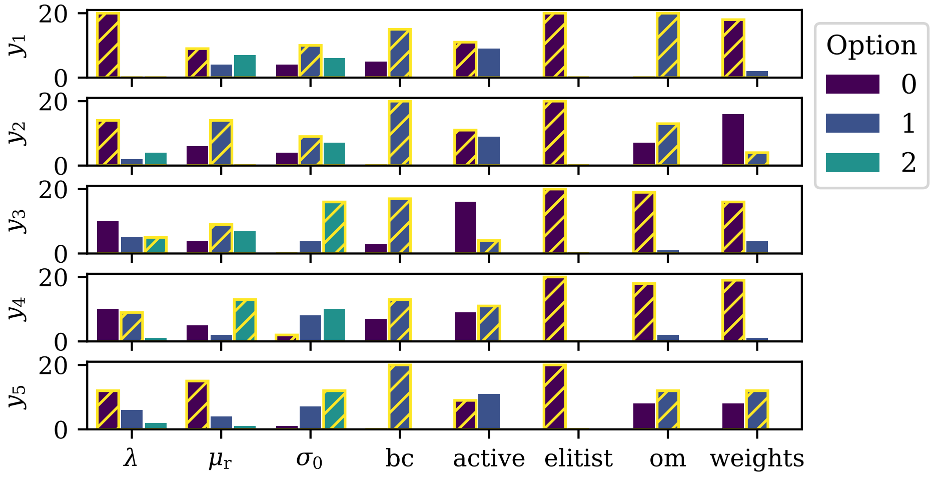


Figure 4.10: Frequency of occurrence of each parameter option given in Table 4.1 for every parameter within the best 20 configurations for each real-world problem y_i. Options corresponding to the best-performing configuration are highlighted with yellow hatching.

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.

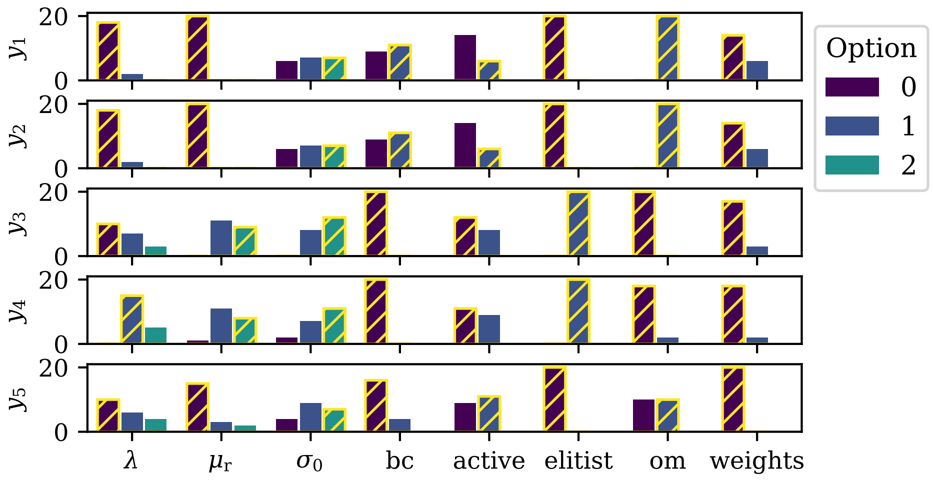


Figure 4.11: Frequency of occurrence of each parameter option given in Table 4.1 for every parameter within the best 20 configurations on the ten most similar artificial functions to each real-world problem y_i. Options corresponding to the best-performing configuration are highlighted with yellow hatching.

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.

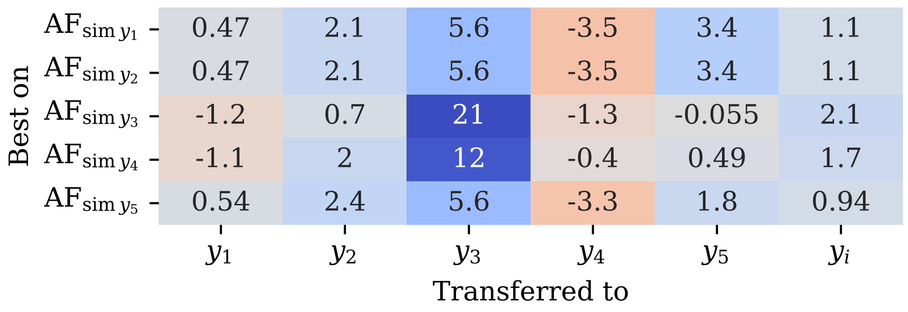


Figure 4.12: Relative improvement of the AUC to the default configuration when transferred to the five real-world problems and across all y_i. The performance of the best configuration, limited to those with a (μ, λ) selection strategy coupled with reflection, across the ten most similar artificial functions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

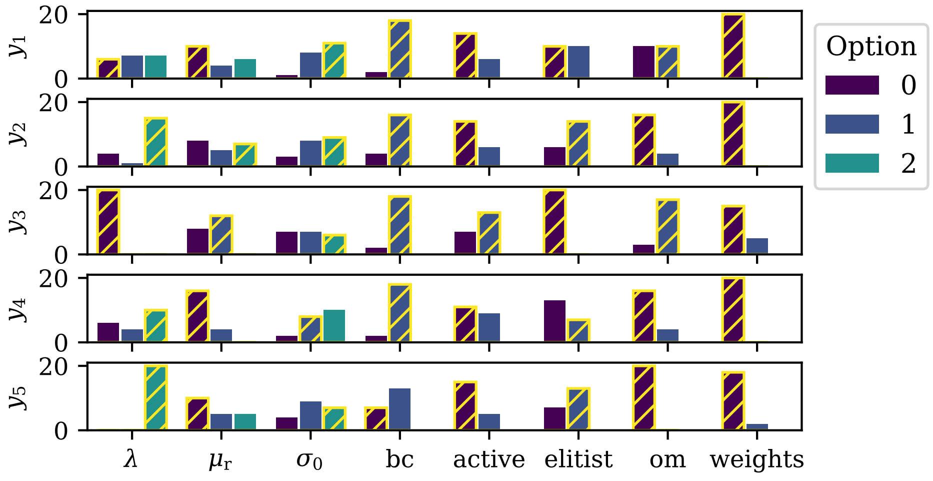


Figure 4.13: Frequency of occurrence of each parameter option given in Table 4.1 for every parameter within the best 20 configurations on the ten most similar BBOB functions to each real-world problem y_i. Options corresponding to the best-performing configuration are highlighted with yellow hatching.

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 $\mu_\text{r} = \frac{\mu}{\lambda}$	$\{\frac{1}{6}, \frac{2}{6}, ..., \frac{5}{6}\}$
σ_0	Initial standard deviation	$\{2, 3, \dots, 6\}$
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.

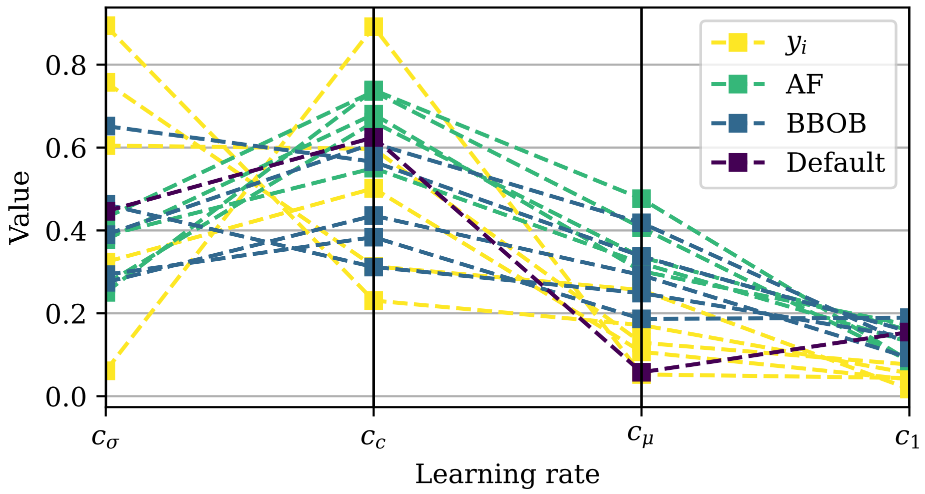


Figure 4.14: Values of the learning rates of the CMA-ES default configuration and the best configurations from five meta-optimization run on the five real-world problems y_i, the most similar artificial functions (AF) and the most similar BBOB functions.

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.

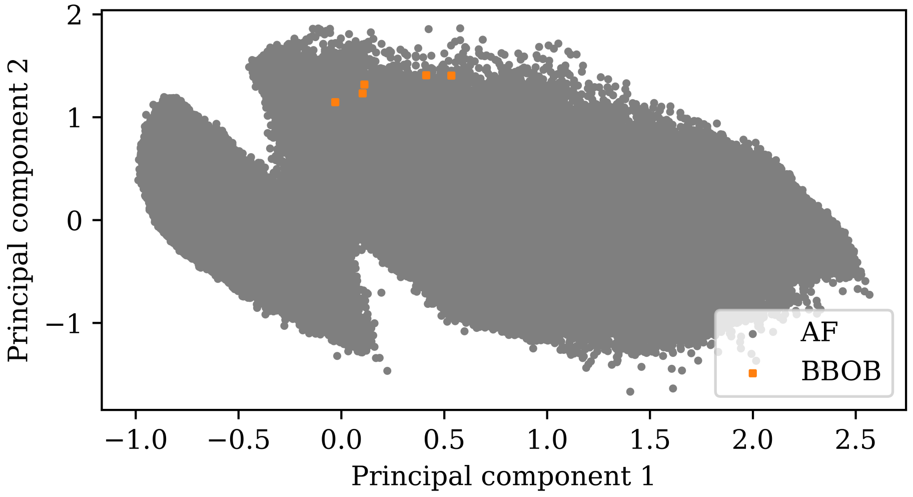


Figure 4.15: Positions defined by the two main PCA components of the 50 ELA features for each of the ten-dimensional artificial function (AF) and the five instances of the ten-dimensional Büche-Rastrigin function from the BBOB benchmark suite.

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.

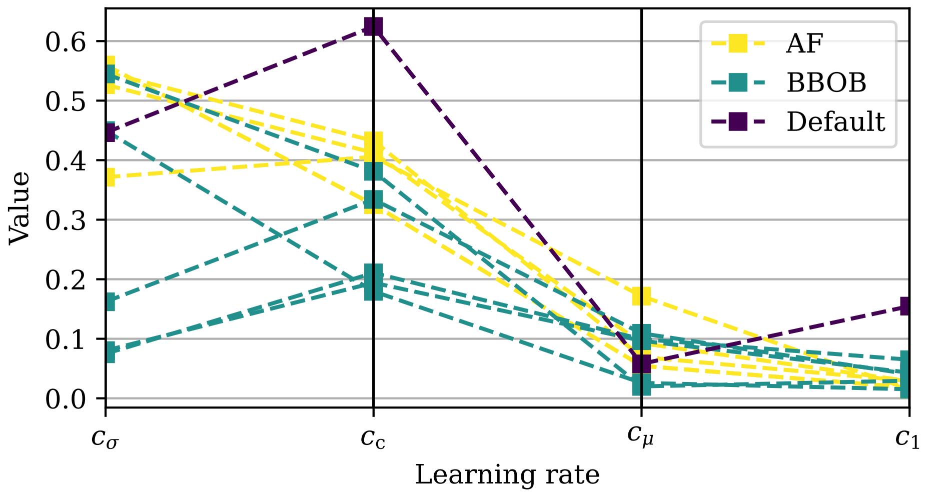


Figure 4.16: Values of the learning rates of the CMA-ES default configuration and the best configurations from five meta-optimization runs on the five instances of the ten-dimensional Büche-Rastrigin function from the BBOB benchmark suite and the most similar artificial functions (AF).

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.

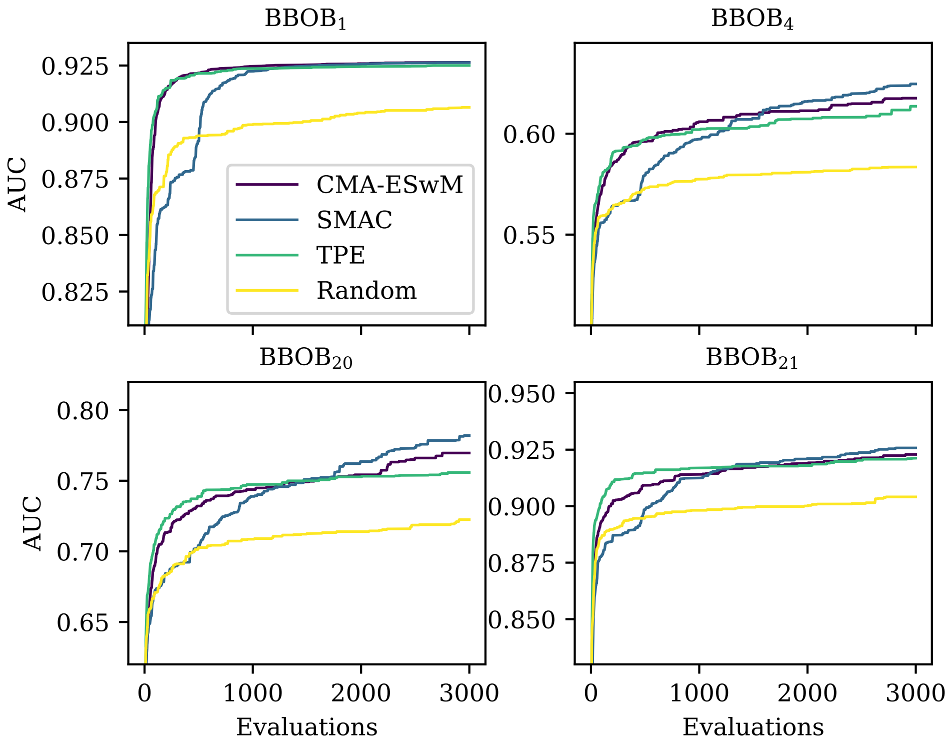


Figure 4.17: Median AUC values over evaluations of 50 runs for the four meta-optimization algorithms considered for tuning CMA-ES parameters on the four two-dimensional BBOB functions $BBOB_1, BBOB_4, BBOB_{20}, BBOB_{21}$.

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).

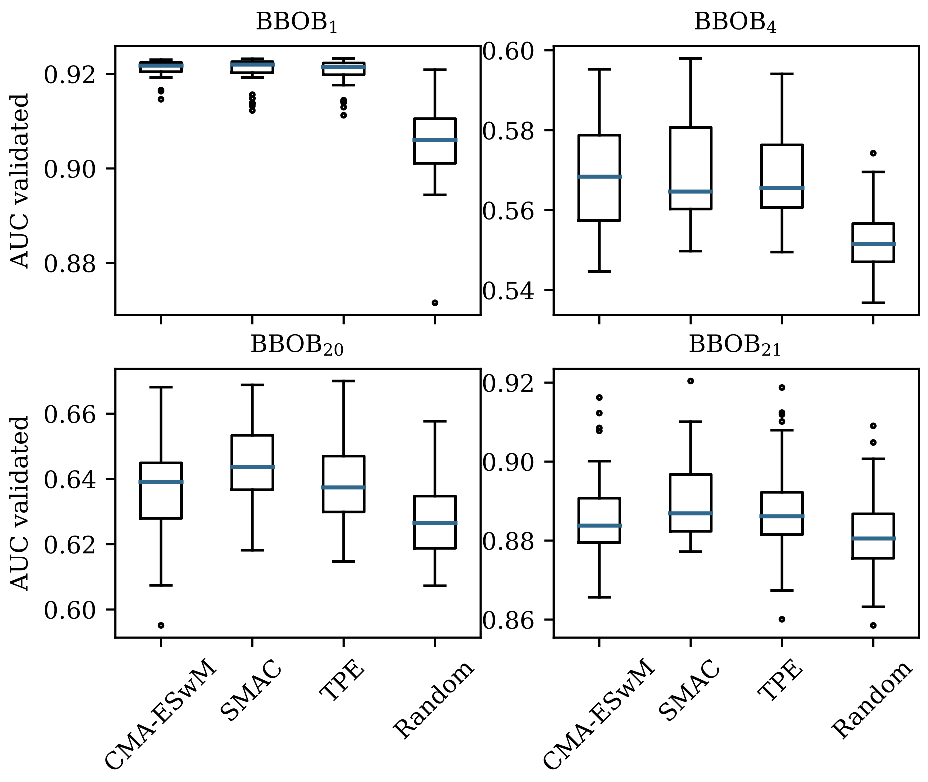


Figure 4.18: Boxplot of the validated AUC values of the best CMA-ES configurations found by the different meta-algorithms on each of the four BBOB functions considered. For each meta-algorithm and BBOB function, 50 parameter tuning runs were performed. Each configuration found in this process is, in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