Transfer of multi-objectively tuned CMA-ES parameters to a vehicle dynamics problem

Thomaser, A.M.; Vogt, M.E.; Kononova, A.V.; Bäck, T.H.W.; Emmerich, M.T.; Deutz, A.; ... ; Yevseyeva, I.

Transfer of Multi-objectively Tuned CMA-ES Parameters to a Vehicle Dynamics Problem

André Thomaser[1,2(✉)], Marc-Eric Vogt[1], Anna V. Kononova[2], and Thomas Bäck[2]

[1] BMW AG, Munich, Germany
{andre.thomaser,marc-eric.vogt}@bmw.de
[2] LIACS, Leiden University, Leiden, The Netherlands
{a.m.thomaser,a.kononova,t.h.w.baeck}@liacs.leidenuniv.nl

Abstract. The conflict between *computational budget* and *quality of found solutions* is crucial when dealing with expensive black-box optimization problems from the industry. We show that through multi-objective parameter tuning of the Covariance Matrix Adaptation Evolution Strategy on benchmark functions different optimal algorithm configurations can be found for specific computational budgets and solution qualities. With the obtained Pareto front, tuned parameter sets are selected and transferred to a real-world optimization problem from vehicle dynamics, improving the solution quality and budget needed. The benchmark functions for tuning are selected based on their similarity to a real-world problem in terms of Exploratory Landscape Analysis features.

Keywords: Multi-objective · Parameter tuning · CMA-ES · Transfer learning · Vehicle dynamics · Exploratory landscape analysis

1 Introduction

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [16,18] is a class of iterative heuristic algorithms for solving generally non-linear, non-convex, single objective, continuous optimization problems by finding a solution within the feasible set $\mathcal{X} \subset \mathbb{R}^n$ that minimizes the objective function f [27] (Sect. 2). CMA-ES has been successfully applied to many real-world optimization problems including topology optimization [13] and hyperparameter optimization of neural networks [30].

Apart from the landscape of the objective function, the performance of CMA-ES on a specific optimization problem is determined by several parameters, and also, by the variant of CMA-ES [5]. In order not to concern the user with the task of selecting the appropriate parameters of an algorithm from a wide range of different settings and variants, automatic parameter tuning as an optimization problem itself was proposed [4,14]. Hence, two optimization problems can be distinguished: solving the original problem and parameter tuning [12]. Components of the former optimization problems are the original problem and the

M. Emmerich et al. (Eds.): EMO 2023, LNCS 13970, pp. 546–560, 2023.
https://doi.org/10.1007/978-3-031-27250-9_39

algorithm to find an optimal solution for this problem; while the latter consists of the algorithm and a meta-algorithm to find optimal parameters for the algorithm to solve the original problem. The quality of solutions for the original problem is called *fitness* and the quality of the parameters of the algorithm is called *utility* [12].

When measuring the utility two main metrics can be formulated: finding the best possible solution within a given budget (fixed-budget) and finding a solution as quickly as possible with a given target quality (fixed-target) [6]. In this paper we investigate the *conflict* of these two objectives when tuning the parameters of an algorithm: solution quality and used budget. We perform multi-objective parameter tuning to obtain a Pareto front, also called "performance front" [11], consisting of non-dominated parameter sets that satisfy both the quality and budget objective (Sect. 5). Since the real-world black-box optimization problem from vehicle dynamics design (Sect. 3) is computationally expensive to evaluate, we conduct and investigate the parameter tuning on similar functions from the black-box optimization benchmark (BBOB) [17].

Relating the performance of an algorithm on synthetic benchmark functions to real-world optimization problems for transfer learning is a difficult task [41]. One way is to assess the similarity between the real-world problem and the benchmark functions. Therefore, we use the same approach as in previous work [45] by performing *Exploratory Landscape Analysis* (ELA) [32] (Sect. 4).

2 Covariance Matrix Adaptation Evolution Strategy

In every generation g of the CMA-ES [16], a population x consisting of λ offspring 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 $\sigma^{(g)} \in \mathbb{R}_{>0}$:

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

Moreover, the mean value $m^{(g)}$, the covariance matrix $C^{(g)}$ and the standard deviation $\sigma^{(g)}$ are adapted in each generation as described below. With the given weights w_i, the new mean value $m^{(g+1)}$ is calculated as the weighted average of μ selected parents from the population:

$$m^{(g+1)} = m^{(g)} + c_m \sum_{i=1}^{\mu} w_i (x_{i:\lambda}^{(g+1)} - m^{(g)}). \tag{2}$$

The covariance matrix $C^{(g)}$ is updated with the evolution path $p_c^{(g)} \in \mathbb{R}^n$:

$$C^{(g+1)} = (1 - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i) C^{(g)} + c_1 \underbrace{p_c^{(g+1)} p_c^{(g+1)^T}}_{\text{rank-one update}} + c_\mu \underbrace{\sum_{i=1}^{\lambda} w_i \, y_{i:\lambda}^{(g+1)} (y_{i:\lambda}^{(g+1)})^T}_{\text{rank-}\mu\text{ update}}, \tag{3}$$

$$p_c^{(g+1)} = (1 - c_c)p_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{eff}}\frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}, \tag{4}$$

$$\mu_{eff} = (\sum_{i=1}^{\mu} w_i^2)^{-1}, \quad y_{i:\lambda}^{(g+1)} = \frac{x_{i:\lambda}^{(g+1)} - m^{(g)}}{\sigma^{(g)}}, \tag{5}$$

and the standard deviation $\sigma^{(g)}$ is updated with the conjugate evolution path $p_\sigma^{(g)} \in \mathbb{R}^n$:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|p_\sigma^{(g+1)}\right\|}{E\left\|\mathcal{N}(0, I)\right\|} - 1\right)\right), \tag{6}$$

$$p_\sigma^{(g+1)} = (1 - c_\sigma)p_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}C^{(g)^{-\frac{1}{2}}}\frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}. \tag{7}$$

The strategy parameters λ, μ, c_1, c_c, c_{mu}, c_σ define the optimization behavior of CMA-ES and can themselves be optimized for specific functions or groups of functions [3,51].

Several CMA-ES variants have been developed so far. In this paper the following variants are considered [36,46]:

1. **Active Update** [23]: Extends the adaptation of the covariance matrix with the most successful individuals by also considering the least successful individuals with a negative factor and therefore actively reducing the probability of searching in unpromising directions.
2. **Elitism** [46]: In the standard (μ, λ)-CMA-ES the μ children replace the λ parents. In the $(\mu + \lambda)$-CMA-ES the children and parents together form the next population.
3. **Mirrored Sampling** [8]: Only half of the search points of a new population are sampled from a multivariate normal distribution, the other half is the mirror image of the first one. Mirrored sampling increases the uniform distribution of the search points.
4. **Orthogonal Sampling** [48]: Orthonormalizes the vectors of the search points with the Gram-Schmidt process [7].
5. **Threshold Convergence:** [39]: Prevents the algorithm from getting stuck in a local optimum by requiring the mutation vectors to reach a length threshold. The threshold decreases after each generation.
6. **Step-Size Adaptation:** The standard step-size control in CMA-ES is Cumulative Step-Size Adaptation (CSA) [16]. Two-Point Step-Size Adaptation (TPA) [15] uses two search points from the population and Median Success Rule (MSR) [1] uses the median fitness of the offspring to an individual from the previous iteration to adjust the step-size.
7. **Weighted Recombination** [19]: Recombination is accomplished in CMA-ES by adjusting the mean values m with a weight vector w_i.

The combination of the different variants with respect to the optimization problem can improve the performance of CMA-ES [46,47].

3 Real-World Problem Description

The anti-lock braking system (ABS) [26] and the variable damper control (VDC) [35] can significantly improve driving safety by reducing the braking distance. The ABS adjusts the brake pressure so that the brake slip remains within the optimal range, thus preventing the wheels from locking and increasing the brake performance. The VDC improves brake performance by adjusting the damper constants of the shock absorbers, which influence the wheel load and, therefore, the braking force.

The *emergency straight-line full-stop braking maneuver with ABS fully engaged* [21] is a standard maneuver in the automotive industry for assessing the braking performance of a vehicle. The braking distance y is defined as the integral of the vehicle longitudinal velocity over time from velocity $v_s = 100\,\mathrm{km/h}$ at time t_s to $v_e = 0\,\mathrm{km/h}$ at time t_e:

$$y = \int_{t_s}^{t_e} v(t)\,dt. \tag{8}$$

Mechanical vehicle and its control systems, the driver and the environment can be simulated via a two-track model implemented in Simulink [44]. To accurately model the steady-state and transient behavior of the tires under slip conditions the MF-Tyre/MF-Swift tire model [42], which is based on Pacejka's Magic Formula [38] is used. On a standard workstation[1], one full simulation run takes about 15 to 20 min.

The objective is to find a parameter setting x within the lower bound B_{lb} and upper bound B_{ub} that minimizes the braking distance $y(x)$. In total 28 ABS and two VDC parameters are considered.

4 Exploratory Landscape Analysis for Transfer Learning

Exploratory Landscape Analysis. (ELA) [32] quantifies high-level properties of the landscape of an optimization problem [33]. The *flacco* package provides a wide collection of ELA features [25]. The total of 68 single features, structured in six sets, are appropriate for the considered real-world problem (Sect. 3): classical ELA (distribution, level, meta) [32], information content [34], dispersion [31], linear model, nearest better clustering [24,40] and principal component.

We use the instance-generating mechanism of the BBOB function suite [17] and consider five instances of each function. To calculate the features, a Sobol' design [37,43] with 16384 samples in $[-5, 5]^{30}$ is used. The resulting set of computed features is filtered [29] to exclude features with zero standard deviation across all problems and feature pairs with Spearman's rank correlation [28] above 0.99. This leaves 39 features. The feature values are then min-max-scaled to $[0, 1]$ for equal weighting.

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

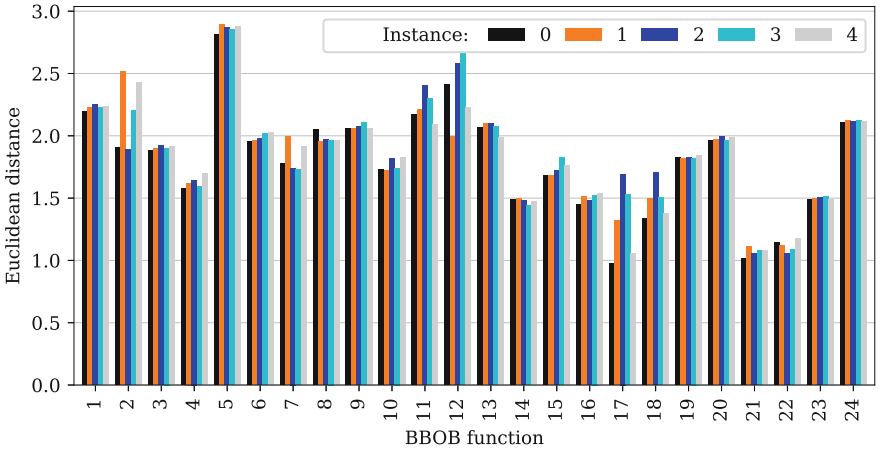

Fig. 1. Euclidean distances in the 39-dimensional ELA feature space between the considered real-world problem and 24 BBOB functions. The five instances {0, 1, 2, 3, 4} of each BBOB function are considered.

We define the similarity of two problems p_1 and p_2 as the Euclidean distance d between their feature vectors F_{p_1} and F_{p_2}:

$$d(p_1, p_2) = \|F_{p_1} - F_{p_2}\|_2 \,. \tag{9}$$

Using such distance, we investigate the similarity between the real-world problem and the five instances of each BBOB function based on the 39-dimensional feature vectors and conclude that such distances vary across functions and instances (Fig. 1). BBOB functions f_{17}, f_{21} are selected as *similar* to the considered real-world problem. Furthermore, to test the transferability of optimal parameters across functions, we augment the tuning reference with BBOB's sphere function f_1 as an example of a *dissimilar reference*.

5 Multi-objective Tuning of Algorithm's Parameters on Reference Functions

We define as a *meta-algorithm* an algorithm used to find the optimal set of parameters θ^* for an optimization algorithm A to solve the original real-world problem. Since the real-world problem is computationally expensive to evaluate, such meta-optimization can be performed on another (similar) problem or to increase generalisability on a set of problems – both such tasks exemplify transfer learning. In the following, we call such a function set a *tuning reference* $\mathit{\Pi}$. For each considered BBOB function f_j we use the five instances $i \in \{0, ..., 4\}$ as a tuning reference: $\mathit{\Pi}_{f_j} = \{f_{j_0}, ..., f_{j_4}\}$.

To compare the quality of a solution found by an algorithm across different problem instances and functions, we consider the distance $\Delta f = f - f^*$ to the known optimum f^* of a BBOB function f as *cost function* C:

$$C = \Delta f = f - f^*. \tag{10}$$

Considering the probabilistic nature of the algorithms, to obtain statistically meaningful results, we conduct $n_{opt} = 100$ optimization runs on each problem of the tuning reference Π. A *performance measure* over the obtained cost values $(c_1, ..., c_{n_{opt}})$ can then be calculated by a statistic h (e.g., mean, median). To focus more on average performance than peak performance and reduce the variance, we consider the median and the standard deviation across the n_{opt} optimization runs for each problem in the tuning reference:

$$h(c_1, ..., c_{n_{opt}}) = median(c_1, ..., c_{n_{opt}}) + std(c_1, ..., c_{n_{opt}}). \tag{11}$$

The quality of a parameter set θ for an algorithm A on the tuning reference Π, in the following referred to as *utility* $u(\theta, \Pi)$, is then assessed as the average performance measure h over the n_p problems in the tuning reference Π:

$$u(\theta, \Pi) = \frac{1}{n_p} \sum_{i=1}^{n_p} h(C(A(\theta), \Pi_i)). \tag{12}$$

In addition to the quality of the solution found across several optimization runs, the wall-clock time spent to find this solution is another crucial performance criterion of an optimization algorithm. On the real-world problem, the wall-clock time needed by the algorithm to generate the next population is negligible. The wall-clock time can be reduced mainly by running several simulations in parallel. Thus, the time for an optimization run correlates not directly with the evaluation budget n_{eval}, but with the number of serial iterations n_{iter}. In each iteration $n_{parallel}$ solution candidates can be evaluated simultaneously. The number of iterations depends on the population size λ, if $\lambda = n_{parallel}$, n_{iter} is equal to the number of generations $\frac{n_{eval}}{\lambda}$:

$$n_{iter} = \frac{n_{eval}}{\lambda} \left\lceil \frac{\lambda}{n_{parallel}} \right\rceil. \tag{13}$$

We assess a parameter set of CMA-ES based on two *conflicting objectives*: the utility u (Eq. 12) as a measurement of the quality of found solutions and the iteration budget n_{iter} (Eq. 13). Many algorithms exist for multi-objective hyperparameter optimization [20]. We employ the NSGA-II [10] implementation from the hyper-parameter optimization tool Optuna [2] as the *meta-algorithm*. To find a so-called performance front of Pareto optimal parameter sets for the CMA-ES algorithm, the meta-algorithm has a budget of 10,000 evaluations.

The set of algorithm's parameters being searched and assessed via multi-objective parameter tuning is specified in Table 1, based on the modular CMA-ES implementation [36,46]. Because of practical limitations in software licenses and computational resources required for the considered real-world problem, a maximum of 30 simulations can be executed in parallel. Therefore, only population sizes which are multiples of 30 are considered for CMA-ES. Infeasible

Table 1. Parameters and variants of CMA-ES with their value space for multi-objective parameter tuning with NSGA-II.

Hyperparameter	Description	Space
λ	Number of children derived from parents	{30, 60, 90}
μ_r	Ratio of parents selected from population	[0.2, 0.8]
σ_0	Initial standard deviation	]0, 1]
C_1	Learning rate rank-one update	]0, 1]
C_c	Learning rate covariance matrix adaption	]0, 1]
C_μ	Learning rate rank-μ update	]0, 1]
C_σ	Learning rate step size control	]0, 1[
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}
Adaptation σ	How to adapt the step size σ	{CSA, TPA, MSR}
Weights	Weights for recombination	{default, equal, $\frac{1}{2}^\lambda$}

solutions generated during the run of CMA-ES are corrected using the "saturate" method [9].

We set the iteration budget to 4200 function evaluations to practically be able to use the same budget once optimized parameters are transferred to the real-world problem (where such budget translates to about two days of simulations for one optimization run). The meta-algorithm can select the following iteration budgets: $\{10, 20, 30, 40, 60, 80, 100, 140\}$.

5.1 Results

A set of parameters θ is tuned based on the quality of found solutions $u(\theta, \Pi)$ on the tuning reference Π and the iteration budget n_{iter}. We obtained k Pareto optimal parameter sets $\theta^*_{\Pi_{f_j},i}$, $i \in \{1, ..., k\}$ from each tuning reference Π_{f_j}, $j \in \{1, 17, 21\}$. For further investigations, we assess these parameter sets on the tuning references $\Pi_{f_{17}}$ and $\Pi_{f_{21}}$ (Fig. 2).

The resulting tuned parameter sets are shown in Table 2. It is important to mention that across all tuned parameter sets CSA is used, mirrored and orthogonal sampling are on and "elitist" is off.

Optimal parameter sets for 140 and 10 iterations on $\Pi_{f_{17}}$ and $\Pi_{f_{21}}$ are additionally assessed for all considered intermediate iteration budgets (marked with circles in Fig. 2). If the optimization run is stopped earlier or continued beyond the optimal iteration budget, another tuned parameter set would have performed better on average. It turns out that performance of the tuned parameter sets is very *sensitive* to the iteration budget – runs of the algorithm with parameters tuned for one budget result in significantly worse performance on other budgets.

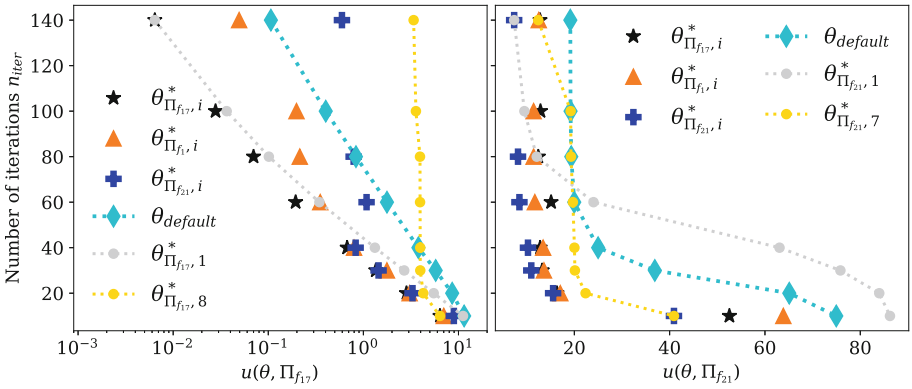

Fig. 2. Values of the quality of found solutions $u(\theta, \Pi)$ (Eq. 12) and the iteration budgets n_{iter} on the tuning references $\Pi_{f_{17}}$ (left) and $\Pi_{f_{21}}$ (right) for the obtained k Pareto optimal parameter sets $\theta^*_{\Pi_{f_j},i}$, $i \in \{1, ..., k\}$ from each tuning reference Π_{f_j}, $j \in \{1, 17, 21\}$ (Table 2). Each tuning reference consists of five instances of the corresponding BBOB function. Setting $\theta_{default}$ refers to a default parameter set (not tuned) recommended in the modular CMA-ES implementation with an adjusted population size of 30. The optimal parameter sets for 140 and 10 iterations on $\Pi_{f_{17}}$, $\Pi_{f_{21}}$ and $\theta_{default}$ are assessed for all considered iteration budgets. Symbols are connected with dotted lines when the same CMA-ES parameter set is used.

For example, $\theta^*_{\Pi_{f_{17}},8}$ (marked with yellow circles in Fig. 2, left) is tuned for 10 iterations and performs worse than the default parameters for iteration budgets grater than 30 (shown as rhombi). The other way around, $\theta^*_{\Pi_{f_{21}},1}$ (marked with silver circles in Fig. 2, right) is tuned for 140 iterations and is worse than the default parameters (shown as rhombi) for iteration budgets less than 80. Thus, a tuned parameter set performs better only for a *specific* iteration budget.

The obtained Pareto front from the multi-objective parameter tuning in Fig. 2 shows the conflict between the budget spent and the solution quality found on a set of optimization problems. With increasing the number of iterations the quality of the found solutions generally increases, exceptions can occur because of the variance across the n_{opt} optimization runs. However, the improvement decreases as the number of iterations increases. In particular, on $\Pi_{f_{21}}$, the quality of the solutions found does not increase significantly beyond 60 iterations, both for the tuned parameter sets and for the default parameter set. Thus, the obtained Pareto front indicates which number of iterations is suitable and at what point even a tuned parameter set for a larger iteration budget is *not recommended*.

Despite the dissimilarity of the landscapes of f_1, f_{17} and f_{21}, *a transfer of CMA-ES parameters* tuned on reference Π_{f_1} improves the performance of CMA-ES compared to the default parameter set on $\Pi_{f_{17}}$ and $\Pi_{f_{21}}$ (marked with triangles and diamonds in Fig. 2). Also, transferring the parameter sets tuned on $\Pi_{f_{17}}$ improves the performance of CMA-ES on $\Pi_{f_{21}}$ and vice versa, except for $\theta^*_{\Pi_{f_{21}},1}$ on $\Pi_{f_{17}}$ (marked with crosses and stars in Fig. 2).

Table 2. Parameter values for CMA-ES of the default parameter set $\theta_{default}$ and Pareto optimal parameter sets $\theta^*_{\Pi_{f_j},i}$ on the tuning references Π_{f_j}, $j \in \{1, 17, 21\}$. Each tuning reference consists of five instances of the BBOB function f_j. The search space of the parameters is given in Table 1. CSA, mirrored, orthogonal sampling is used and elitist is not across all tuned parameter sets. In the default settings these variants are all off. For parameter values between zero and one, higher values are shaded darker.

CMA-ES	n_{iter}	λ	μ_r	σ_0	c_1	c_c	c_μ	c_σ	active	threshold	weights
$\theta_{default}$	-	30	0.5000	0.2000	0.0020	0.1208	0.0049	0.1600	off	off	default
$\theta^*_{\Pi_{f_{17}},1}$	140	30	0.4506	0.3151	0.0038	0.0060	0.0241	0.3038	off	on	default
$\theta^*_{\Pi_{f_{17}},2}$	100	30	0.5282	0.3277	0.0038	0.0060	0.0241	0.4072	off	on	default
$\theta^*_{\Pi_{f_{17}},3}$	80	30	0.4506	0.3249	0.0038	0.0060	0.0241	0.4502	off	on	default
$\theta^*_{\Pi_{f_{17}},4}$	60	30	0.4506	0.2819	0.0038	0.0060	0.0241	0.6545	off	on	default
$\theta^*_{\Pi_{f_{17}},5}$	40	30	0.4506	0.4409	0.0038	0.2904	0.0246	0.9613	on	off	default
$\theta^*_{\Pi_{f_{17}},6}$	30	30	0.4506	0.4409	0.0038	0.2904	0.0246	0.9613	on	off	default
$\theta^*_{\Pi_{f_{17}},7}$	20	30	0.7140	0.5780	0.0852	0.0060	0.0246	0.9712	on	off	default
$\theta^*_{\Pi_{f_{17}},8}$	10	30	0.7140	0.7145	0.2886	0.0060	0.0246	0.9327	on	off	default
$\theta^*_{\Pi_{f_{21}},1}$	140	60	0.7208	0.8618	0.0026	0.7898	0.1301	0.1266	on	on	default
$\theta^*_{\Pi_{f_{21}},2}$	80	60	0.7208	0.8618	0.0026	0.7133	0.1254	0.5105	off	on	default
$\theta^*_{\Pi_{f_{21}},3}$	60	60	0.7208	0.8618	0.0026	0.7898	0.1301	0.8456	on	on	default
$\theta^*_{\Pi_{f_{21}},4}$	40	30	0.7208	0.7145	0.0852	0.0135	0.0173	0.9817	off	on	default
$\theta^*_{\Pi_{f_{21}},5}$	30	30	0.7751	0.7145	0.0852	0.0135	0.0173	0.9817	off	on	default
$\theta^*_{\Pi_{f_{21}},6}$	20	30	0.7208	0.7145	0.0852	0.0135	0.0173	0.9448	off	on	default
$\theta^*_{\Pi_{f_{21}},7}$	10	30	0.7208	0.2824	0.2761	0.0060	0.0173	0.9145	off	on	$\frac{1}{2}^{\lambda}$
$\theta^*_{\Pi_{f_1},1}$	140	30	0.4651	0.6867	0.0096	0.5414	0.0056	0.7292	off	off	default
$\theta^*_{\Pi_{f_1},2}$	100	30	0.4651	0.4948	0.0096	0.0021	0.0056	0.9712	off	off	default
$\theta^*_{\Pi_{f_1},3}$	80	30	0.4651	0.4948	0.0096	0.0021	0.0056	0.9712	off	off	default
$\theta^*_{\Pi_{f_1},4}$	60	30	0.4651	0.4948	0.0330	0.0021	0.0056	0.9712	off	off	default
$\theta^*_{\Pi_{f_1},5}$	40	30	0.4651	0.4948	0.0429	0.0021	0.0056	0.9712	on	off	default
$\theta^*_{\Pi_{f_1},6}$	30	30	0.4651	0.5557	0.1044	0.0021	0.0056	0.9712	on	off	default
$\theta^*_{\Pi_{f_1},7}$	20	30	0.4651	0.5557	0.1044	0.0021	0.0056	0.9712	on	off	default
$\theta^*_{\Pi_{f_1},8}$	10	30	0.4651	0.5852	0.3947	0.0021	0.0056	0.9712	off	on	default

Looking at individual values in the tuned parameter sets, the used variants mirrored and orthogonal sampling across all tuned parameter sets indicate a general increase in utility of CMA-ES for different problems. The population size was increased from the default value of 14 to at least 30 because of the number of parallel evaluations. In accordance with [48,49], mirrored and orthogonal sampling especially improves the exploration effects of a large population. The initial step size σ_0 is higher than the default step size for all tuned parameter

sets using the larger population size for an initial higher exploration. The tuned parameters are also adjusted to features of the problem and the algorithm that are identical or similar on the other problems like dimension and population size. *Transfer learning* thus can improve the performance of the algorithm.

A major difference between $\Pi_{f_{17}}$ and $\Pi_{f_{21}}$ is the comparatively worse utility of the solutions found regardless of the parameter set. On $\Pi_{f_{21}}$ the algorithm often gets stuck in a local optimum and does not find a solution near the global optimum. Therefore, exploration is especially important on $\Pi_{f_{21}}$, resulting in significant differences in parameter values of $\theta^*_{\Pi_{f_{21}},i}$ to $\theta^*_{\Pi_{f_1},i}$ and $\theta^*_{\Pi_{f_{17}},i}$. An open question is whether, instead of one single long run, two or three shorter runs result in better peak performance.

It is of interest to note that the learning rates mostly decrease or increase constantly across the iterations for the tuned parameter sets, especially c_σ. Also active covariance matrix update variation is mostly on for lower number of iterations and length threshold for mutation vectors is set to on for higher numbers of iterations. This highlights the need to conduct more research in the direction of landscape-aware adaptive parameter tuning [22].

5.2 Transfer of Tuned Parameters to the Real-World Problem

We transfer parameter sets obtained from the tuning references to the real-world problem from vehicle dynamics design and analyze whether they also improve the search performance of CMA-ES compared to the default parameter set. One optimization run with 140 iterations on the real-world problem takes about two days. Therefore, only two optimization runs for the same parameter set are conducted and only 40 and 140 iterations are considered as budget for one optimization run. Thus, we transfer the optimal parameter sets on the considered tuning references Π_{f_1}, $\Pi_{f_{17}}$, $\Pi_{f_{21}}$ for 140 iterations $\theta^*_{\Pi_{f_1},1}$, $\theta^*_{\Pi_{f_{17}},1}$, $\theta^*_{\Pi_{f_{21}},1}$ and for 40 iterations $\theta^*_{\Pi_{f_1},5}$, $\theta^*_{\Pi_{f_{17}},5}$, $\theta^*_{\Pi_{f_{21}},4}$ to the real-world problem (Fig. 3). The initialization of the mean value $m^{(0)}$ is set to the default parametrization of the control parameters for all optimizations runs.

Overall, small differences between the curves can be observed. For an iteration budget of 140 (Fig. 3, left), the solutions found with $\theta^*_{\Pi_{f_{21}},1}$ tend to be worse compared to the other parameter sets, especially at the beginning of the optimization run. This is the price for the higher exploration of the search-space. Unexpectedly, by far the shortest braking distance is found by the first run of CMA-ES with the parameter set $\theta^*_{\Pi_{f_1},1}$ (tuned on a dissimilar function f_1) within only 41 iterations and the second run can compete with the others as well. This confirms the observations on the tuning references (Sect. 5.1), where a transfer of $\theta^*_{\Pi_{f_1},i}$ also improved the performance of CMA-ES compared to the default parameter set. Tuning the parameters of CMA-ES to general problem characteristics and algorithm settings like problem dimension and population size improves the search behavior.

With an iteration budget of only 40, the variance across the found braking distances increases compared to 140 iterations. Overall, the best parameter sets

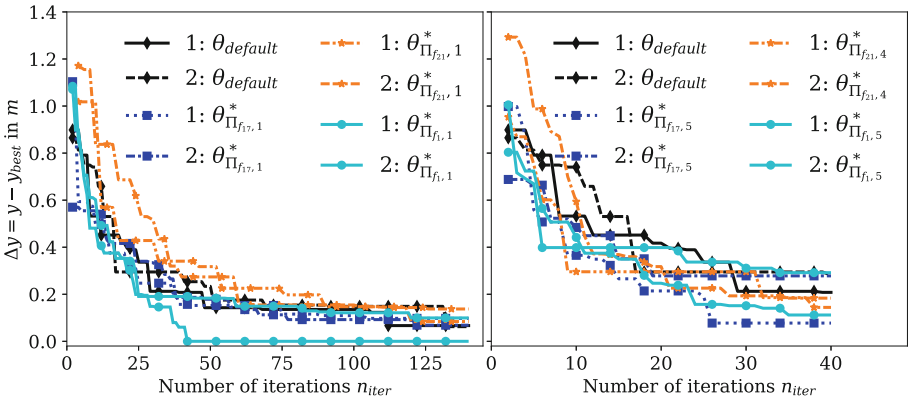

Fig. 3. Distance to the best known braking distance attainable on the considered real-world problem within intermediate computational budgets by CMA-ES configurations tuned for iteration budgets of 140 (left) and 40 (right). Line colours define the tuning reference Π_{f_j} (where index $j \in \{1, 17, 21\}$ defines the BBOB function, five instances were considered) on which CMA-ES (Table 2) has been tuned. Two independent runs are shown for each CMA-ES parameter set. Setting $\theta_{default}$ refers to a default parameter set recommended in the modular CMA-ES implementation with an adjusted population size of 30 (not tuned). (Color figure online)

for 40 iterations tend to find shorter braking distances faster than the default parameter set $\theta_{default}$.

In summary, similar phenomena in the performance of CMA-ES on benchmark functions can also be observed on the real-world problem, encouraging further investigation of transfer learning.

6 Conclusion

In this paper, we tuned different parameters and variants of CMA-ES with the two objectives of computational budget needed and quality of the solution found on functions taken from the BBOB benchmark test set. A tuned parameter set is only optimal for a given budget and problem or set of problems, so a Pareto front consisting of different parameter sets for each set of problems was derived. In order not to tune the parameters to a specific budget, the area under the empirical cumulative distribution function curve could be an alternative objective [50].

The use of certain variants of CMA-ES results in an improvement across all problems considered. One reason for this is the adaptation to general problem characteristics and algorithm settings. For example, orthogonal and mirrored sampling generally improve search performance for relatively large populations, while a higher initial step size and "threshold" improve exploration of the large search space. A simple solution besides tuning variants of CMA-ES would be to provide simple heuristics and recommendations (rules of thumb).

The values of the parameter sets tuned on different sets of problems differ significantly, but lead to similar results on the real-world problem from vehicle dynamics. A new best solution on the real-world problem was found by a tuned parameter set on the sphere function f_1. The improvement of this solution is 1.8 times better than the improvement of the best solution in a Sobol' design with 16384 samples compared to the default parameterization of the real-world problem.

The similarity of the considered benchmark functions to the real-world problem was quantified by the Euclidean distance of Exploratory Landscape Analysis feature values. The assumption of correlation between similarity of two problems quantified by ELA features and the difficulty for an algorithm configuration of solving them needs further investigations.

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.

The authors would like to thank Jacob de Nobel and Diederick Vermetten for their support with the modular CMA-ES implementation.

References

1. Ait Elhara, O., Auger, A., Hansen, N.: A median success rule for non-elitist evolution strategies: study of feasibility. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO 2013, Association for Computing Machinery, New York, pp. 415–422 (2013). https://doi.org/10.1145/2463372.2463429
2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Association for Computing Machinery, New York, pp. 2623–2631 (2019). https://doi.org/10.1145/3292500.3330701
3. Andersson, M., Bandaru, S., Ng, A.H., Syberfeldt, A.: Parameter tuned CMA-ES on the CEC'15 expensive problems. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 1950–1957 (2015). https://doi.org/10.1109/CEC.2015.7257124
4. Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press Inc, Oxford (1996). https://doi.org/10.1093/oso/9780195099713.001.0001
5. Bäck, T., Foussette, C., Krause, P.: Contemporary Evolution Strategies. Natural Computing Series, 1st edn. Springer, Berlin (2013)
6. Bartz-Beielstein, T., et al.: Benchmarking in Optimization: Best Practice and Open Issues. Technical report (2020). http://arxiv.org/2007.03488arxiv.org/pdf/2007.03488
7. Björck, Å.: Numerics of gram-Schmidt orthogonalization. Linear Algebra Appl. **197**, 297–316 (1994). https://doi.org/10.1016/0024-3795(94)90493-6

8. Brockhoff, D., Auger, A., Hansen, N., Arnold, D.V., Hohm, T.: Mirrored sampling and sequential selection for evolution strategies. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 11–21. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_2
9. Caraffini, F., Kononova, A.V., Corne, D.: Infeasibility and structural bias in differential evolution. Inf. Sci. **496**, 161–179 (2019)
10. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
11. Dréo, J.: Using Performance Fronts for Parameter Setting of Stochastic Metaheuristics. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, pp. 2197–2200. ACM Conferences, Association for Computing Machinery, New York (2009). https://doi.org/10.1145/1570256.1570301
12. Eiben, A.E., Smit, S.K.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol. Comput. **1**(1), 19–31 (2011). https://doi.org/10.1016/j.swevo.2011.02.001
13. Fujii, G., Takahashi, M., Akimoto, Y.: CMA-ES-based structural topology optimization using a level set boundary expression-application to optical and carpet cloaks. Comput. Methods Appl. Mech. Eng. **332**, 624–643 (2018). https://doi.org/10.1016/j.cma.2018.01.008
14. Grefenstette, J.: Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man Cybern. **16**(1), 122–128 (1986). https://doi.org/10.1109/TSMC.1986.289288
15. Hansen, N.: CMA-ES with Two-Point Step-Size Adaptation. Technical report RR-6527, INRIA (2008). https://www.hal.inserm.fr/INRIA/inria-00276854
16. Hansen, N.: The CMA Evolution Strategy: A Tutorial. Technical report (2016). https://arxiv.org/pdf/1604.00772
17. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Technical report RR-6829, INRIA (2009). https://hal.inria.fr/inria-00362633/
18. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 312–317 (1996). https://doi.org/10.1109/ICEC.1996.542381
19. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001). https://doi.org/10.1162/106365601750190398
20. Hernández, A.M., van Nieuwenhuyse, I., Rojas-Gonzalez, S.: A survey on multiobjective hyperparameter optimization algorithms for Machine Learning. ArXiv (2021). https://arxiv.org/pdf/2111.13755.pdf
21. International Organization for Standardization: ISO 21994:2007 - Passenger cars - Stopping distance at straight-line braking with ABS - Open-loop test method (2007)
22. Jankovic, A., Eftimov, T., Doerr, C.: Towards feature-based performance regression using trajectory data. In: Castillo, P.A., Jiménez Laredo, J.L. (eds.) EvoApplications 2021. LNCS, vol. 12694, pp. 601–617. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72699-7_38
23. Jastrebski, G.A., Arnold, D.V.: Improving evolution strategies through active covariance matrix adaptation. In: IEEE International Conference on Evolutionary Computation, pp. 2814–2821 (2006). https://doi.org/10.1109/CEC.2006.1688662

24. Kerschke, P., Preuss, M., Wessing, S., Trautmann, H.: Detecting funnel structures by means of exploratory landscape analysis. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 265–272. ACM Digital Library, Association for Computing Machinery, New York (2015). https://doi.org/10.1145/2739480.2754642
25. Kerschke, P., Trautmann, H.: 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., Vichi, M. (eds.) Applications in Statistical Computing. SCDAKO, pp. 93–123. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25147-5_7
26. Koch-Dücker, H.-J., Papert, U.: Antilock braking system (ABS). In: Reif, K. (ed.) Brakes, Brake Control and Driver Assistance Systems. BPAI, pp. 74–93. Springer, Wiesbaden (2014). https://doi.org/10.1007/978-3-658-03978-3_6
27. Kochenderfer, M.J., Wheeler, T.A.: Algorithms for Optimization. The MIT Press, Cambridge and London (2019)
28. Kokoska, S., Zwillinger, D.: CRC Standard Probability and Statistics Tables and Formulae, CRC Press, Boca Raton (2000). https://doi.org/10.1201/b16923
29. Long, F.X., van Stein, B., Frenzel, M., Krause, P., Gitterle, M., Bäck, T.: Learning the characteristics of engineering optimization problems with applications in automotive crash. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO 2022, Association for Computing Machinery, New York, (2022). https://doi.org/10.1145/3512290.3528712
30. Loshchilov, I., Hutter, F.: CMA-ES for Hyperparameter Optimization of Deep Neural Networks (2016). https://arxiv.org/abs/1604.07269
31. Lunacek, M., Whitley, D.: The dispersion metric and the CMA evolution strategy. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, p. 477. Association for Computing Machinery (2006). https://doi.org/10.1145/1143997.1144085
32. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Lanzi, P.L. (ed.) Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, ACM Conferences, ACM, New York, pp. 829–836 (2011). https://doi.org/10.1145/2001576.2001690
33. Mersmann, O., Preuss, M., Trautmann, H.: Benchmarking evolutionary algorithms: towards exploratory landscape analysis. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 73–82. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_8
34. Muñoz, M.A., Kirley, M., Halgamuge, S.K.: Exploratory landscape analysis of continuous space optimization problems using information content. IEEE Trans. Evol. Comput. **19**(1), 74–87 (2015). https://doi.org/10.1109/TEVC.2014.2302006
35. Niemz, T.: Reducing Braking Distance by Control of Semi-Active Suspension. Dissertation, Technische Universität Darmstadt (2007). https://tuprints.ulb.tu-darmstadt.de/912/
36. de Nobel, J., Vermetten, D., Wang, H., Doerr, C., Bäck, T.: Tuning as a Means of Assessing the Benefits of New Ideas in Interplay with Existing Algorithmic Modules. Technical report (2021). https://arxiv.org/pdf/2102.12905
37. Owen, A.B.: Scrambling sobol' and niederreiter-xing points. J. Complex. **14**(4), 466–489 (1998). https://doi.org/10.1006/jcom.1998.0487
38. Pacejka, H.B., Bakker, E.: The magic formula tyre model. Veh. Syst. Dyn. **21**(sup001), 1–18 (1992). https://doi.org/10.1080/00423119208969994

39. Piad-Morffis, A., Estévez-Velarde, S., Bolufé-Röhler, A., Montgomery, J., Chen, S.: Evolution strategies with thresheld convergence. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 2097–2104 (2015). https://doi.org/10.1109/CEC.2015.7257143
40. Preuss, M.: Improved topological niching for real-valued global optimization. In: Chio, C., et al. (eds.) EvoApplications 2012. LNCS, vol. 7248, pp. 386–395. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29178-4_39
41. Sala, R., Müller, R.: Benchmarking for metaheuristic black-box optimization: perspectives and open challenges. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2020). https://doi.org/10.1109/CEC48606.2020.9185724
42. Siemens Digital Industries Software: Tire Simulation & Testing (2020). https://www.plm.automation.siemens.com/global/en/products/simulation-test/tire-simulation-testing.html
43. Sobol', I.M.: On the distribution of points in a cube and the approximate evaluation of integrals. Comput. Math. Math. Phys. **7**(4), 86–112 (1967). https://doi.org/10.1016/0041-5553(67)90144-9
44. The MathWorks Inc: Simulink (2015). https://www.mathworks.com/'
45. Thomaser, A., Kononova, A.V., Vogt, M.E., Bäck, T.: One-shot optimization for vehicle dynamics control systems: towards benchmarking and exploratory landscape analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 2036–2045. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3520304.3533979
46. van Rijn, S., Wang, H., van Leeuwen, M., Bäck, T.: Evolving the structure of evolution strategies. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8 (2016). https://doi.org/10.1109/SSCI.2016.7850138
47. van Rijn, S., Wang, H., van Stein, B., Bäck, T.: Algorithm configuration data mining for cma evolution strategies. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Association for Computing Machinery, New York, pp. 737–744 (2017). https://doi.org/10.1145/3071178.3071205
48. Wang, H., Emmerich, M., Bäck, T.: Mirrored orthogonal sampling with pairwise selection in evolution strategies. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC 2014, Association for Computing Machinery, New York, pp. 154–156 (2014). https://doi.org/10.1145/2554850.2555089
49. Wang, H., Emmerich, M., Bäck, T.: Mirrored orthogonal sampling for covariance matrix adaptation evolution strategies. Evol. Comput. **27**(4), 699–725 (2019). https://doi.org/10.1162/evco_a_00251
50. Ye, F., Doerr, C., Wang, H., Bäck, T.: Automated configuration of genetic algorithms by tuning for anytime performance. IEEE Trans. Evol. Comput. **26**(6), 1526–1538 (2022). https://doi.org/10.1109/TEVC.2022.3159087
51. Zhao, M., Li, J.: Tuning the hyper-parameters of CMA-ES with tree-structured Parzen estimators. In: 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI), pp. 613–618 (2018). https://doi.org/10.1109/ICACI.2018.8377530