One-Shot Optimization for Vehicle Dynamics Control Systems: Towards Benchmarking and Exploratory Landscape Analysis

André Thomaser
BMW Group
München, Germany
andre.thomaser@bmw.de

Anna V. Kononova
LIACS, Leiden University
Leiden, The Netherlands
a.kononova@liacs.leidenuniv.nl

Marc-Eric Vogt
BMW Group
München, Germany
marc-eric.vogt@bmw.de

Thomas Bäck
LIACS, Leiden University
Leiden, The Netherlands
t.h.w.baeck@liacs.leidenuniv.nl

ABSTRACT

Many real-world black-box optimization problems from industry are computationally expensive. Due to the advantage in wall-clock time, fully parallel sampling (one-shot search) is therefore often chosen over iterative search and adaptive sampling approaches. Our contribution shows how using a surrogate model (one-shot optimization with surrogate) can enhance the best solution found within the initial sample, requiring no further problem evaluations.

We test several surrogate types for one-shot optimization on a real-world problem from the field of vehicle dynamics control systems and the 24 well-known BBOB benchmark test functions.

For the real-world problem and most of the benchmark functions considered, a multi-layer perceptron (neural network) as surrogate model for one-shot optimization leads to worse solutions than the one-shot search, in contrast to random forest and support vector machine. Moreover, our results show that mean squared error as a commonly used quality metrics for regression models is not feasible for selecting a surrogate model for one-shot optimization.

To characterize the considered problems and to assess the similarity between the real-world problem and the benchmark functions, exploratory landscape analysis was performed. We provide some guidance on how to utilize this information to select a surrogate type for specific problems.

CCS CONCEPTS

• **Applied computing** → **Engineering**; • **Mathematics of computing** → *Continuous optimization*; *Exploratory data analysis*; • **Computing methodologies** → *Supervised learning by regression.*

KEYWORDS

one-shot optimization, vehicle dynamics, exploratory landscape analysis, benchmarking, surrogate-assisted optimization

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3533979

ACM Reference Format:
André Thomaser, Anna V. Kononova, Marc-Eric Vogt, and Thomas Bäck. 2022. One-Shot Optimization for Vehicle Dynamics Control Systems: Towards Benchmarking and Exploratory Landscape Analysis. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3520304.3533979

1 INTRODUCTION

Competition in the automotive industry is intensifying from year to year. This results in the need to bring technologically advanced vehicles to the market within ever shorter development intervals. To achieve this, large parts of the development process take place virtually. This means that instead of building physical prototypes, the vehicles are modeled virtually. One development task consists in tuning the parameters of control systems in vehicle dynamics design. For this, virtual models are developed that allow the search for optimal parameter sets for these control systems.

Vehicle dynamics is affected by many control systems, e.g., yaw stability control, engine control, active suspensions, rollover prevention, etc. [44]. Each of these control systems requires the specification of many highly interdependent parameters. Hence, the field of control systems in vehicle dynamics design offers a variety of highly complex optimization problems.

In this paper we will look at two vehicle dynamic control systems that can significantly improve driving safety by reducing the braking distance while simultaneously maintaining the vehicle's lateral stability: the anti-lock braking system (ABS) [26] and the variable damper control (VDC) [38].

1.1 Problem Description

The ABS controls the relative motion between a tire and the road surface during braking, so-called brake slip, by adjusting the brake pressure so that the brake slip remains within the optimal range, thus preventing the wheels from locking and keeping the vehicle stable during braking. The system achieves this by predicting the degree of slip between the wheels and the road surface. Meanwhile the VDC regulates damper constants of the shock absorbers, which influence the wheel load and, therefore, the braking force.

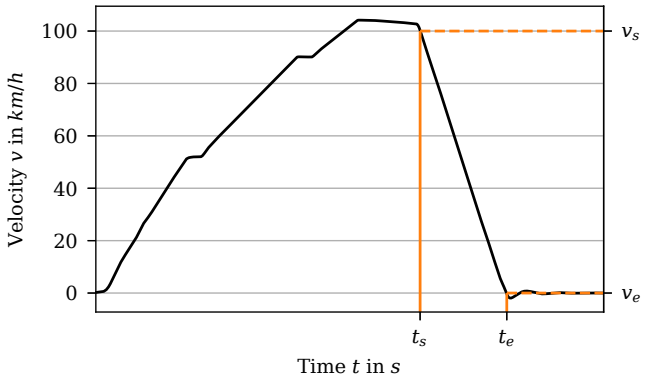

Figure 1: Start time t_s and end time t_e for calculating the braking distance while an emergency straight-line full-stop braking maneuver from v_s = 100 km/h to v_e = 0 km/h.

One maneuver used as a standard in the industry for assessing braking performance of a vehicle is the *emergency straight-line full-stop braking maneuver with ABS fully engaged* [17], which consists of the following phases (Figure 1):

- At the beginning of the maneuver the vehicle is accelerated until it reaches a maximum velocity of 103.5 km/h.
- Then a waiting phase follows in which neither acceleration nor deceleration is applied.
- The braking begins at a velocity of 103 km/h.
- When the vehicle stands still, the maneuver is completed.

The braking distance y is then defined as the integral of the vehicle longitudinal velocity over time from velocity v_s = 100 km/h at time t_s to v_e = 0 km/h at time t_e (Figure 1):

$$y = \int_{t_s}^{t_e} v(t)\, dt. \tag{1}$$

To avoid possible disturbances which might affect the beginning of the braking process, the starting velocity of the breaking maneuver v_s is considered lower than the velocity at the beginning of the braking process.

There are hundreds of ABS and VDC control parameters in a vehicle covering different driving situations such as straight-line braking and braking while cornering at different velocities, and different environmental situations such as low or high friction values between the tire and the road surface. Not all of them affect the braking distance during straight-line braking.

Based on sensitivity analysis with the Morris method [35][1], we have selected 28 ABS and 2 VDC parameters that have been shown to have the largest influence on the braking distance during a straight-line braking maneuver from 100 km/h to 0 km/h. The chosen number of parameters is large enough to describe the problem realistically for the industrial setting. For technical reasons, there is a certain resolution for each parameter, which makes them discrete. The number of levels varies across the 30 parameters from 31 to 10001. Also, each parameter has a defined lower bound B_{lb} and upper bound B_{ub}.

[1]Details of this study are outside the scope of this paper.

Therefore, for each of the 30 parameter a set of values D_i with the resolution as equal distance between the values is allowed. The n-dimensional input space over all parameters $\mathbb{D}^n = \times_{i=1}^{n} D_i$ is the corresponding Cartesian product (n = 30 in our case).

The objective is to find a parameter setting x within $\mathbb{D}^n$ that minimizes the braking distance $y(x)$ while simulating the straight-line braking maneuver, as defined in equation (1):

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

To apply algorithms for continuous input spaces, we consider this problem as quasi-continuous (Section 3.3).

1.2 Simulation

For the simulation of the vehicle dynamics, driver and environment, we use a two-track model implementation in Simulink [53]. The vehicle dynamics is partitioned into the mechanical vehicle and control systems, which include ABS and VDC models. The mechanical vehicle is implemented as a five-body model (car body and four wheels) moving in 16 degrees of freedom with the following components: equation of motion, tires, drive-train, aerodynamics, suspension, steering, and braking. The control systems consists of sensors, logic and actuators. The interaction of these modeled components enables the simulation of an integrated control system.

For tire models, we use the MF-Tyre/MF-Swift [49], which is based on the Pacejka's so-called Magic Formula [40] to accurately simulate the steady-state and transient behavior of the tires under slip conditions. The road surface is described by a curved regular grid (CRG) track [55], which defines the road width and elevation along a predefined reference line. CRG tracks can model 3D roads in great detail while keeping the memory usage to a minimum. On a standard workstation[2], one full simulation run takes about 15 to 20 minutes.

2 STATE OF THE ART

2.1 Optimization

Real-world optimization problems are often called black-box problems, meaning that the underlying objective function is unknown. There are many algorithms for solving single-objective (continuous) black-box optimization problems. Two different approaches can be distinguished [10]: *iterative optimization heuristics* and *one-shot optimization algorithms* – both are sampling-based heuristics. While iterative optimization heuristics evaluate solution candidates sampled adaptively according to the algorithm's logic to derive new solution candidates, one-shot optimization algorithms are non-adaptive and select the set of solution candidates prior to the first evaluation.

In this paper, we concentrate on one-shot optimization algorithms, that approach the problem in a fully parallel fashion. This is particularly advantageous for *computationally expensive problems* like many real-world problems, as it allows to distribute the simulations across multiple machines without the need for inter-machine communication, thereby the wall clock time is reduced by parallelization.

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

We distinguish between one-shot search and one-shot optimization with surrogate:

- In *one-shot search*, s solution candidates are evaluated in parallel and the best candidate x_{best} determines the performance $f(x_{best})$.
- In *one-shot optimization* (with surrogate), a surrogate model $\hat{f}$ is build based on these s solution candidates as training-data to approximate the actual function f. In combination with an optimization algorithm, a solution candidate $\hat{x}_{best}$ can then be derived. The performance is measured by computing its actual function value $f(\hat{x}_{best})$.

Commonly used strategies for generating solution candidates for one-shot search and optimization are grid sampling, uniform samples, Latin hypercube sampling [31], and samplings based on low-discrepancy sequences such as Sobol' [51] or Halton [12].

One-shot search has recently gained attention in hyperparameter optimization for machine learning models like deep neural networks and heuristic optimization techniques [3, 5, 7]. Moreover, surrogate models are traditionally used for expensive optimization problems in the engineering field [25, 28, 41, 42, 52].

2.2 Exploratory Landscape Analysis

A variety of benchmark problems are traditionally used for comparative assessments of black-box optimization algorithms [8, 18]. A current research direction is to relate the resulting algorithm performance assessments on these benchmark problems to relevant real-world problems from industry [29, 48]. The aim is to find the best optimization algorithm out of a set of algorithms for a specific problem, called the *algorithm selection problem* [46]. Therefore, a generalization of the characteristic of a problem is required.

In the context of continuous single-objective optimization, problems can be described by high-level properties, such as separability or multi-modality [33]. *Exploratory landscape analysis* (ELA) [32] quantifies these characteristics of an optimization problem with mathematical and statistical techniques. Therefore, features are calculated on a sample of points of the objective function. The feature values are sensitive to the sampling strategy and sample size, and a recommended strategy is Sobol' sequences [45]. Moreover, the feature values can vary because of the stochastic properties of the sampling strategies used. Many ELA features are implemented on different platforms or at least across several packages. The flacco package provides a wide collection within only a single package [24].

Applications of ELA are for example the analysis of similarities of problems across benchmark sets [50], the selection of an optimization algorithm [23] or the hyperparameter optimization of an optimization algorithm [2]. Muñoz et al. [37] gives an overview on the research field combining feature-based landscape analysis and algorithm selection for continuous black-box optimization problems.

3 METHODOLOGY

Our investigation is divided into two parts. In the first part we characterize our problem by computed landscape features. The aim is to find functions similar to our problem that may, for one-shot optimization algorithms, be of similar difficulty and therefore might serve as a proxy for the real-world problem. For comparison with our real-world problem we have chosen the 24 noiseless problems f_j, $j \in \{1, 2, ..., 24\}$ from the black-box optimization benchmark (BBOB) [13]. We consider the first instance of each problem and the default input space in n dimensions, $[-5, 5]^n$.

In the second part we then juxtapose the performance of one-shot search and one-shot optimization on our real-world problem and the 24 BBOB functions.

3.1 Exploratory Landscape Analysis

To characterize single-objective (continuous) optimization problems we use ELA. Therefore, we consider all six available feature sets in the flacco package [24], that are appropriate and can be computed for our 30 dimensional problem:

- classical ELA features (distribution, level, meta) [32],
- information content features [36],
- dispersion features [30],
- linear model features,
- nearest better clustering features [21, 43],
- principal component features.

Together, these six sets contain 68 single features. The large number of considered features can lead to redundant features [50]. Therefore, we conduct two feature selection steps [29]. First, we remove all features with a standard deviation of zero across all problems considered. In the second step, we remove highly correlated features by using the Spearman's rank correlation coefficient [27]. For each feature pair with a higher correlation than 0.99, the feature with the higher average correlation to the other features is removed.

To quantify the similarity between problems, 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}:

$$d(p_1, p_2) = \left\| F_{p_1} - F_{p_2} \right\|_2 . \tag{3}$$

Moreover, to weight the features equally, we re-scale the feature values to $[0, 1]$ according to the minimal and maximal values for the specific feature over all considered problems. Thus, we can quantify the similarity between our real-world problem and each of the BBOB functions.

As another approach to find groups of similar problems, [29] use the pairwise distances $d(p_1, p_2)$ between the problems to perform hierarchical clustering based on Ward's minimum variance method [19]. At the beginning each problems is one cluster. Then similar clusters are are merged to a combined clusters until all problems are grouped in one large cluster.

3.2 One-Shot Optimization

Algorithm 1 shows the steps of our implementation of the one-shot optimization with surrogate. During one run of the algorithm the surrogate type is predefined and will not be changed.

Initially, a design containing the input-data X is created (Line 1), e.g. with Sobol' sampling, and evaluated with the actual function f to get the output-data y (Line 2).

The performance of the surrogate model depends on its hyperparameters. Therefore, we perform Bayesian optimization [20, 34] to efficiently find the optimal hyperparameter values (Line 5). We

Table 1: Hyperparameter-space of the surrogate types random forest, support vector machine and multi-layer perceptron for the hyperparameter optimization with Bayesian optimization. The hyperparameters not listed remain in their default configuration.

Surrogate Type	Hyperparameter Description	Search Space
Random forest	Number of trees in the forest	$[100, 1000] \cap \mathbb{Z}$
	Maximal number of features considered for splitting a node	$[5, 30] \cap \mathbb{Z}$
	Minimum number of samples required to be at a leaf node	$[1, 5] \cap \mathbb{Z}$
Support vector machine	Kernel type	{rbf}
	Regularization parameter C	[1e-3, 1e+3], log-uniform
	Kernel coefficient gamma	[1e-4, 1e+2], log-uniform
Multi-layer perceptron (one hidden layer)	Number of neurons in hidden layer	$[3, 20] \cap \mathbb{Z}$
	alpha: L2 penalty (regularization term) parameter	[1e-2, 1e+2], log-uniform
	Activation function of hidden layer	{tanh, relu}
	Solver for weight optimization	{lbfgs}
	Maximum number of iterations of the solver	{5000}

assess the accuracy of a surrogate model by the average mean squared error (MSE) over a 10-fold cross-validation. Then the surrogate model with the optimized hyperparameters hp_{opt} is trained on the entire data-set (Line 6). For the training of the surrogate model, we scale the input-data to $[0, 1]^n$.

Since the training of the surrogate models is stochastic and so is the optimization of the hyperparameters, we perform $k = 1, ..., 10$ Bayesian optimizations with subsequent training on the entire data-set, obtaining 10 surrogate models $\hat{f}_k$ (Line 4).

To find the solution on the surrogate model, we use a (μ,λ) evolution strategy (ES) [1] as an optimization algorithm with a budget of 100000 surrogate evaluations (Line 8). Furthermore, to ensure that the optimum on the surrogate $\hat{f}_k$ is found, we execute $l = 1, ..., 10$ optimization runs on each surrogate model and select the best solution $\hat{x}_{best,k}$ from these 10 possible candidates $\hat{x}_{best,k,l}$ (Line 10).

Next, the best found solution on each surrogate model $\hat{x}_{best,k}$ is validated in the simulation. We obtain 10 validated solution candidates for a given data set (X, y). In the last step, the best solution $\hat{x}_{best}$ within these 10 validated solutions is selected (Line 12).

Algorithm 1: One-shot optimization with surrogate

1 $X \leftarrow$ Sampling ▷ generate design
2 $y \leftarrow f(X)$ ▷ evaluate function
3 $hp \leftarrow dict$ ▷ hyperparameters (Table 1)
4 **for** $k = 1, ..., 10$ **do** ▷ 10 times hp optimization
5 $\quad hp_{opt,k} \leftarrow BayesSearchCV(X, y, hp)$ ▷ optimized hp
6 $\quad \hat{f}_k \leftarrow \text{train}(X, y, hp_{opt,k})$ ▷ surrogate training
7 $\quad$ **for** $l = 1, ..., 10$ **do** ▷ 10 times evolution strategy
8 $\quad\quad \hat{x}_{best,k,l} \leftarrow \underset{x \in \mathbb{R}^n}{\arg\min}\ \hat{f}_k(x),\ B_{lb} \leq x \leq B_{ub}$
9 $\quad$ **end**
10 $\quad \hat{x}_{best,k} \leftarrow \underset{x \in \{\hat{x}_{best,k,l}\}}{\arg\min}\ \hat{f}_k(x)$ ▷ best solution on $\hat{f}_k$
11 **end**
12 $\hat{x}_{best} \leftarrow \underset{x \in \{\hat{x}_{best,k}\}}{\arg\min}\ f(x)$ ▷ best validated solution
Result: $\hat{x}_{best}$

3.3 Experimental Setup

For our experiment we have generated four "scrambled" Sobol' designs [39, 51] $X_i, i \epsilon \{1, 2, 3, 4\}$ of dimension $n = 30$ and 4096 samples each. Due to the parameter-specific resolution (Section 1.1), we round the values of the Sobol' designs and the found solution candidates for our real-world problem to the nearest possible values [3]. We then computed the associated responses y_i of our real-world problem using simulation.

As a compromise between accuracy and computational effort in ELA a sample size of $50 \times n$ to classify the BBOB functions with ELA features is recommended [22]. For our problem this results in a sample size of 1500. However, to further improve the accuracy, we use the highest considered sample size of 4096 in our investigation.

Before calculating the feature values F for the four data sets of our real-world problem, we scale the input-data X_i to $[-5, 5]^n$ according to the BBOB functions' input space. Moreover, to calculate the feature values of the 24 BBOB, we evaluated the BBOB functions with the scaled design X_1.

After the post-processing of the calculated features, 40 features remain for further investigation. Five features were removed because of a variance of zero and 23 features were removed because of a high correlation of over 0.99 to other features.

As surrogates for the one-shot optimization (Algorithm 1), we compare three standard regression techniques: random forest (RF) [6], support vector machine (SVM) [9], and multi-layer perceptron (MLP) [15] with one hidden layer. Table 1 contains the specific hyperparameters for the hyperparameter optimization which have a major influence on the MSE of the surrogate models.

The samples of a Sobol' design are taken one after the other from a Sobol' sequence, thus the generation of a Sobol' desing is a sequential process. This enables the investigation of different evaluation budgets by taking only the first samples out of each Sobol' design. For the one-shot optimization with surrogate we consider the three sample sizes $s \epsilon \{512, 2048, 4096\}$. This results in 4 * 3 = 12 scenarios and therefore, 360 derived solution candidates

[3] Evaluating also the BBOB functions f_{17} and f_{21} with the four rounded Sobol' designs would change $f(x_{best})$ by at most ±5% (relative improvement, not statistically result).

Table 2: Used software packages in our experiment.

Package Name	Description	Version
scipy [56]	Sobol' sequences, clustering	1.8.0
scikit-learn [11]	Surrogate models, t-SNE	1.0.2
scikit-optimize [16]	Hyperparameter optimization	0.9.0
pymoo [4]	Optimization with ES, CMA-ES	0.5.0
flacco [24]	Calculating features for ELA	1.8

from the surrogates for each problem (12 scenarios for 3 surrogate types and 10 runs, each).

We then juxtapose one-shot search with one-shot optimization by comparing the function value of the best sample $f(x_{best})$ with the actual function values of the solution candidates from the surrogates $f(\hat{x}_{best})$ for each scenario. We conduct these steps for one-shot search and one-shot optimization with surrogate (Algorithm 1) on our real-world problem and the 24 BBOB functions as well.

Table 2 gives an overview of the versions of the software packages used in our implementation. All packages are implemented in Python, except the flacco package which is implemented in R.

4 RESULTS

Figure 2 shows the distribution of the computed braking distances of our real-world problem samples y_i for the four input-designs $X_i, i \in \{1, 2, 3, 4\}$. The braking distances range from 32.1 m to 43.6 m. The shape of the distributions is very similar. Moreover, we find in each design several solution candidates with a braking distance below 32.5 m.

4.1 Exploratory Landscape Analysis

Using equation (3), we can compare the four sampled data-sets of our real-world problem to each of the 24 BBOB functions via the Euclidean distance between their computed feature values (Figure 3).

As figure 3 shows, similar and dissimilar BBOB functions to the real-world problem in terms of the distance can be identified. The most *similar* BBOB functions to our real-world problem are f_{17}

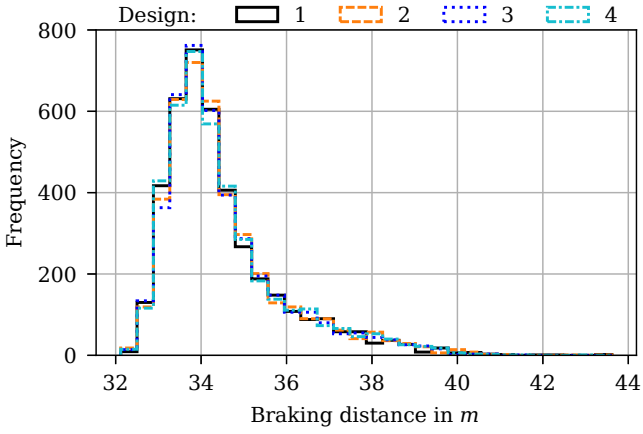

Figure 2: Distribution of the computed braking distances for the four Sobol' designs with each 4096 samples.

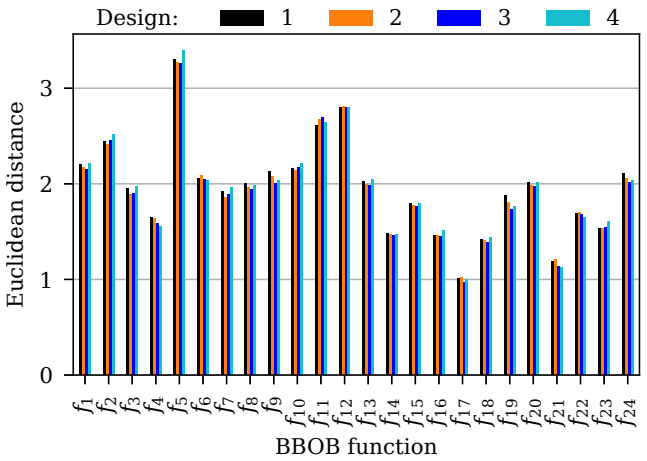

Figure 3: Euclidean distance of the 40 scaled feature values of the real-world problem (computed for the four designs with each 4096 samples) to the 40 scaled feature values of each of the 24 BBOB functions f_j.

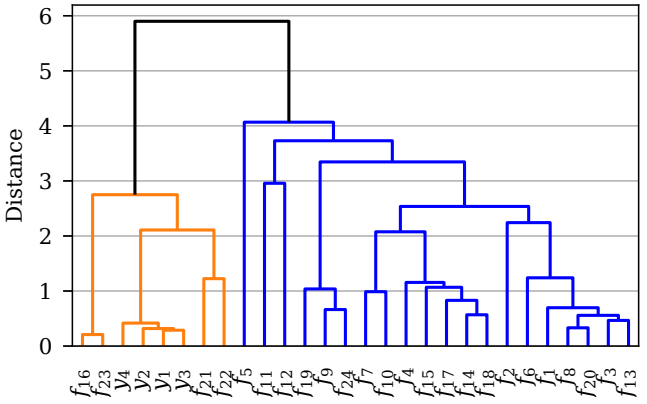

Figure 4: Dendrogram of the hierarchical clustering for the distances between the feature values of the four responses y_i of the real-world problem and the feature values of the 24 BBOB functions f_j.

with a distance around 1.0 and f_{21} with a distance around 1.2. The most *dissimilar* BBOB function is f_5 with a distance around 3.3. As a reference, the distances between the four designs of our real-world problem vary between 0.3 and 0.4.

These computed distances were used to perform hierarchical clustering and plotted as a dendrogram (Figure 4). As expected, the four designs of our real-world problem form one cluster with a relative small distance. The nearest cluster is f_{21} and f_{22}, and then f_{16} and f_{23}. These three clusters then form one of the two major clusters. The other major cluster contains the remaining BBOB functions.

f_{22} has a higher distance to the real-world problem than f_{17} (Figure 3), so we would expect f_{17} to be in the first major cluster with the real-world problem as f_{22} is, and not in the other major cluster. The reason is that the hierarchical clustering does not compare

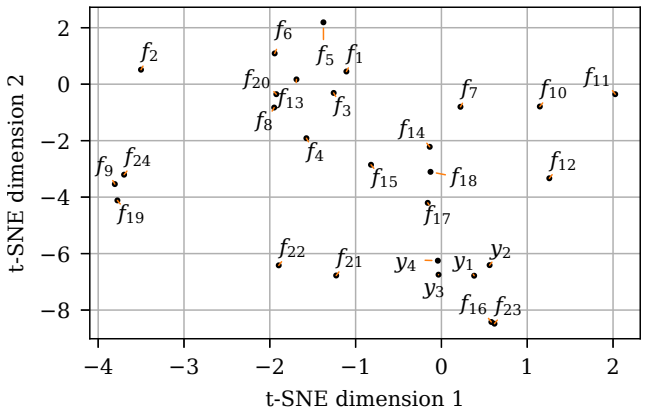

Figure 5: Two-dimensional projection of the 40-dimensional feature spaces through t-SNE visualization of the four responses y_i of the real-world problem and the 24 BBOB functions f_j.

single distances, but distances of clusters to each other and then the closest clusters are combined into one cluster and so on.

To visualize this in two dimensions we apply the t-SNE approach [54] to the 40-dimensional feature spaces (Figure 5). For computation we use the t-SNE implementation in scikit-learn [11] with a maximum number of 5000 iterations to ensure convergence and a perplexity of 10 because of the low data density of 4 + 24 problems.

The BBOB function f_{17} is closer to the four responses of the real-world problem y_i as f_{22} in the visualisation (Figure 5) and also with respect to the Euclidean distance (Figure 3). But f_{17} is first merged with f_{14} and f_{18} in a cluster, which is then further away from the cluster of the real-world problem. In contrast, f_{22} forms a cluster with f_{21}, which is then closer to the cluster of the real-world problem.

Therefore, clustering is well suited for finding similar groups of problems, but not individual functions that are similar to a particular problem.

4.2 Optimization

To juxtapose one-shot search and one-shot optimization with surrogate on our real-world problem, we generated four Sobol' designs (Section 3.3). Although all samples of the four Sobol' designs can be evaluated in fully parallel, the generation of the samples is a sequential process. This allows posteriori the identification of the best solution that the one-shot search would have found with a given budget of problem evaluations (Figure 6). Moreover, Figure 6 shows the best solution derived from the surrogates trained with the samples from the one-shot search for the four designs and at the three samples sizes {512, 2048, 4096}. On our real-world problem one-shot optimization with surrogate (Algorithm 1) finds a better solution than one-shot search, which leads to a smaller braking distance in 11 out of the 12 scenarios.

However, the variance of the optimization results on the surrogate can be tremendous, especially for the smallest considered sample size with 512 samples. This has the consequence that for design 2 the best solution is found with the lowest sample size of 512 samples at 32.0 m. Within the solution candidates from one-shot optimization for this scenario, the second best found solution is at 32.3 m. For design 3 the solution on the surrogate does not improve significantly over the sample size, while for design 1 it does. The best solution from the one-shot search improves constantly over the problem evaluations with minor differences across the four designs.

Figure 7 provides a closer analysis of the performance of each surrogate type. The best validated solutions from the MLP are never better than the best sample from the one-shot search. For design 1 and 3 SVM is never better than one-shot search. Within one Sobol' design RF always performs better than SVM and MLP except for design 4 with 2048 samples, where SVM is better. Moreover, compared to SVM and MLP, the solution candidates from RF are more often better than the solution from one-shot search, in two cases seven out of 10 (Figure 8). Nevertheless, often only one out of 10 solution candidates from one-shot optimization runs is better, for design 3 with 4096 sample none is better.

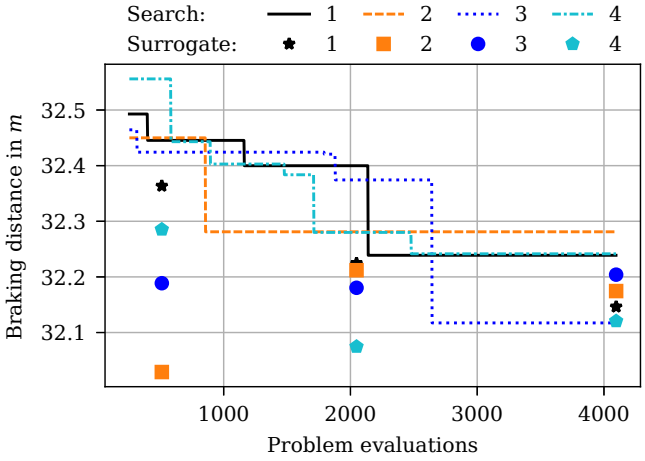

Figure 6: Smallest braking distance over problem evaluations for one-shot search and one-shot optimization with surrogate for the four designs and the sample sizes {512, 2048, 4096}.

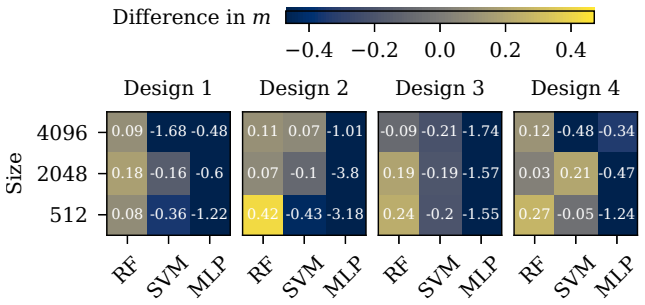

Figure 7: Difference $f(x_{best}) - f(\hat{x}_{best})$ in the braking distance between one-shot search and one-shot optimization with surrogate for the four designs, sample sizes {512, 2048, 4096} and surrogate types random forest (RF), support vector machine (SVM), multi-layer perceptron (MLP).

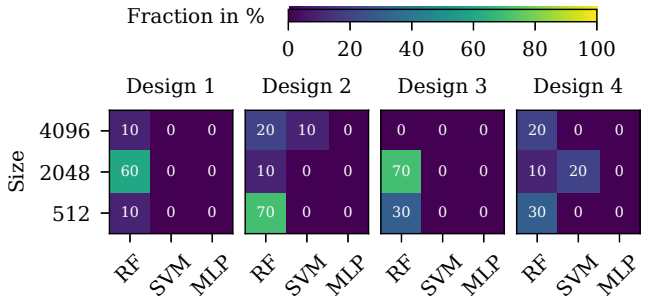

Figure 8: Fraction of the 10 one-shot optimization runs that are better than the one-shot search for the four Sobol' designs, the sample sizes {512, 2048, 4096} and surrogate types random forest (RF), support vector machine (SVM), multi-layer perceptron (MLP).

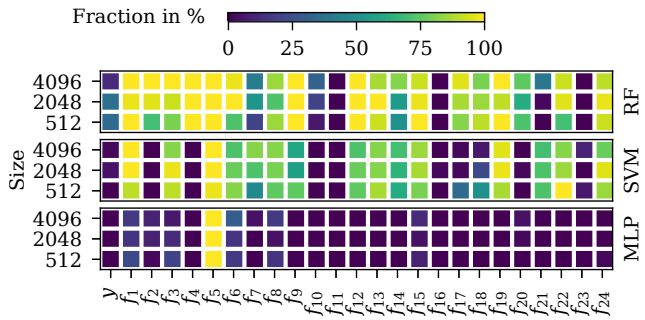

Figure 9: Average over the four designs for the fraction of the 10 one-shot optimization runs that are better than the one-shot search for the sample sizes {512, 2048, 4096} and the surrogate types random forest (RF), support vector machine (SVM), multi-layer perceptron (MLP) for the real-world problem y and the 24 BBOB functions f_i.

We conducted the same investigation on one-shot search and one-shot optimization with surrogate to each of the 24 BBOB functions. Figure 9 summarizes the results.

MLP as a surrogate type for one-shot optimization almost never leads to better solutions, only for the BBOB function f_5 MLP seems to be as good as RF and SVM, in fact it even outperforms them in terms of the absolute value found. The reason is that the minimum on MLP for a majority of the parameters lies near the upper or lower bounds of the input space, which actually is only the case for f_5. For example, for our real-world problem, f_{17} and f_{21} over 90 % of all parameter values from the one-shot optimization runs using MLP as surrogate lie near the upper or lower bounds, within 1 %. For comparison, when using RF as surrogate for our real-world problem or f_{17} below 2 % and for f_{21} below 7 % parameter values lie near the upper or lower bounds.

Furthermore, Figure 9 shows that the fraction of one-shot optimization runs performing better than one-shot search does not necessarily increase with higher sample size. This is not a contradiction, as both the search and the optimization with surrogate can find better solutions at higher sample sizes.

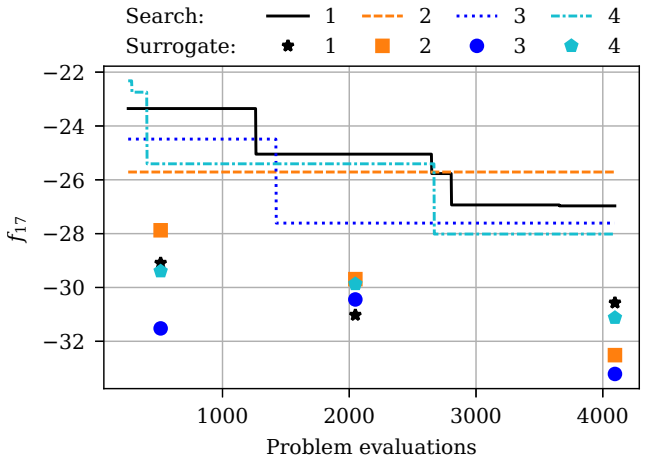

Figure 10: Smallest function values of f_{17} over problem evaluations for one-shot search and one-shot optimization with surrogate for the four designs and the sample sizes {512, 2048, 4096}.

For the BBOB function f_{17} RF and for f_{21} SVM finds more often better solutions, while SVM respectively RF achieve this less often. Thus, despite the similarity between our real-world problem and f_{21} in terms of their feature value distances, f_{17} apparently exhibits similar phenomena when it comes to surrogate type selection. However, the one-shot optimization is much less likely to find a better solution for our real-world problem than for f_{17}. One reason for this could be the similarity of the real-world problem to f_{16} and f_{23}, for which the search is almost always better.

The best found solution with one-shot optimization over all runs for f_{17} is -33.2 m (Figure 10). The actual optimum of f_{17} is -38.7. For comparison, the average solution over 1000 optimization runs, that can be found with an iterative optimization heuristic as CMA-ES [14] in default setting and a budget of 4096 function evaluations per run is -38.5 with a standard deviation of 0.3.

Besides the similarity between our real-world problem and f_{17}, there are some different phenomena. For example, the best solution found with the surrogates is without exception always better than the one-shot search and improves with increasing the sample size (Figure 10). Also, the fraction for which the search is better is much higher for the real-world problem than for f_{17} (Figure 9).

4.3 Surrogate Selection

The surrogate type and the surrogate model itself have a great influence on the quality of the solution found. We have measured the quality of different surrogate models by validating the derived solution candidates. To minimize the computational cost, a common alternative for selecting a surrogate is to measure the quality with a metric such as the average MSE over a 10-fold cross-validation and picking the best one. This approach assumes a correlation between the MSE and the quality of the solution found, which does not necessarily have to be the case (Figure 11).

For the real-world problem, the median MSE for MLP is only slightly worse than for RF and even better than SVM, yet the median

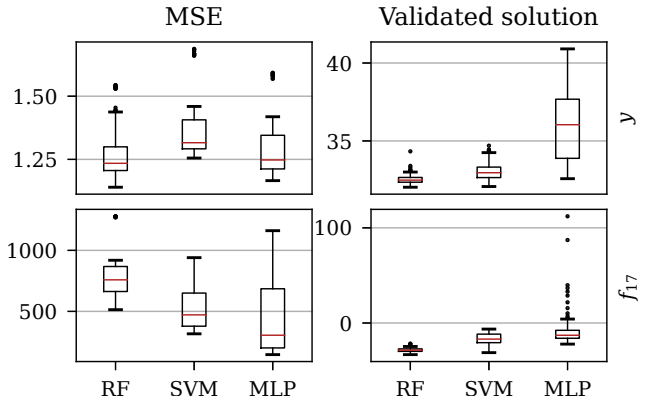

Figure 11: Box-plots of average mean squared error (MSE) over a 10-fold cross-validation and validated solution values from one-shot optimization over all 120 trained surrogates (10 runs for the three sample sizes and four designs) for surrogate types random forest (RF), support vector machine (SVM), multi-layer perceptron (MLP) for the real-world problem y and the BBOB function f_{17}.

Table 3: Average validated solution from one-shot optimization for different surrogate model selection strategies over the 12 scenarios (four designs and three sample sizes) for the real-world problem y and the BBOB function f_{17}.

Problem	Best model by MSE	Best RF model by MSE	Best model by validation
y	33.8	32.6	32.2
f_{17}	-8.3	-27.6	-30.5

of the validated braking distances for MLP is much worse than for RF and SVM. This is even more the case for the BBOB function f_{17}. MLP has the lowest MSE median but the highest median of validated solutions, lower solutions are better.

Figure 11 indicates for our real-world problem and the BBOB function f_{17} that selecting the surrogate depending on MSE would lead to worse performance in one-shot optimization. Indeed this is the case. Table 3 shows the average validated solution over the four designs and three sample sizes for the three strategies for selecting a surrogate model: picking best model by MSE, picking the best RF model by MSE and for comparison testing all and picking the model with the best validated solution.

Thus, selecting the surrogate type according to the MSE leads to worse solutions for our real-world problem and also for $\mathbf{f_{17}}$. Also within a surrogate type, selecting the model according to the MSE leads to worse solutions. But with the knowledge of the performance of MLP on the similar functions f_{17} and f_{21} to our real-world problem, MLP could have been omitted a priori as a surrogate type for the real-world problem to save computational effort without causing performance degradation.

5 CONCLUSIONS

In this paper, we analyzed the performance of one-shot search and one-shot optimization with a surrogate model on a real-world problem from the field of vehicle dynamic control systems and the 24 problems from the black-box optimization benchmark (BBOB).

One-shot optimization with surrogate is able to improve the found solution compared to one-shot search for the real-world problem and almost all BBOB functions, only for the BBOB functions f_{11}, f_{16} and f_{23} the search is significantly better. Because the surrogate training is stochastic, it is important to train several surrogates with the same data to obtain a set of solution candidates that can then be validated on the original problem. This is an effective data-driven way to improve the solution from the one-shot search, especially in terms of problem evaluations. However, it remains to investigate whether iterative optimization heuristics such as CMA-ES can find better solutions than the one-shot optimization algorithms for the real-world problem, and how much the disadvantage in terms of wall-clock time is due to the limitation of parallelization of the computationally expensive simulation.

Furthermore, we have observed that the mean square error (MSE) as a commonly used quality metric for regression models is not feasible for the surrogate selection in one-shot optimization. In particular, a multi-layer perceptron as a surrogate type for one-shot optimization performs worse than support vector machine or random forest for our real-world problem and a variety of the BBOB functions, despite its comparatively good MSE.

We used exploratory landscape analysis (ELA) to compute the similarity between problems by the Euclidean distance of their feature values. In a nutshell, a single similar function cannot explain all characteristics of the real-world problem, but it can give hints whether the use of a surrogate for optimization improves the solution compared to the one-shot search, and if so, which surrogate type is promising. To improve the similarity, a larger benchmark set could be an approach for further investigations, since the BBOB functions do not cover the entire problem space [50].

For automated surrogate type selection, [47] used a classifier trained with computed ELA features for a large set of benchmark problems as input and the best surrogate type as output. This classifier can then predict for a new problem the best surrogate type for one-shot optimization. We consider this as a promising research direction. However, selecting the right surrogate for one-shot optimization is a difficult task.

To improve the generality, it remains to investigate whether other sampling strategies and surrogate types confirm our results on the real-world problem and the 24 BBOB functions.

ACKNOWLEDGMENTS

The contribution of this paper was written as part of the joint 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

[1] Thomas Bäck. 1996. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, Inc, USA.

[2] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer. 2017. Per Instance Algorithm Configuration of CMA-ES with Limited Budget. In *GECCO '17: Proceedings of the Genetic and Evolutionary Computation Conference (ACM Digital Library)*, Peter A. N. Bosman (Ed.). ACM, New York, NY, 681–688. https://doi.org/10.1145/3071178.3071343

[3] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research* 13, 1 (2012), 281–305.

[4] Julian Blank and Kalyanmoy Deb. 2020. Pymoo: Multi-Objective Optimization in Python. *IEEE Access* 8 (2020), 89497–89509. https://doi.org/10.1109/ACCESS.2020.2990567

[5] Olivier Bousquet, Sylvain Gelly, Karol Kurach, Olivier Teytaud, and Damien Vincent. 2017. *Critical Hyper-Parameters: No Random, No Cry.* Technical Report. https://arxiv.org/pdf/1706.03200

[6] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[7] Marie-Liesse Cauwet, Camille Couprie, Julien Dehos, Pauline Luc, Jeremy Rapin, Morgane Riviere, Fabien Teytaud, Olivier Teytaud, and Nicolas Usunier. 2020. Fully Parallel Hyperparameter Search: Reshaped Space-Filling. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 1338–1348.

[8] Christodoulos A. Floudas, Pardalos, Panos M., Claire S. Adjiman, William R. Esposito, Zeynep H. Gümüş, Stephen T. Harding, John L. Klepeis, Clifford A. Meyer, and Carl A. Schweiger. 1999. *Handbook of Test Problems in Local and Global Optimization.* Nonconvex Optimization and Its Applications, Vol. 33. Springer, Boston, MA. https://doi.org/10.1007/978-1-4757-3040-1

[9] Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20, 3 (1995), 273–297. https://doi.org/10.1007/BF00994018

[10] Carola Doerr. 2020. *Theory of Iterative Optimization Heuristics: From Black-Box Complexity over Algorithm Design to Parameter Control.* Habilitation à diriger des recherches. Sorbonne Université. https://hal.sorbonne-universite.fr/tel-03168778

[11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

[12] J. H. Halton. 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* 2, 1 (1960), 84–90. https://doi.org/10.1007/BF01386213

[13] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. 2009. *Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions.* Technical Report RR-6829. INRIA. https://hal.inria.fr/inria-00362633v1/file/RR-6829.pdf

[14] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9, 2 (2001), 159–195. https://doi.org/10.1162/106365601750190398

[15] Simon Haykin. 1994. *Neural networks: a comprehensive foundation.* Prentice Hall PTR.

[16] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. 2021. scikit-optimize. (2021). https://doi.org/10.5281/ZENODO.5565057

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

[18] Momin Jamil and Xin She Yang. 2013. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation* 4, 2 (2013), 150–194. https://doi.org/10.1504/IJMMNO.2013.055204

[19] Joe H. Ward Jr. 1963. Hierarchical Grouping to Optimize an Objective Function. *J. Amer. Statist. Assoc.* 58, 301 (1963), 236–244. https://doi.org/10.1080/01621459.1963.10500845

[20] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization* 13, 4 (1998), 455–492. https://doi.org/10.1023/A:1008306431147

[21] Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. 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)*, Anna I. Esparcia-Alczar (Ed.). ACM, New York, NY, 265–272. https://doi.org/10.1145/2739480.2754642

[22] Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. 2016. Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models. In *GECCO '16: Proceedings of the Genetic and Evolutionary Computation Conference 2016 (ACM Digital Library)*, Frank Neumann (Ed.). ACM, New York, NY, USA, 229–236. https://doi.org/10.1145/2908812.2908845

[23] Pascal Kerschke and Heike Trautmann. 2019. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation* 27, 1 (2019), 99–127. https://doi.org/10.1162/evco_a_00236

[24] Pascal Kerschke and Heike Trautmann. 2019. Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package Flacco. In *Applications in Statistical Computing*, Nadja Bauer, Katja Ickstadt, Karsten Lübke, Gero Szepannek, Heike Trautmann, and Maurizio Vichi (Eds.). Springer, 93–123. https://doi.org/10.1007/978-3-030-25147-5_7

[25] Morteza Kiani and Ali R. Yildiz. 2016. A Comparative Study of Non-traditional Methods for Vehicle Crashworthiness and NVH Optimization. *Archives of Computational Methods in Engineering* 23, 4 (2016), 723–734. https://doi.org/10.1007/s11831-015-9155-y

[26] Heinz-Jürgen Koch-Dücker and Ulrich Papert. 2014. Antilock braking system (ABS). In *Brakes, Brake Control and Driver Assistance Systems*, Konrad Reif (Ed.). Springer Vieweg, Wiesbaden, 74–93. https://doi.org/10.1007/978-3-658-03978-3_6

[27] Stephen Kokoska and Daniel Zwillinger. 2000. *CRC Standard Probability and Statistics Tables and Formulae.* CRC Press. https://doi.org/10.1201/b16923

[28] Wei Li, Mi Xiao, Xiongbin Peng, Akhil Garg, and Liang Gao. 2019. A surrogate thermal modeling and parametric optimization of battery pack with air cooling for EVs. *Applied Thermal Engineering* 147 (2019), 90–100. https://doi.org/10.1016/j.applthermaleng.2018.10.060

[29] Fu Xing Long, Bas van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. 2022. Learning the Characteristics of Engineering Optimization Problems with Applications in Automotive Crash. In *GECCO '22: Proceedings of the Genetic and Evolutionary Computation Conference 2022 (ACM Digital Library).* ACM, New York, NY, USA. https://doi.org/10.1145/3512290.3528712

[30] Monte Lunacek and Darrell Whitley. 2006. The dispersion metric and the CMA evolution strategy. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation (ACM Conferences)*, Mike Cattolico (Ed.). ACM, New York, NY, 477. https://doi.org/10.1145/1143997.1144085

[31] M. D. McKay, R. J. Beckman, and W. J. Conover. 1979. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* 21, 2 (1979), 239–245. https://doi.org/10.2307/1268522

[32] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory Landscape Analysis. In *GECCO '11: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (ACM Conferences)*, Pier Luca Lanzi (Ed.). ACM, New York, NY, USA, 829–836. https://doi.org/10.1145/2001576.2001690

[33] Olaf Mersmann, Mike Preuss, and Heike Trautmann. 2010. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In *Parallel Problem Solving from Nature, PPSN XI*, Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph (Eds.). Springer, Berlin, 73–82. https://doi.org/10.1007/978-3-642-15844-5_8

[34] J. Močkus. 1975. On Bayesian Methods for Seeking the Extremum. In *Optimization Techniques IFIP Technical Conference*, G. I. Marchuk (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg and s.l., 400–404. https://doi.org/10.1007/978-3-662-38527-2_55

[35] Max D. Morris. 1991. Factorial Sampling Plans for Preliminary Computational Experiments. *Technometrics* 33, 2 (1991), 161–174. https://doi.org/10.1080/00401706.1991.10484804

[36] Mario A. Muñoz, Michael Kirley, and Saman K. Halgamuge. 2015. Exploratory Landscape Analysis of Continuous Space Optimization Problems Using Information Content. *IEEE Transactions on Evolutionary Computation* 19, 1 (2015), 74–87. https://doi.org/10.1109/TEVC.2014.2302006

[37] Mario A. Muñoz, Yuan Sun, Michael Kirley, and Saman K. Halgamuge. 2015. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences* 317 (2015), 224–245. https://doi.org/10.1016/j.ins.2015.05.010

[38] Tobias Niemz. 2007. *Reducing Braking Distance by Control of Semi-Active Suspension.* Dissertation. Technische Universität Darmstadt. http://tuprints.ulb.tu-darmstadt.de/912/

[39] Art B. Owen. 1998. Scrambling Sobol' and Niederreiter–Xing Points. *Journal of Complexity* 14, 4 (1998), 466–489. https://doi.org/10.1006/jcom.1998.0487

[40] Hans B. Pacejka and Egbert Bakker. 1992. The magic formula tyre model. *Vehicle System Dynamics* 21, sup001 (1992), 1–18. https://doi.org/10.1080/00423119208969994

[41] K. Palmer and M. Realff. 2002. Metamodeling Approach to Optimization of Steady-State Flowsheet Simulations: Model Generation. *Chemical Engineering Research and Design* 80, 7 (2002), 760–772. https://doi.org/10.1205/026387602320776830

[42] K. Palmer and M. Realff. 2002. Optimization and Validation of Steady-State Flowsheet Simulation Metamodels. *Chemical Engineering Research and Design* 80, 7 (2002), 773–782. https://doi.org/10.1205/026387602320776849

[43] Mike Preuss. 2012. Improved Topological Niching for Real-Valued Global Optimization. In *Applications of Evolutionary Computation*, Cecilia Di Chio (Ed.). Lecture Notes in Computer Science, Vol. 7248. Springer, Berlin and Heidelberg, 386–395. https://doi.org/10.1007/978-3-642-29178-4_39

[44] Rajesh Rajamani. 2012. *Vehicle Dynamics and Control* (2 ed.). Springer, Boston, MA. https://doi.org/10.1007/978-1-4614-1433-9

[45] Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. 2020. Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy. In *Parallel Problem Solving from Nature - PPSN XVI*, Thomas Bäck, Mike Preuss, André. Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann (Eds.). Lecture Notes in Computer Science, Vol. 12270. Springer International Publishing, Cham, 139–153. https://doi.org/10.1007/978-3-030-58115-2_10

[46] John R. Rice. 1976. The Algorithm Selection Problem. In *Advances in Computers*, Morris Rubinoff and Marshall C. Yovits (Eds.). Vol. 15. Elsevier Science & Technology, Saint Louis, 65–118. https://doi.org/10.1016/S0065-2458(08)60520-3

[47] Bhupinder Singh Saini, Manuel López-Ibáñez, and Kaisa Miettinen. 2019. Automatic Surrogate Modelling Technique Selection Based on Features of Optimization Problems. In *GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference Companion (ACM Digital Library)*, Manuel López-Ibáñez (Ed.). Association for Computing Machinery, New York, NY, USA, 1765–1772. https://doi.org/10.1145/3319619.3326890

[48] Ramses Sala and Ralf Müller. 2020. Benchmarking for Metaheuristic Black-Box Optimization: Perspectives and Open Challenges. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Piscataway, NJ, USA, 1–8. https://doi.org/10.1109/CEC48606.2020.9185724

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

[50] Urban Škvorc, Tome Eftimov, and Peter Korošec. 2020. Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. *Applied Soft Computing* 90 (2020). https://doi.org/10.1016/j.asoc.2020.106138

[51] I. M. Sobol'. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *Computational Mathematics and Mathematical Physics* 7, 4 (1967), 86–112. https://doi.org/10.1016/0041-5553(67)90144-9

[52] Xueguan Song, Guangyong Sun, Guangyao Li, Weizhao Gao, and Qing Li. 2013. Crashworthiness optimization of foam-filled tapered thin-walled structure using multiple surrogate models. *Structural and Multidisciplinary Optimization* 47, 2 (2013), 221–231. https://doi.org/10.1007/s00158-012-0820-6

[53] The MathWorks, Inc. 2015. Simulink. https://www.mathworks.com/products/simulink.html

[54] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. https://jmlr.org/papers/v9/vandermaaten08a.html

[55] VIRES Simulationstechnologie GmbH. 2020. Opencrg. https://www.asam.net/standards/detail/opencrg/

[56] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, Polat, VanderPlas, Jake, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2