Real-World Optimization Benchmark from Vehicle Dynamics: Specification of Problems in 2D and Methodology for Transferring (Meta-)Optimized Algorithm Parameters

André Thomaser[1,2][a], Marc-Eric Vogt[1][b], Thomas Bäck[2][c] and Anna V. Kononova[2][d]
[1]*BMW Group, Knorrstraße 147, Munich, Germany*
[2]*LIACS, Leiden University, Niels Bohrweg 1, Leiden, The Netherlands*

Keywords: Parameter Tuning, CMA-ES, Vehicle Dynamics Design, Benchmarking, Exploratory Landscape Analysis, Artificial Benchmarking Functions.

Abstract: The algorithm selection problem is of paramount importance in achieving high-quality results while minimizing computational effort, especially when dealing with expensive black-box optimization problems. In this paper, we address this challenge by using randomly generated artificial functions that mimic the landscape characteristics of the original problem while being inexpensive to evaluate. The similarity between the artificial function and the original problem is quantified using Exploratory Landscape Analysis. We demonstrate a significant performance improvement on five real-world vehicle dynamics problems by transferring the parameters of the Covariance Matrix Adaptation Evolution Strategy tuned to these artificial functions.
We provide a complete set of simulated values of braking distance for fully enumerated 2D design spaces of all five real-world optimization problems. So, replication of our results and benchmarking directly on the real-world problems is possible. Beyond the scope of this paper, this data can be used as a benchmarking set for multi-objective optimization with up to five objectives.

1 INTRODUCTION

Optimization algorithms play a vital role in solving complex real-world problems in a variety of domains, including engineering, finance, logistics, and machine learning. Evaluating and comparing these algorithms is critical to selecting the best approach for a given (class of) problem(s). This means running multiple full optimizations. Since real-world problems are often comparatively expensive, this can be archived by running the different algorithms on standardized, well-defined benchmark problems. While these benchmark problems, such as the black-box optimization benchmark suit (BBOB) (Hansen et al., 2009), are inexpensive to evaluate, they often cannot fully capture the intricacies and complexities present in real-world scenarios. Randomly generated artificial functions can be used to create more similar problems (Long et al., 2022).

The *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) (Hansen, 2016; Hansen and Ostermeier, 1996) is a class of iterative heuristic algorithms and is considered as state-of-the-art in single objective, continuous black-box optimization, with many successful real-world applications such as topology optimization (Fujii et al., 2018) and hyperparameter optimization of neural networks (Loshchilov and Hutter, 2016). Many variants have been developed over the years. The task of identifying and selecting the variant with the best performance for a given problem is a key challenge and is known as the *algorithm selection problem* (ASP) (Rice, 1976).

In this paper, we present five real-world 2D optimization problems from vehicle dynamics (Section 2). We have computed all possible combinations of the two enumerated design parameters. Thus, this set of values allows the replication of our results and benchmarking directly on the real-world problems. In the next step, we generate many suitable artificial functions and select the most similar ones to each real-world problem in terms of Exploratory

[a] https://orcid.org/0000-0002-6210-8784
[b] https://orcid.org/0000-0003-3476-9240
[c] https://orcid.org/0000-0001-6768-1478
[d] https://orcid.org/0000-0002-4138-7024

Thomaser, A., Vogt, M., Bäck, T. and Kononova, A.
Real-World Optimization Benchmark from Vehicle Dynamics: Specification of Problems in 2D and Methodology for Transferring (Meta-)Optimized Algorithm Parameters.
DOI: 10.5220/0012158000003595
In *Proceedings of the 15th International Joint Conference on Computational Intelligence (IJCCI 2023)*, pages 31-40
ISBN: 978-989-758-674-3; ISSN: 2184-3236

Landscape Analysis features (Section 3). These similar functions are then used as a tuning reference to which the parameters of the optimization algorithm are tuned before being applied to the original expensive real-world problem. As an optimization algorithm, we use the modular CMA-ES implementation with several different variants that can be combined arbitrarily (Section 4). Finally, we compare the performance on the 2D real-world problems of the tuned CMA-ES parameters on the artificial functions and BBOB functions with the directly tuned parameters on the 2D real-world problems (Section 5).

2 REAL-WORLD PROBLEMS

2.1 Description and Objective

Vehicle dynamics control systems, such as the Antilock Braking System (ABS) (Koch-Dücker and Papert, 2014), have revolutionized the automotive industry by enhancing vehicle safety and performance. The ABS mitigates the risk of wheel lock-up during braking, by adjusting the brake pressure to keep brake slip within an optimal range. This reduces the braking distance and allows the driver to maintain control and steer the vehicle even in an emergency braking situation. The parameters of these control systems must be carefully calibrated to achieve optimal performance under different scenarios and vehicle settings.

An industry standard maneuver for evaluating a vehicle's braking performance is the *emergency straight-line full-stop braking maneuver with ABS fully engaged* (International Organization for Standardization, 2007). The maneuver consists of the following three phases (Figure 1):

- 1: Accelerate the vehicle to 103.5 km/h;
- 2: No acceleration or deceleration until 103 km/h;
- 3: Apply brakes until the vehicle stops.

The braking distance can be calculated as the integral of the vehicle's longitudinal velocity over time from v_s = 100 km/h at time t_s to v_e = 0 km/h at time t_e (Figure 1). The beginning of the braking process between 103 km/h and 100 km/h is not taken into account when calculating the braking distance to avoid possible disturbances occurring at the beginning of the braking process. According to ISO 21994:2007, a measurement sequence for calculating the braking distance consists of ten valid single measurements. The braking distance y is defined as:

$$y = \frac{1}{10}\sum_{k=1}^{10}\int_{t_s}^{t_e} v_k(t)\,dt. \tag{1}$$

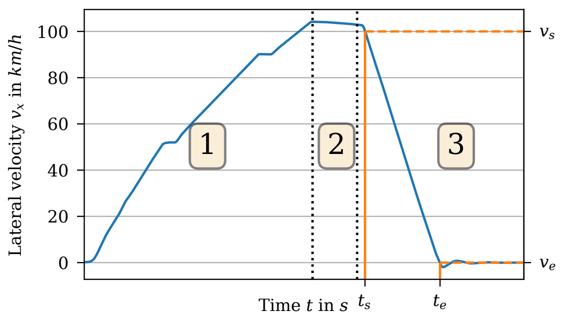


Figure 1: An emergency straight-line full-stop braking maneuver for calculating the braking distance from $\mathbf{v_s}$ = 100 km/h to $\mathbf{v_e}$ = 0 km/h (start time $\mathbf{t_s}$ and end time $\mathbf{t_e}$).

2.2 Simulation Environment

The vehicle dynamics, driver, and environment are simulated using a two-track model implemented in Simulink (The MathWorks, Inc., 2015). The vehicle dynamics part includes the mechanical vehicle and the control systems. The mechanical vehicle is modeled as a five-body model (car body and four wheels) with 16 degrees of freedom and consists of the following components: equation of motion, tires, drive train, aerodynamics, suspension, steering, and braking. The control system components are sensors, logic, and actuators. By virtually mapping the interaction between these components, the simulation is able to represent an integrated control system.

MF-Tyre/MF-Swift (Siemens Digital Industries Software, 2020), which is based on Pacejka's so-called Magic Formula (Pacejka and Bakker, 1992), is used to accurately simulate the steady-state and transient behavior of tires under slip conditions. The road surface is described by a curved regular grid (CRG) track (VIRES Simulationstechnologie GmbH, 2020), which provides information about the road width and elevation along a predefined reference line. CRG tracks are able to model 3D roads in great detail while keeping memory usage to a minimum. On a standard workstation[1], it takes about 15 minutes to simulate a braking maneuver. To reduce the wall clock time, the simulation is run in parallel on multiple computers.

2.3 Real-World Benchmark-Problems

There are two very important ABS control parameters that have a huge impact on the braking distance, referred to here as x_1 and x_2. Each parameter has a defined lower B_{lb} and upper B_{ub} bounds: $x_1 \in [-5, 6]$ and $x_2 \in [-5, 4]$. Furthermore, for x_1 and x_2 only a

[1]HP Workstation Z4 G4 Intel Xeon W-2125 4.00GHz/4.50GHz 8.25MB 2666 4C 32GB DDR4-2666 ECC SDRAM

Table 1: Explaination of the vehicle settings.

Name	Tires	Vehicle Load
y1	High performance	Partially loaded
y2	Medium performance	Partially loaded
y3	Under performance	Partially loaded
y4	High performance	Fully loaded
y5	High performance	Little loaded

set of values D_i with a resolution of 0.1 as equal distance between the values is allowed. This means that 111 distinct values are possible for x_1 and 91 for x_2. The 2-dimensional input space $\mathbb{D}^2 = \times_{i=1}^{2} D_i$ is then the corresponding cartesian product. Thus there are 10 101 possible combinations for the two ABS parameters x_1 and x_2.

By simulating every possible combination (brute forcing) of the two ABS parameters, we can fully learn the functional dependence between the two ABS parameters and the resulting braking distance $y(x)$. Furthermore, in order to apply algorithms for continuous input spaces, in the following we consider the problem as *quasi-continuous*: given an input $x \in \mathbb{R}^2$ within the lower and upper bounds, the corresponding braking distance $y(x)$ is determined by rounding x_1 and x_2 to the nearest data point within the 2-dimensional input space $\mathbb{D}^2 = \times_{i=1}^{2} D_i$.

In summary, the objective is to find a parameter setting x that minimizes the simulated braking distance $y(x)$, as defined by the equation (1):

$$\underset{x \in X}{\text{minimize}}\, y(x), X = \{x \in \mathbb{R}^2 : B_{lb} \leq x \leq B_{ub}\}. \quad (2)$$

Generally, an optimal parameter setting x^* is only optimal for one vehicle setting. In this paper, we provide the data of five different vehicle settings. Here, a setting consists of a vehicle load and a tire (Table 1).

Since we are interested in the distance to the global optimum, all braking distances of a particular vehicle setting i are specified as the distance in meters to the corresponding optimum. Figure 2 shows the distribution of the 10 101 data points for each real-world problem *yi*. The data with additional Python code for benchmarking has been made available on our Zenodo repository (Thomaser et al., 2023a).

Note that minimizing the braking distance for each vehicle setting is one objective. For each vehicle setting respectively objective function, a different parameter set (x_1, x_2) is optimal, and the objectives are in conflict with each other. Therefore, the provided data *can* be used for benchmarking multi-objective optimization algorithms with up to five objectives. However, in this paper, we focus only on *single-objective* optimization and do not investigate the conflict between the different vehicle settings.

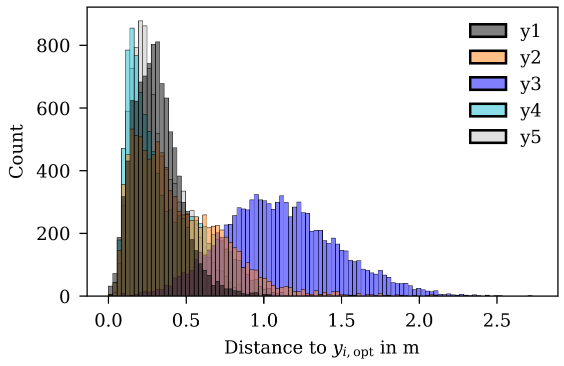


Figure 2: Distribution of the 10 101 data points for each of the five vehicle settings *yi* (Table 1).

3 TUNING REFERENCES

3.1 Benchmark Functions

Benchmarking algorithms is an important practice that facilitates the evaluation and comparison of algorithm performance. A widely accepted benchmark suite for single-objective optimization problems is the Black-Box Optimization Benchmark (BBOB) (Hansen et al., 2009).

In addition to the 24 BBOB functions, we randomly generate 100 000 artificial functions in 2D. Therefore, we use the Python implementation of (Long et al., 2022), which is based on (Tian et al., 2020). The input space is the same as for the five real-world problems *yi*, $x_1 \in [-5, 6]$, and $x_2 \in [-5, 4]$.

We use the instance-generating mechanism of the BBOB function suite (Hansen et al., 2009) and consider five instances of each function. We also apply the same mechanism four times to each of the 100 000 randomly generated artificial functions to obtain slightly different functions, that are simply rotated and shifted relative to the original function. Maximizing or minimizing a function has a huge impact on the landscape from the point of view of an optimization algorithm, so we also consider the inverse function (multiplied by -1) of each function. This results in a set of 1 000 000 different artificial functions.

3.2 Exploratory Landscape Analysis

To quantify high-level properties, such as global structure or multi-modality, of the landscape of an optimization problem we use *Exploratory Landscape Analysis* (ELA) (Mersmann et al., 2011; Mersmann et al., 2010). This allows us to compute the similarity between two optimization problems. The ELA-features can be computed primarily with a sample of

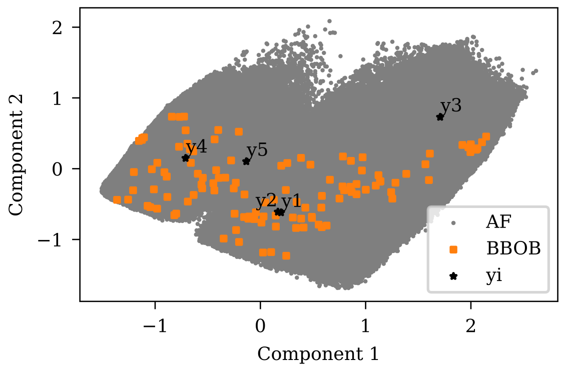


Figure 3: Scatter plot of the two main PCA components of the 50 ELA-features for the artificial functions (AF), the BBOB functions, and the real-world problems *yi*.

points of the objective function. The *pflacco* package (Prager, 2023), which provides a native Python implementation of the large collection of ELA features in the *flacco* package (Kerschke and Trautmann, 2019). We consider only those ELA features that can be computed cheaply without additional resampling. This results in 55 individual features, which are grouped in five sets: classical ELA (distribution, level, meta) (Mersmann et al., 2011), information content (Muñoz et al., 2015), dispersion (Lunacek and Whitley, 2006), nearest better clustering (Preuss, 2012; Kerschke et al., 2015) and principal component. Five features could not be computed for each function and were discarded.

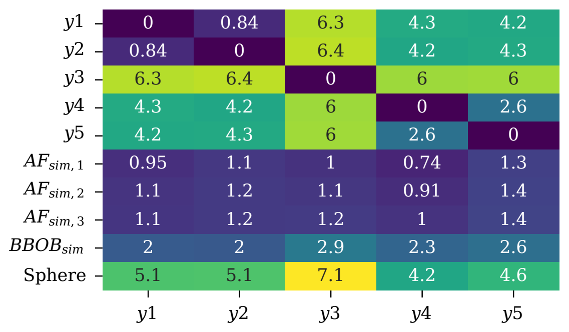
As a *compromise* between accuracy and computational effort in ELA, a sample size of 50 times the dimensionality is recommended to classify the BBOB functions with ELA features (Kerschke et al., 2016). To increase the accuracy we use a Sobol' design (Owen, 1998; Sobol', 1967) with 1000 samples (500×2). For equal weighting, the feature values are min-max-scaled to $[0, 1]$. Overall, the feature calculation was successful for 99.5% of the randomly generated artificial functions. Reasons for unsuccessful calculations are 'not a number' values as function values or a flat fitness function.

A large number of considered features leads to redundancy within the features (Škvorc et al., 2020). Therefore, to remove redundant features and reduce the dimensionality of the feature space, we perform *Principal Component Analysis* (PCA) (Jolliffe, 1986). With the requirement of a cumulative variance greater than 0.999, the dimensionality of the feature space is reduced to 31.

Furthermore, Figure 3 shows the position defined by the two main components from the PCA for each function considered. The BBOB functions cover only a partial space. The space in between is filled with the randomly generated artificial functions. The different vehicle settings of the real-world problem (Table 1) are relatively widely scattered. This indicates dissimilarities within the landscape of the vehicle settings. Only $y1$ and $y2$ are relatively close to each other.

Table 2: Most similar BBOB function to each of the real-world problems *yi*.

Name	Most similar BBOB Function
y1	Büche-Rastrigin Function f_4
y2	Büche-Rastrigin Function f_4
y3	Weierstrass Function f_{16}
y4	Rastrigin Function f_3
y5	Rastrigin Function f_3



	$y1$	$y2$	$y3$	$y4$	$y5$
$y1$	0	0.84	6.3	4.3	4.2
$y2$	0.84	0	6.4	4.2	4.3
$y3$	6.3	6.4	0	6	6
$y4$	4.3	4.2	6	0	2.6
$y5$	4.2	4.3	6	2.6	0
$AF_{sim,1}$	0.95	1.1	1	0.74	1.3
$AF_{sim,2}$	1.1	1.2	1.1	0.91	1.4
$AF_{sim,3}$	1.1	1.2	1.2	1	1.4
$BBOB_{sim}$	2	2	2.9	2.3	2.6
Sphere	5.1	5.1	7.1	4.2	4.6

Figure 4: Similarity of the landscape (as defined in equation 3) of the five real-world problems *yi* (x-axis) to each other, the Sphere function (BBOB function f_1), the three most similar artificial generated functions ($AF_{sim,i}$) and the most similar BBOB function ($BBOB_{sim}$) for each of the real-world problems.

3.3 Similar Functions

Based on the reduced set of features produced by PCA, we can quantify the similarity between two problems p_1 and p_2 via the city-block distance d between their feature vectors F_{p_1} and F_{p_2}:

$$d(p_1, p_2) = \sum_i |F_{p_1,i} - F_{p_2,i}|. \quad (3)$$

Using equation 3, we compute the distances between the five real-world problems *yi* and other functions. In the following, we examine the three most similar randomly generated artificial functions for each of the real-world problems, denoted as $AF_{sim,j}$, and for comparison also each the most similar BBOB function and the Sphere function (Table 2). Figure 4 shows these distances, which range from 0.74 to 7.1.

We observe that the distance between $y1$ and $y2$ is relatively small with a value of 0.84, for comparison the most similar BBOB function is for both the Büche-Rastrigin function with a distance of 2.0 and the distances to the other real-world problems $y3$, $y4$,

$y5$ are always greater than 4 (Figure 4). Thus $y1$ and $y2$ can be considered as similar to each other, while the other real-world problems are more dissimilar to $y1$ and $y2$ and also to each other.

Within the randomly generated artificial functions, the three most similar to each real-world problem yi all have a distance between 0.74 and 1.4. Figure 5 shows the actual landscape of real-world problems and these most similar artificial functions. Only tiny changes within the straight-line braking maneuvers can influence the braking distance, thus all real-world problems yi are overlaid with noise, which leads to a highly multi-modal landscape. This noise or multi-modality is also depicted by the similar artificially generated functions. Furthermore, we observe from Figure 5 that, with the exception of $y3$ (under performance tires), the real-world problems have a global structure. This global structure is very similar for $y1$ and $y2$ - as expected from the distance (Figure 4). The similar artificially generated functions also manage to capture such a global structure.

The Sphere function is very dissimilar to the real-world problems, especially to $y3$. This is not surprising, since the Sphere function is uni-modal and does not capture the noise within the landscape of the real-world problem. In the following, the Sphere function will be used as an example of a dissimilar function for comparison and validation.

4 TUNING CMA-ES

4.1 CMA-ES

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen, 2016; Hansen and Ostermeier, 1996) is a class of iterative heuristic algorithms for solving single objective, continuous optimization problems. A population x of CMA-ES consists of λ offspring and is sampled from a multivariate normal distribution with mean value $m^{(g)} \in \mathbb{R}^n$, covariance matrix $C^{(g)} \in \mathbb{R}^{n \times n}$ and standard deviation $\sigma^{(g)} \in \mathbb{R}_{>0}$:

$$x_k^{(g+1)} \sim m^{(g)} + \sigma^{(g)} \mathcal{N}(0, C^{(g)}) \quad \forall k = 1, ..., \lambda. \tag{4}$$

In each generation g of the CMA-ES, the best μ individuals are selected from the population to compute the new mean value $m^{(g+1)}$ with the given weights w_i:

$$m^{(g+1)} = m^{(g)} + c_m \sum_{i=1}^{\mu} w_i (x_{i:\lambda}^{(g+1)} - m^{(g)}). \tag{5}$$

Besides the landscape of the objective function, the performance of CMA-ES on a specific optimization problem is determined by the combination of several parameters and variants (Bäck et al., 2013). In this paper, we use the modular CMA-ES implementation (de Nobel et al., 2021; van Rijn et al., 2016) with several different variants that can be combined arbitrarily. For a restart of CMA-ES, we consider two variants: the population size is increased by a factor (IPOP) (Auger and Hansen, 2005) or alternated between a smaller and a larger population (BIPOP) (Hansen, 2009). Table 3 contains the considered parameters and variants for tuning. The total number of possible combinations of CMA-ES variants and parameters is 816480.

4.2 Meta-Algorithm

The task of finding the optimal parameters of an algorithm for solving an optimization problem such as the five real-world problems is difficult without further knowledge. For example, CMA-ES has a large number of different settings and variants that can be selected. To solve this algorithm selection problem (ASP) (Rice, 1976) an automatic parameter tuning has been proposed as a second optimization problem besides solving the original problem (Bäck, 1994; Grefenstette, 1986).

Figure 6 illustrates the relationship between the two optimization problems: solving the original problem and parameter tuning (Eiben and Smit, 2011). The former optimization problem consists of the original problem and the algorithm to find an optimal solution to this problem. The latter consists of a meta-algorithm to find optimal parameters for the algorithm to solve the original problem. The so-called *fitness* determines the quality of solutions for the original problem and the *utility* the quality of the parameters of the algorithm (Eiben and Smit, 2011).

When tuning the parameters of an optimization algorithm with a given budget, the obtained parameters are optimal only for that particular budget (Thomaser et al., 2023b). To avoid tuning the parameters to a specific budget, we use the area under the empirical cumulative distribution function curve AUC to calculate the utility of an optimization algorithm (Ye et al., 2022). The higher the AUC, the better the performance of the optimization algorithm, thus the objective is to maximize the AUC, or formulated as a minimization problem for the meta-algorithm: the objective is to minimize $1 - AUC$.

The evaluation budget for CMA-ES is 2000 and we perform 100 individual optimization runs. To compute the empirical cumulative distribution function curves, we consider 81 target values logarithmically distributed from 10^8 to 10^{-8}. This procedure is

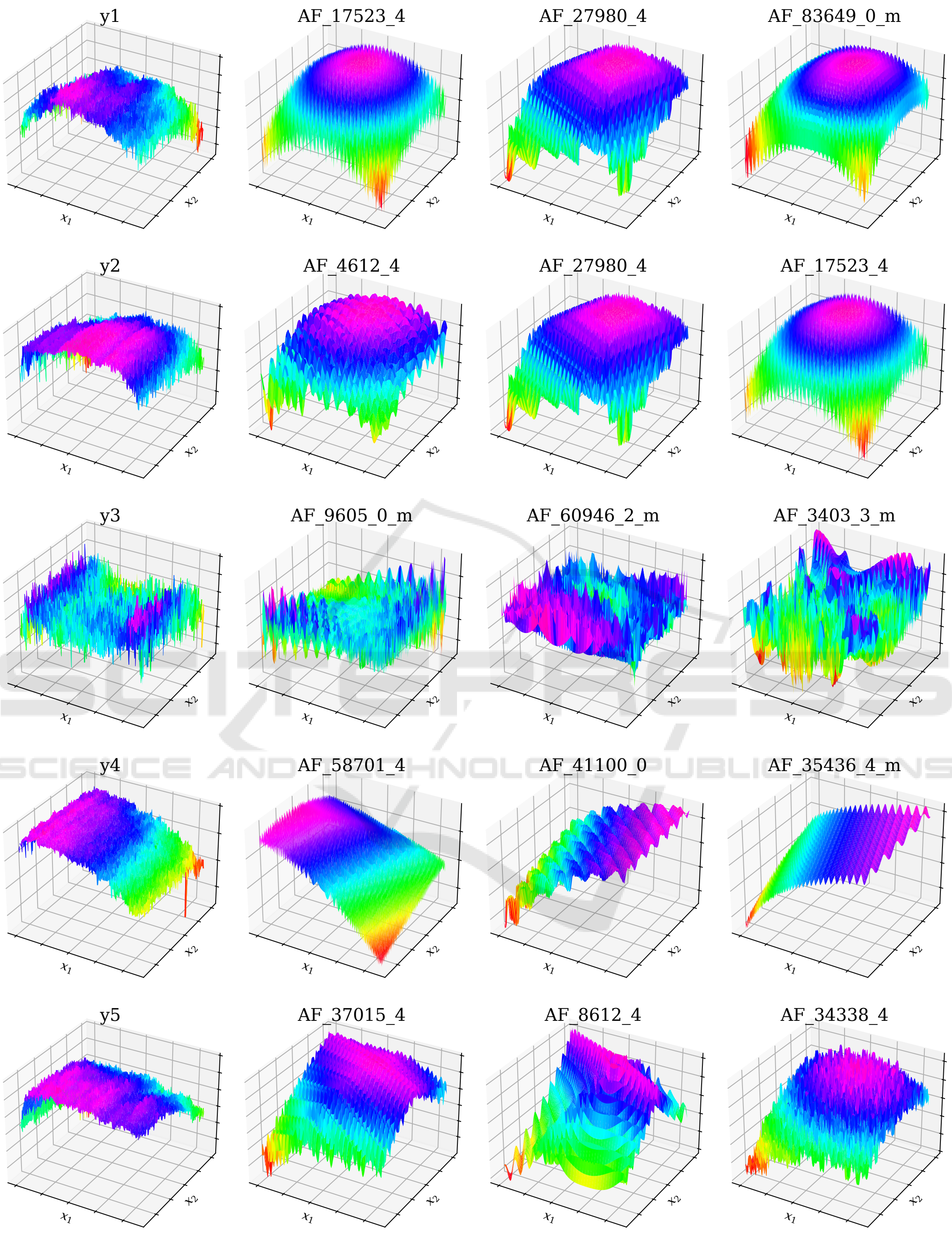


Figure 5: Landscape of the five 2d real-world problems *yi* and the three most similar artificial functions (AF) to each (Figure 4). The objective is minimization, the axes are shown inverted, thus pink indicates better solutions and red worse.

Table 3: Parameters and variants of CMA-ES with their value space for parameter tuning (de Nobel et al., 2021; van Rijn et al., 2016).

Hyperparameter	Description	Space
λ	Number of children derived from parents	{4,6,..,20}
μ_r	Ratio of parents selected from population	{0.2,0.3,..,0.8}
σ_0	Initial standard deviation	{0.1,0.2,..,0.9}
Bound correction	Correction if individual out of bounds	{saturate, unif, COTN, toroidal, mirror}
Active update	Covariance matrix update variation	{on, off}
Elitism	Strategy of the evolutionary algorithm	$\{(\mu,\lambda),(\mu+\lambda)\}$
Mirrored sampling	Mutations are the mirror image of another	{on, off}
Orthogonal	Orthogonal sampling	{on, off}
Threshold	Length threshold for mutation vectors	{on, off}
Weights	Weights for recombination	{default, equal, $\frac{1}{2}^{\lambda}$}
Restart	Local restart of CMA-ES	{off, IPOP, BIPOP}

used to calculate the utility of a CMA-ES configuration during parameter tuning and final validation.

To tune the parameters and select the best combination of variants of CMA-ES, we use SMAC3 (Version 2.0.0) (Lindauer et al., 2022) as the meta-algorithm. We use the default configuration of SMAC with an evaluation budget of 1 000 and perform three full-parameter tuning runs on each tuning reference and real-world problem.

5 TUNING RESULTS

We tune CMA-ES individually on each of the five real-world problems yi and on the three most similar randomly generated artificial functions (Figure 5). Then, the best derived configuration from each parameter tuning run is validated on the real-world problem.

Figure 7 shows the Empirical Cumulative Distribution Function for different CMA-ES configurations on the real-world problem $y1$. We compared the results with the default CMA-ES configuration and the CMA-ES with IPOP enabled. By simply enabling

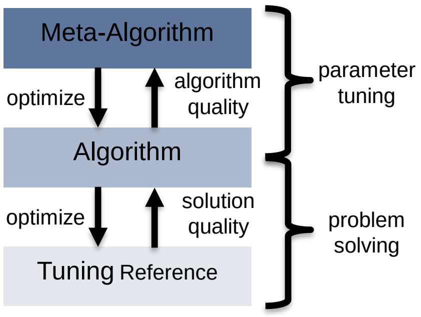


Figure 6: Solving the original problem and parameter tuning as two different optimization problems.

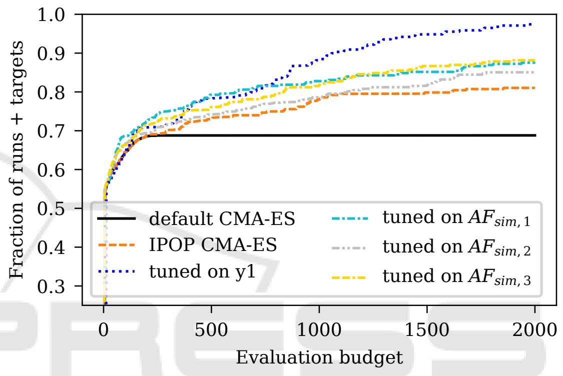
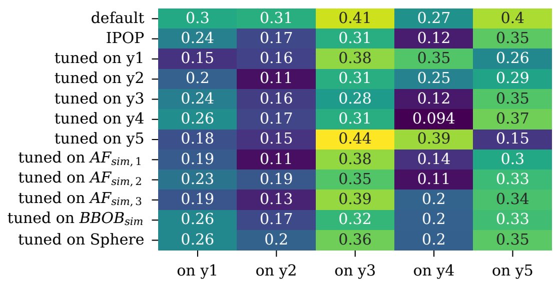


Figure 7: Empirical Cumulative Distribution Function for different CMA-ES configurations on the real-world problem $y1$ that solved the problem within the budget given by the x-axis.

IPOP, the performance increases with higher budgets. The default CMA-ES stagnates because it does not restart. Not surprisingly, the best performing configuration is the CMA-ES configuration tuned to $y1$, which almost always solves the optimization problem within a budget of 2 000 evaluations. It is important to note that configurations tuned to a similar artificial function improve the performance compared not only to the default configuration but also to the CMA-ES with IPOP enabled.

The parameter tuning and the validation vary slightly due to random effects within the optimization of SMAC and CMA-ES. Therefore, we perform three parameter tuning runs on each problem and then validate each of the derived configurations on the five real-world problems yi with three individual validation runs. Figure 8 summarizes the results as the average of the three validation runs. Not surprisingly, the IPOP CMA-ES always outperforms the pure default configuration of the CMA-ES without restart. The parameter configuration tuned to a real-world prob-



	on y1	on y2	on y3	on y4	on y5
default	0.3	0.31	0.41	0.27	0.4
IPOP	0.24	0.17	0.31	0.12	0.35
tuned on y1	0.15	0.16	0.38	0.35	0.26
tuned on y2	0.2	0.11	0.31	0.25	0.29
tuned on y3	0.24	0.16	0.28	0.12	0.35
tuned on y4	0.26	0.17	0.31	0.094	0.37
tuned on y5	0.18	0.15	0.44	0.39	0.15
tuned on $AF_{sim,1}$	0.19	0.11	0.38	0.14	0.3
tuned on $AF_{sim,2}$	0.23	0.19	0.35	0.11	0.33
tuned on $AF_{sim,3}$	0.19	0.13	0.39	0.2	0.34
tuned on $BBOB_{sim}$	0.26	0.17	0.32	0.2	0.33
tuned on Sphere	0.26	0.2	0.36	0.2	0.35

Figure 8: Utility ($1-AUC$) of the default and IPOP CMA-ES configuration and the CMA-ES configurations tuned to different functions (y-axis) when validating them on the five real-world problems yi (x-axis). The smaller the value, the better is the performance of the CMA-ES configuration.

lem always performs best when validated again on the same real-world problem.

In agreement with the similarity between two problems (Figure 4), transferring CMA-ES configurations tuned to one of the similar real-world problems $y1$ to $y2$ and transferring them to the other does indeed improve performance. Transferring the more dissimilar real-world problems $y4$ to $y5$ and vice versa does not improve performance compared to the IPOP CMA-ES.

CMA-ES configurations tuned on similar artificial functions (AF) can almost always beat or compete with the IPOP CMA-ES. Thus, tuning on similar functions improves the performance on the original optimization problem. Only for $y3$, IPOP CMA-ES is always better than the CMA-ES configurations tuned on the similar artificial functions.

The second most similar artificial function $AF_{sim,2}$ is the same function for $y1$ and $y2$ (Figure 5) and therefore we get the same three tuned configurations. The performance of these three tuned configurations on $AF_{sim,2}$ is comparatively worse than the configurations tuned to $AF_{sim,1}$ and $AF_{sim,3}$ when validated on $y1$ and $y2$. The reason for this is the second parameter tuning run, where the obtained best configuration of CMA-ES has no restart variant enabled. This leads to the average poor performance of $AF_{sim,2}$. Thus, restart is not necessary for every function, even if this function is multimodal and very similar in terms of ELA characteristics. In the future, to obtain a more robust tuned configuration of CMA-ES, the parameter tuning of CMA-ES should be performed on more than one similar artificial function.

The comparison with the most similar BBOB function shows that tuning on the more similar artificial functions leads to better results. Thus, generating artificial functions as new tuning references instead of selecting a similar BBOB function is recommended and worth the effort to improve the performance of CMA-ES.

Finally, tuning on the Sphere function also improves the performance of CMA-ES compared to the default configuration of CMA-ES and also competes with the IPOP CMAES. Tuning CMA-ES even on a dissimilar function can improve the performance. This confirms the results from (Thomaser et al., 2023b).

In summary, our results show that tuning improves the performance of CMA-ES. Moreover, the more similar two optimization problems are in terms of ELA features, the better the performance when the tuned configuration of CMA-ES is transferred to the other optimization problem.

6 CONCLUSION

In this paper, we compared CMA-ES configurations tuned to five different real-world 2D vehicle dynamics problems and compared the performance to CMA-ES configurations tuned to other functions, referred to as tuning references. We used Exploratory Landscape Analysis to quantify the similarity between two optimization problems and to select the most similar optimization problems from approximately 1 000 000 randomly generated artificial functions. The visual appearance of the most similar artificial functions is very similar to the original real-world problems.

Moreover, the CMA-ES configuration tuned to the similar artificial functions improved the performance on the real-world problems compared to the default CMA-ES configuration, IPOP CMA-ES, and also to CMA-ES configurations tuned to BBOB functions. This *encourages* further pursuit of the approach of transferring algorithm parameters tuned to similar problems.

Future research should investigate whether the additional computational effort for computing the ELA features and tuning the parameters is justified. Another topic of future research is to investigate whether the parameter tuning on several similar functions can increase the robustness of the CMA-ES configuration when transferred to the original optimization problem.

ACKNOWLEDGEMENTS

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

REFERENCES

Auger, A. and Hansen, N. (2005). A Restart CMA Evolution Strategy With Increasing Population Size. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776.

Bäck, T. (1994). Parallel Optimization of Evolutionary Algorithms. In Goos, G., Hartmanis, J., Leeuwen, J., Davidor, Y., Schwefel, H.-P., and Männer, R., editors, *Parallel Problem Solving from Nature — PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 418–427. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bäck, T., Foussette, C., and Krause, P. (2013). *Contemporary Evolution Strategies*. Natural Computing Series. Springer Berlin, Heidelberg, Berlin, Heidelberg, 1st ed. edition.

de Nobel, J., Vermetten, D., Wang, H., Doerr, C., and Bäck, T. (2021). Tuning as a Means of Assessing the Benefits of New Ideas in Interplay with Existing Algorithmic Modules. Technical report.

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

Fujii, G., Takahashi, M., and Akimoto, Y. (2018). CMA-ES-based structural topology optimization using a level set boundary expression-Application to optical and carpet cloaks. *Computer Methods in Applied Mechanics and Engineering*, 332:624–643.

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

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

Hansen, N. (2016). The CMA Evolution Strategy: A Tutorial. Technical report.

Hansen, N., Finck, S., Ros, R., and Auger, A. (2009). Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Technical Report RR-6829, INRIA.

Hansen, N. and Ostermeier, A. (1996). Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 312–317.

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

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer eBook Collection Mathematics and Statistics. Springer, New York, NY.

Kerschke, P., Preuss, M., Wessing, S., and Trautmann, H. (2015). Detecting Funnel Structures by Means of Exploratory Landscape Analysis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ACM Digital Library, pages 265–272, New York, NY. Association for Computing Machinery.

Kerschke, P., Preuss, M., Wessing, S., and Trautmann, H. (2016). Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models. In Neumann, F., editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM Digital Library, pages 229–236, New York, NY, USA. Association for Computing Machinery.

Kerschke, P. and Trautmann, H. (2019). Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package Flacco. In Bauer, N., Ickstadt, K., Lübke, K., Szepannek, G., Trautmann, H., and Vichi, M., editors, *Applications in Statistical Computing*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 93–123. Springer.

Koch-Dücker, H.-J. and Papert, U. (2014). Antilock braking system (ABS). In Reif, K., editor, *Brakes, Brake Control and Driver Assistance Systems*, Bosch professional automotive information, pages 74–93. Springer Vieweg, Wiesbaden.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. (2022). SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research*, 23(54):1–9.

Long, F. X., van Stein, B., Frenzel, M., Krause, P., Gitterle, M., and Bäck, T. (2022). Learning the Characteristics of Engineering Optimization Problems with Applications in Automotive Crash. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, New York, NY, USA. Association for Computing Machinery.

Loshchilov, I. and Hutter, F. (2016). CMA-ES for Hyperparameter Optimization of Deep Neural Networks.

Lunacek, M. and Whitley, D. (2006). The Dispersion Metric and the CMA Evolution Strategy. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, page 477. Association for Computing Machinery.

Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., and Rudolph, G. (2011). Exploratory Landscape Analysis. In Lanzi, P. L., editor, *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ACM Conferences, pages 829–836, New York, NY, USA. ACM.

Mersmann, O., Preuss, M., and Trautmann, H. (2010). Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In Schaefer, R., Cotta, C., Kołodziej, J., and Rudolph, G., editors, *Parallel Problem Solving from Nature, PPSN XI*, Lecture

Notes in Computer Science, pages 73–82. Springer, Berlin.

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

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

Pacejka, H. B. and Bakker, E. (1992). THE MAGIC FORMULA TYRE MODEL. *Vehicle System Dynamics*, 21(sup001):1–18.

Prager, R. P. (2023). pflacco: The R-package flacco in native Python code. https://github.com/Reiyan/pflacco.

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

Rice, J. R. (1976). The Algorithm Selection Problem. In Rubinoff, M. and Yovits, M. C., editors, *Advances in Computers*, volume 15, pages 65–118. Elsevier Science & Technology, Saint Louis.

Siemens Digital Industries Software (2020). Tire Simulation & Testing. https://www.plm.automation.siemens.com/global/en/products/simulation-test/tire-simulation-testing.html.

Škvorc, U., Eftimov, T., and Korošec, P. (2020). Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. *Applied Soft Computing*, 90.

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

The MathWorks, Inc. (2015). Simulink. https://www.mathworks.com/.

Thomaser, A., Vogt, M.-E., Bäck, T., and Kononova, A. V. (2023a). Real-world Optimization Benchmark from Vehicle Dynamics - Data and Code. https://doi.org/10.5281/zenodo.8307853.

Thomaser, A., Vogt, M.-E., Kononova, A. V., and Bäck, T. (2023b). Transfer of Multi-objectively Tuned CMA-ES Parameters to a Vehicle Dynamics Problem. In Emmerich, M., Deutz, A., Wang, H., Kononova, A. V., Naujoks, B., Li, K., Miettinen, K., and Yevseyeva, I., editors, *Evolutionary Multi-Criterion Optimization*, pages 546–560, Cham. Springer Nature Switzerland.

Tian, Y., Peng, S., Zhang, X., Rodemann, T., Tan, K. C., and Jin, Y. (2020). A Recommender System for Metaheuristic Algorithms for Continuous Optimization Based on Deep Recurrent Neural Networks. *IEEE Transactions on Artificial Intelligence*, 1(1):5–18.

van Rijn, S., Wang, H., van Leeuwen, M., and Bäck, T. (2016). Evolving the structure of Evolution Strategies. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8.

VIRES Simulationstechnologie GmbH (2020). Opencrg. https://www.asam.net/standards/detail/opencrg/.

Ye, F., Doerr, C., Wang, H., and Bäck, T. (2022). Automated Configuration of Genetic Algorithms by Tuning for Anytime Performance. *IEEE Transactions on Evolutionary Computation*, 26(6):1526–1538.