SFB
823

Time efficient optimization of instance based problems with application to tone onset detection

Nadja Bauer, Klaus Friedrichs,
Claus Weihs

Nr. 85/2016

Discussion Paper

SFB
823

Time efficient optimization of instance based problems with application to tone onset detection

Nadja Bauer[a,*], Klaus Friedrichs[a], Claus Weihs[a]

[a]*Department of Statistics, TU Dortmund University, D-44221 Dortmund, Germany*

Abstract

A time efficient optimization technique for instance based problems is proposed, where for each parameter setting the target function has to be evaluated on a large set of problem instances. Computational time is reduced by beginning with a performance estimation based on the evaluation of a representative subset of instances. Subsequently, only promising settings are evaluated on the whole data set.

As application a comprehensive music onset detection algorithm is introduced where several numerical and categorical algorithm parameters are optimized simultaneously. Here, problem instances are music pieces of a data base.

Sequential model based optimization is an appropriate technique to solve this optimization problem. The proposed optimization strategy is compared to the usual model based approach with respect to the goodness measure for tone onset detection. The performance of the proposed method appears to be competitive with the usual one while saving more than 84% of instance evaluation time on average. One other aspect is a comparison of two strategies for handling categorical parameters in Kriging based optimization.

Keywords:
model based optimization, instance optimization, Kriging, onset detection, categorical parameters

1. Introduction

Parameter optimization is an important issue for industrial optimizations as well as for computer based applications. The established industrial optimization approach is based on a design of experiments where the trials are conducted according to a pre-defined plan [1]. The so called plan matrix should satisfy some special statistical criteria (like A- or D-optimality). After evaluation of the plan a relationship between the target variable and the influencing parameters

*Correspondence to: Department of Statistics, Vogelpothsweg 87, D-44227 Dortmund, TU Dortmund University, Germany. Tel.: +49231 755 7222; Fax: +49231 755 4387.

Email address: `bauer@statistik.tu-dortmund.de` (Nadja Bauer)

is fitted by a regression model. Usually, a linear model with interactions and quadratic terms is used for this task. The optimal parameter setting is then defined as the point with best model prediction value.

Despite the long success history of the optimization approach mentioned above [2, p. 467] it shows two essential weak points. First, the real experiments are often time and cost consuming. In some cases the real processes can be replaced by a sophisticated computer simulation. However, even one run of such simulation can take several days or months [3]. Hence, it is crucial to avoid trials in not promising areas and conduct them in the neighborhood of the supposed optimum instead. Second, the assumptions of the linear model are often violated so that it does not represent the true complex relationship between the target and the influencing factors. The sequential Model Based Optimization (MBO) – also called as efficient global optimization – which gained popularity with the seminal work of [4] is a modern nonlinear optimization technique.

The main idea of MBO is: After the initial phase – evaluation of some randomly chosen starting points – new points are proposed and evaluated iteratively with respect to a surrogate model fitted to all previous evaluations and an appropriate infill criterion to decide which point is the most promising. The most prominent surrogate model and infill criterion are the Kriging model in combination with the Expected Improvement (EI) criterion [4]. EI looks for a compromise of surrogate model uncertainty in one point and its estimated function value compared to the current optimum. We will call this combination classical MBO which is explained in detail in Section 2.

Nevertheless, also classical MBO has a number of shortcomings. For this work two of them are especially relevant: limitation to only numerical parameters and the absent concept for instance based function evaluations. The first problem is mainly caused by the standard Kriging model which works only with numerical parameters.

In this paper we aim to compare two simple approaches for handling categorical parameters, which we call naive Kriging and dummy Kriging, respectively. By naive Kriging categorical parameters are converted to numerical ones. This leads to an unnatural order with equal distances between the parameter levels. The dummy Kriging solution is more intuitive from a statistical point of view. Here, every categorical parameter is replaced by a set of dummy variables. In each MBO iteration only one dummy variable can have the value 1 while the others are set to 0.

Regarding the second MBO shortcoming the definition of an instance based problem is most important. Such situation occurs in applications where in each MBO iteration the target function has to be evaluated on a set of problem instances. The mean performance on this set is mostly considered as the target function value. However, it is not always advisable to evaluate the target function on all instances (as their quantity can be very high) and sampling a random subset in each iteration will lead to noise in the target function landscape. Hence, a strategy for a sensible selection of instances is needed.

In [5] we proposed such instance based optimization strategy. The main idea is that instead of evaluating of all instances, a few instances might already be sufficient to recognize unpromising parameter settings. Therefore, a subset of representative instances is chosen which is used in the consecutive stages for the classification of the proposed parameter settings into "good" and "bad" settings. Only if the probability of beating the current optimum is above a specified threshold, the setting is classified as "good" and evaluation is continued on all instances. For an adequate performance prediction, the representative subset should be as diverse as possible. This can be achieved, e.g., by clustering the instances with respect to their individual target function performance or other features. In this paper, we provide a new concept for reducing the representative subset's size without loosing its predictive ability. The instance based optimization is introduced more deeply in Section 3.

Here, the application for comparing classical MBO with the modified instance based MBO is tone Onset Detection (OD). Other applications for instance based optimization are, e.g., speech recognition (training of algorithms for different voices) or algorithm configuration (see, e.g, [6]). A tone onset is the time point of the beginning of a musical note. OD in music signals is an important step for many subsequent applications like music transcription and rhythm recognition. Several approaches have been proposed, but most of them can be reduced to the same basic algorithm just differing in the parameter settings [7, 8, 9, 10]. Most of them follow the same scheme: windowing the signal, calculating an Onset Detection Function (ODF) for each window and localizing the tone onsets by picking the relevant peaks of the ODF. Many numerical and categorical parameters are involved in this procedure like the window size, the window overlap and the applied ODF which have a strong influence on the algorithm performance. However, neither these influences nor the optimal parameter values are realistically quantified in most related publications (s. Section 4.4 for more details). Furthermore, the optimization results are mostly not validated on an additional test data base which can lead to over-fitting ([8]).

Obviously, optimization of OD is an instance based problem as each parameter setting is rated by its average performance over all music pieces of a data base. Hence, it seems to be an appropriate application for testing the proposed instance based approach. In contrast to our previous study in [5], the number of parameters of the optimized algorithm is enlarged. Among other things, the extended algorithm also enables online variants of OD. Furthermore, the optimization is conducted on a larger data base combining almost all data sets frequently used in previous studies. The onset detection algorithm, which will be optimized here, is briefly described in Section 4.

Validation of the considered optimization approaches is conducted in a sophisticated manner by repeatedly dividing the data base into training and test data. Section 5 presents research questions, comparison experiments, validation scheme and statistical analysis methods. Afterwards, the experimental results are analyzed in Section 6. Finally, in Section 7 the main findings are summarized and several ideas for future research are discussed.

2. Classical model based optimization

The aim of model based optimization is the minimization of a time intensive and highly complex target function $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$, $f(\boldsymbol{x}) = y$, $\boldsymbol{x} = (x_1, \ldots, x_d)^T$. Each function parameter x_i is assumed to be a value of a continuous feature[1] in a pre-fixed optimization interval $[\ell_i, u_i]$. The parameter space is the Cartesian product of the individual intervals: $\mathcal{X} = [\ell_1, u_1] \times \ldots \times [\ell_d, u_d]$. A possible parameter setting $\boldsymbol{x}_i \in \mathcal{X}$ is called a *point* and y_i is the target value in this point. A set of n points is called a *design* and is denoted with $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$. Moreover, $\boldsymbol{y} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^T$ is the vector of the target values on $\mathcal{D}$. Here, we assume only the deterministic target functions. If optimization of a noisy function is required, the readers are advised to acquaint themselves with the extensions provided in [11].

Algorithm 1 provides the scheme of MBO which can be summarized as follows: In the first step, an initial design with n points is evaluated and a surrogate model is fitted. The surrogate model is then used for the prediction of a new design point. As long as the optimization budget is not exhausted, a new point is chosen in the parameter space based on a so-called infill criterion derived from the surrogate model. The target function is evaluated in this point. The surrogate model is then updated on the design extended by the new point. The updated model is used for the next iteration. The point with the minimal target function value is taken as the result of the optimization. In what follows, these steps will be discussed in more detail.

Algorithm 1: Sequential model based optimization.

```
1 generate an initial design $\mathcal{D} \subset \mathcal{X}$;
2 evaluate $f$ on the initial design: $\boldsymbol{y} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^T$;
3 while optimization budget is not exhausted do
4     fit the surrogate model on $\mathcal{D}$ and $\boldsymbol{y}$;
5     find $\boldsymbol{x}^*$ with the best infill criterion value;
6     evaluate $f$ on $\boldsymbol{x}^*$: $y^* = f(\boldsymbol{x}^*)$;
7     update $\mathcal{D} \leftarrow (\mathcal{D}, \boldsymbol{x}^*)^T$ and $\boldsymbol{y} \leftarrow (\boldsymbol{y}, y^*)^T$;
8 end
9 return $y_{min} = \min(\boldsymbol{y})$ and the corresponding $\boldsymbol{x}_{min}$.
```

2.1. Initial design and optimization budget

The optimization budget should depend on the dimension of the target function (number of function parameters d) and is a trade-off between a good optimization result and the required computational time effort. The choice of the initial design should take into account that too small designs are not able to sufficiently scan the target function which might lead to local convergence. In

[1]The treatment of categorical features will be discussed later.

[12] the influence of the initial design size ($4d$ vs. $10d$) and the optimization budget ($20d$ vs. $40d$) is studied with respect to the optimization results. The authors recommend small initial designs and larger optimization budgets. Similar results are also concluded in [13] and [14].

Here, Latin Hypercube Sampling (LHS, [15]) designs are used for the initialization step. LHS designs are very popular in the computational optimization community due to their properties of uniformly covering the interesting parameter space and arbitrary selectable size (in contrast to the classical orthogonal designs). Because of the complexity of the optimization problem and the corresponding high computation time required for function evaluations, the size of the initial design is set to $5d$ and the number of sequential steps (iterations) is set to $20d$.

2.2. Surrogate model

A surrogate model is used for proposing a new design point. Theoretically, an arbitrary regression model can be used as a surrogate model. Very popular is the Kriging model (as proposed by [16]) which can model high dimensional multimodal function landscapes in good quality already with few points.

We use the ordinary Kriging modell [12]:

$$Y(\boldsymbol{x}) = \mu + Z(\boldsymbol{x}).$$

$Y(\boldsymbol{x})$ is a random variable and the error term $Z(\boldsymbol{x})$ is a Gaussian process expressing the uncertainty in $Y(\boldsymbol{x})$ having the properties:

$$\begin{aligned} &\mathbb{E}(Z(\boldsymbol{x})) = 0, \\ &K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = Cov(Z(\boldsymbol{x}), Z(\tilde{\boldsymbol{x}})) = \sigma_z^2 k(\boldsymbol{x}, \tilde{\boldsymbol{x}}), \end{aligned}$$

where $K(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ is the so-called covariance kernel, $k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ the spatial Correlation Function (CF), and σ_z^2 the process variance [17]. The CF models the structure in the data points and is, obviously, the most decisive part of Gaussian process specification ([18]). For multidimensional data the product correlation rule is applied:

$$k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \prod_{j=1}^{d} k_j(x_j, \tilde{x}_j).$$

Many different CFs were proposed in the last decades. See [18] for a detailed summary. We use the 3/2 - Matérn CF:

$$k_j(x_j, \tilde{x}_j) = \left(1 + \frac{\sqrt{3}|x_j - \tilde{x}_j|}{\theta_j}\right) exp\left(-\frac{\sqrt{3}|x_j - \tilde{x}_j|}{\theta_j}\right).$$

The greater the distance between x_j and $\tilde{x}_j$, the smaller is the value of $k_j(x_j, \tilde{x}_j)$. Therefore, the influence of already evaluated points on the prediction of new points shrinks with increasing distance. The parameter θ_j controls the speed of influence reduction: the greater θ_j, the greater is the influence region. The

CF parameters $(\theta_1, \ldots, \theta_d)^T$ as well as the process variance σ_z^2 are estimated by means of the Maximum-Likelihood method. See [12] for a detailed description.

The Kriging mean (model prediction, $\hat{f}(\boldsymbol{x})_{\mathcal{D}}$) and the Kriging variance (model uncertainty, $\hat{s}(\boldsymbol{x})^2_{\mathcal{D}}$) for any new point $\boldsymbol{x} \in \mathcal{X}$ are defined as:

$$\begin{aligned}\hat{f}(\boldsymbol{x})_{\mathcal{D}} &= \mathbb{E}\left[Y(\boldsymbol{x})|\boldsymbol{y}\right], \\ \hat{s}(\boldsymbol{x})^2_{\mathcal{D}} &= \mathit{Var}\left[Y(\boldsymbol{x})|\boldsymbol{y}\right].\end{aligned}$$

The index $\mathcal{D}$ clarifies the dependency of these two functions on already evaluated points. The detailed derivation of $\hat{f}(\boldsymbol{x})_{\mathcal{D}}$ and $\hat{s}(\boldsymbol{x})^2_{\mathcal{D}}$ can be found in [12].

2.3. Adapting Kriging for categorical parameters

One of the most important disadvantages of the standard Kriging model is its limitation to numerical influencing parameters. If there are also categorical parameters to be optimized, Kriging can be extended in several manners.

A naive variant is to treat categorical parameters as if they were continuous. First, each level is assigned to an integer. The proposed values of the corresponding continuous parameter in the sequential steps are rounded and converted back to the nearest categorical level. In this manner, the structure of the input $\boldsymbol{x}$ is not affected since it is an internal modification. This procedure has to be treated with care as we artificially define order and intervals between the levels which actually do not exist.

Another possibility is dummy Kriging, where each categorical parameter with m possible values is expressed by m different parameters (called dummy variables) which take the value 1 if the corresponding level is taken and 0 otherwise. These dummy variables are included into the vector of influence variables $\boldsymbol{x}$. Hence, the number of parameters in the surrogate model increases leading to longer run times for model fits.

2.4. Infill criterion

The infill criterion is a rating function which estimates how promising a target function evaluation at a given point is. Very intuitive criteria relate to the goodness of model predictions. However, a disadvantage of such a criterion would be a possible convergence to a local optimum.

Instead, typically the expected improvement criterion, as proposed in [4], is used as a compromise between exploitation of the surrogate model and exploration of the function landscape. The EI criterion supports global convergence [19] and becomes the standard criterion in many applications. The EI in a point $\boldsymbol{x}$ is defined as the expected value of a positive improvement of the target function in this point:

$$\begin{aligned}\mathrm{EI}(\boldsymbol{x}) &= \mathbb{E}[\max\{0, y_{min} - \hat{f}(\boldsymbol{x})_{\mathcal{D}}\}] \\ &= (y_{min} - \hat{f}(\boldsymbol{x})_{\mathcal{D}})\ \Phi\left(\frac{y_{min} - \hat{f}(\boldsymbol{x})_{\mathcal{D}}}{\hat{s}(\boldsymbol{x})_{\mathcal{D}}}\right) + \hat{s}(\boldsymbol{x})_{\mathcal{D}}\ \phi\left(\frac{y_{min} - \hat{f}(\boldsymbol{x})_{\mathcal{D}}}{\hat{s}(\boldsymbol{x})_{\mathcal{D}}}\right),\end{aligned}$$

where ϕ and Φ are the density and the cumulative distribution function of the standard normal distribution. y_{min} denotes the, so far, minimal value of the target function. The expected improvement should be maximized leading to the new design point

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\mathcal{X}}{\arg\max}\ \mathrm{EI}(\boldsymbol{x}).$$

2.5. Optimization of infill criterion

In each MBO iteration we would like to choose a new point by maximizing the infill criterion. To solve the corresponding non-linear optimization problem, we use the focus search algorithm implemented in the R package **mlrMBO** [20] which successively focuses the parameter space on the most promising regions. The main idea is: Generate a random LHS design of the size N_{points} on $\mathcal{X}$ and calculate the corresponding values of the infill criterion. Then, shrink the parameter space in each dimension to the environment of the best point. Iterate the shrinking N_{maxit} times. As this procedure can lead to a local optimum, focus search should be replicated several ($N_{restarts}$) times. As our final new point $\boldsymbol{x}^*$ we will take the best point over all iterations and repetitions. We will use the following settings: $N_{points} = 10\,000$, $N_{maxit} = 5$ and $N_{restarts} = 3$.

3. Model based optimization of instance based problems

In many applications the target function does not have to be evaluated just once in a point $\boldsymbol{x}_i \in \mathcal{X}$, but on a set of k problem instances $\mathcal{I} = (I_1, \ldots, I_k)$. Let $y_i^k = f(\boldsymbol{x}_i, I_k)$ define the individual performance measure of the kth instance in the ith point. Mostly, the mean response over all instances $\bar{y}_i = 1/k \sum_{j=1}^{k} f(\boldsymbol{x}_i, I_j)$ should be optimized. Obviously, one can apply MBO and calculate the mean response as the target. However, the problem with many instances is the high computational time needed for the evaluation. Therefore, we are looking for a short cut by excluding unpromising parameter settings without evaluating all instances. For simplification we will call promising settings "good" and unpromising ones "bad" in the following.

One possible approach is the instance based SMAC method (Sequential Model-based optimization for general Algorithm Configuration) of [6], where the initialization step is only based on the evaluation of a few randomly selected instances and newly proposed points are iteratively evaluated on random instances until there is an indication that the point is worse than the so far best. However, instead of a randomly chosen subset a systematic selection procedure seems natural. Therefore, in [5] we proposed a novel method for the optimization of instance based problems which is summarized in Algorithm 2 and explained in the following.

The first two steps of Algorithm 2 are the same as in Algorithm 1: an initial design $\mathcal{D}$ is generated and all instances $\mathcal{I}$ are evaluated on all points of $\mathcal{D}$. Now, our aim is to define a representative subset of instances $\mathcal{I}^{pretest} \subseteq \mathcal{I}$ in order to pretest the proposed new points on them. Here, we propose to cluster the k problem instances in $k_{pretest}$ clusters according to some features. Note, the

Algorithm 2: Model based optimization of instance based problems.

```
1  generate an initial design $\mathcal{D} \subset \mathcal{X}$;
2  evaluate $\mathcal{D}$ on all problem instances: $\boldsymbol{y} = (\bar{y}_i, \ldots, \bar{y}_n)^T$;
3  cluster all instances in $k_{pretest}$ clusters according to their features;
4  choose class representatives randomly for the pretest set $\mathcal{I}^{pretest} \subseteq \mathcal{I}$;
5  fit $M_{pretest}$: $y \sim f(\boldsymbol{x}, I_1^{pretest}) + \cdots + f(\boldsymbol{x}, I_{k_{pretest}}^{pretest})$;
6  apply forward variable selection for $M_{pretest}$ regarding the $R^2_{adj}$ measure
   by choosing $k'_{pretest} \leq k_{pretest}$ instances so that $M'_{pretest}$ has at least
   $R^2_{adj} = 0.98$ (if achievable);
7  while budget is not exceeded do
8      fit surrogate model $M_{sur}$ on $\mathcal{D}$ and $\boldsymbol{y}$;
9      find $\boldsymbol{x}^*$ by infill criterion optimization;
10     evaluate $\boldsymbol{x}^*$ on $k'_{pretest}$ selected instances;
11     predict mean performance: $\hat{\bar{y}}^* = M'_{pretest}(\boldsymbol{x}^*)$;
12     calculate 99% confidence interval: $[\hat{\bar{y}}^*_{low}, \hat{\bar{y}}^*_{upp}]$;
13     if $\hat{\bar{y}}^*_{low} \leq y_{min}$ then
14         evaluate $\boldsymbol{x}^*$ on remaining $k - k'_{pretest}$ instances:
           $\bar{y}^* = 1/k \sum_{j=1}^{k} f(\boldsymbol{x}^*, I_j)$;
15         update $\mathcal{D}$ and $\boldsymbol{y}$: $\mathcal{D} \leftarrow (\mathcal{D}, \boldsymbol{x}^*)^T$ and $\boldsymbol{y} \leftarrow (\boldsymbol{y}, \bar{y}^*)^T$;
16         update model $M'_{pretest}$ using the new observation;
17         update elements of $\boldsymbol{y}$ corresponding to "bad" points by the
           prediction values of $M'_{pretest}$;
18     else
19         update $\mathcal{D}$ and $\boldsymbol{y}$: $\mathcal{D} \leftarrow (\mathcal{D}, \boldsymbol{x}^*)^T$ and $\boldsymbol{y} \leftarrow (\boldsymbol{y}, \hat{\bar{y}}^*)$;
20     end
21 end
22 return $y_{min} = \min(\boldsymbol{y})$ and corresponding $\boldsymbol{x}_{min}$.
```

$k_{pretest}$ number should be chosen by the user and is a trade-off between good representatives (for large $k_{pretest}$) and computational time in sequential steps. If instances have a set of special features, they could be used for clustering. Otherwise, their individual target function performance can be used as feature. Subsequently, the pretest set could be compounded by randomly chosen representatives of each cluster.

A linear pretest model $M_{pretest}$ is fitted on $\mathcal{D}$ with the mean performance over all instances as the target and the individual performance of the pretest instances as influencing factors (line 5):

$$1/k \sum_{j=1}^{k} f\left(\boldsymbol{x}_i, I_j\right) = \beta_0 + \beta_1 \cdot f(\boldsymbol{x}, I_1^{pretest}) + \ldots + \beta_{k_{pretest}} \cdot f(\boldsymbol{x}, I_{k_{pretest}}^{pretest}) + \varepsilon.$$

In this manner we aim to build a model for predicting the overall performance of a new parameter setting by observing just the individual performance of the selected instances. In our previous experiments in [5] we found that $M_{pretest}$ has mostly very high values of the adjusted coefficient of determination measure (R^2_{adj}), even if the number of influencing factors is reduced. Hence, a variable selection step seems to be meaningful in order to further reduce the subset size (line 6). Here, forward variable selection regarding the R^2_{adj} measure is newly proposed. Stopping criterion is achieving of $R^2_{adj} = 0.98$. Through this step the number of pretest instances can be reduced to $k^{'}_{pretest}$. If $R^2_{adj} = 0.98$ is not achievable, $k^{'}_{pretest} = k_{pretest}$. The model on $k^{'}_{pretest}$ instances is denoted with $M^{'}_{pretest}$.

In the following repeated steps (lines 7-21), the surrogate model is fitted on $\mathcal{D}$ and $\boldsymbol{y}$ (line 8) and the new parameter setting $\boldsymbol{x}^*$ is proposed by optimizing the infill criterion (line 9). Then, the proposed point is first evaluated on the $k^{'}_{pretest}$ instances (line 10) and classified as "good" or "bad". This can be done in different ways. We base our decision on prediction intervals: for the point $\boldsymbol{x}^*$ we calculate the prediction $\hat{\bar{y}}^*$ and the 99% prediction interval[2] $[\hat{\bar{y}}^*_{low}, \hat{\bar{y}}^*_{upp}]$ based on $M^{'}_{pretest}$ (lines 11 - 12). If the lower limit of the prediction interval is smaller than the so far reached performance minimum, $\boldsymbol{x}^*$ is classified as a "good" point, otherwise as "bad".

The "good" points are evaluated on the remaining $k - k^{'}_{pretest}$ problem instances (line 14), $\mathcal{D}$ and $\boldsymbol{y}$ are updated (line 15), and the pretest model is newly estimated in order to include the new information of the additional observation (line 16). If a "bad" point is found, $\bar{y}^*$ is estimated only by $\hat{\bar{y}}^*$ and given back to the MBO loop (line 19). It is important to notice that these estimations are updated after each new update of the pretest model (line 16). Obviously, the proposed approach leads to saving of function evaluations compared to the classical MBO. On the other hand, the ability to find an optimal solution could be affected thereby.

4. Application problem: onset detection

The aim of onset detection (OD) is to recognize the time points of the beginnings of new musical tones. Figure 1 shows the amplitude of the same tone sequence played by piano (left plot) and flute (right plot), respectively. The onset times are marked by vertical lines. This figure illustrates that depending on the music instrument different signal features might to be sensible for OD task. At first, the classical OD algorithm is briefly introduced. Next, the goodness measures of OD and the data sets used for optimization are discussed. Finally, some details regarding the application of the proposed MBO to OD are provided.

[2]In [5] we compared 99% and 95% prediction intervals. The 95% prediction interval led to more as "bad" classified points. As better results were achieved with the 99% prediction interval, we propose to use this interval here.

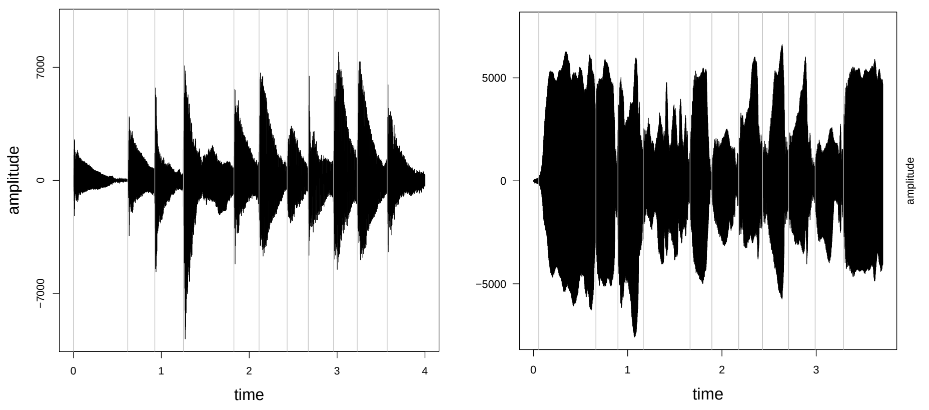

Figure 1: Amplitude of the same tone sequence played by piano (left) and flute (right). The vertical lines mark the true tone onset times.

4.1. Onset detection algorithm

The classical onset detection approach is presented in Figure 2 (see, e.g., [7, 21, 10, 5]). In each step, many algorithm parameters are defined which have to be optimized in the experimental part of this work. The considered ranges for the parameter values are discussed at the end of this section.

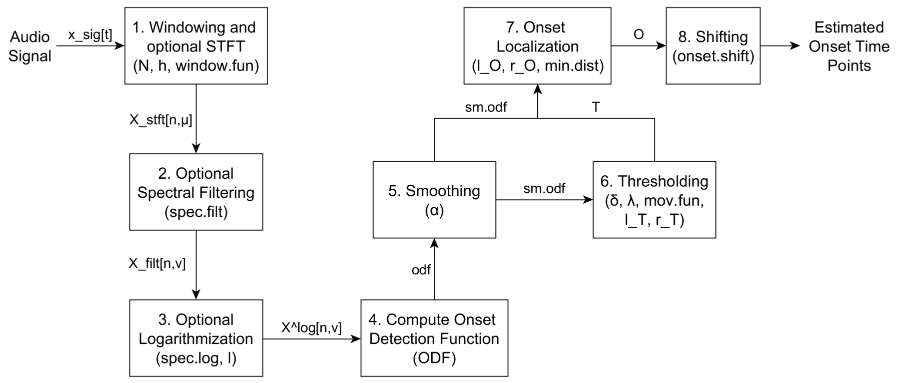
Audio Signal
x_sig[t]
1. Windowing and optional STFT (N, h, window.fun)
X_stft[n,μ]
2. Optional Spectral Filtering (spec.filt)
X_filt[n,v]
3. Optional Logarithmization (spec.log, l)
X^log[n,v]
4. Compute Onset Detection Function (ODF)
odf
5. Smoothing (α)
sm.odf
6. Thresholding (δ, λ, mov.fun, l_T, r_T)
sm.odf
T
7. Onset Localization (l_O, r_O, min.dist)
O
8. Shifting (onset.shift)
Estimated Onset Time Points

Figure 2: Eight steps of the classical onset detection approach. Each step has a several parameters to optimize which are given in parantheses.

There are two kinds of onset detection: online and offline. For online OD – in contrast to offline OD – no future signal information is allowed for distinguishing between 'onset' and 'no onset' in a current signal frame. We assume a digital audio signal sampled with a rate of $F_s = 44.1$ kHz. This signal is split in the first step into l (overlapping) windows (or frames) of length N samples. If required by the used OD feature, a Short-Time Fourier Transformation (STFT, [22]) is applied in each signal frame:

$$X_{stft}[n,\mu] = \frac{1}{N}\sum_{k=1}^{N} x_{sig}[h \cdot (n-1) + k] w_N(k) e^{-2\pi i \mu k/N}, \tag{1}$$

where x_{sig} is the music signal and $X_{stft}[n,\mu]$ is the Fourier coefficient (a complex number) of the μth frequency bin in nth frame, $n = 1,\ldots,l$. The hop size parameter h determines the distance in samples between the windows. $w_N()$ is the window function (parameter *window.fun* for optimization) which is used to weight the signal amplitude in the frames. See [23] for an overview of such functions.

In the second step, the output of the STFT can be optionally modified. According to [21] we apply spectral filtering and logarithmize the spectral magnitudes. Hence, pre-processing is used only for spectral based OD featured. For spectral filtering (parameter *spec.filt*) a filter bank is applied to the spectral magnitudes (absolute values of the Fourier coefficients) which bounds the frequency bins according to the semitones of the western music scale:

$$|X_{filt}[n,\nu]| = \sum_{j=1}^{N/2} |X_{stft}[n,\mu]| \cdot F[\mu,\nu]. \tag{2}$$

Spectral filtering reduces the number of frequency bins for the subsequent feature calculation.

Furthermore, the (optionally filtered) spectral coefficients can be logarithmized (parameter *spec.log* in step 3):

$$|X^{log}[n,\nu]| = \log_{10}(\ell|X_{stft}[n,\nu]| + 1) \tag{3}$$

where ℓ is a compression parameter.

The computation of an onset detection function in a window of the preprocessed signal is often called reduction (step 4), since after this step not the signal is analyzed anymore but only the ODF values. Many ODFs are based on the comparison of neighboring windows. An increase of an ODF generally indicates a tone onset, a decrease a tone offset. However, also offset information can improve onset detection (see [24]). The algorithm parameter *od.fun* has 18 levels corresponding to 18 ODFs considered here. These ODF's are based on the following signal features, some of them use tone offset information: zero-crossing rate, absolute maximum [25], amplitude energy, weighted spectral energy [26], spectral centroid [27], spectral spread [27], spectral skewness [27], spectral flux [7, 8], spectral Euclidean distance [28], phase deviation [29, 8] and complex domain [8]. For example, one of the most popular OD features – spectral flux – is defined as follows:

$$Spec.Flux(n) = \sum_{j=1}^{N/2} H(|X[n,\nu]| - |X[n-1,\nu]|), \tag{4}$$

where $H(x) = (x + |x|)/2$ is a filter which sets the decrease of the spectral magnitude in a frequency bin to zero.

The aim of the normalization (step 5) is to transform the *odf* feature vector into a standardized form for the subsequent thresholding. At first, exponential smoothing with parameter α can be applied, where for $\alpha = 1$ the time series stays unchanged and for $\alpha = 0$ all values of a feature are equal:

$$\begin{aligned} sm.odf_1 &= odf_1, \\ sm.odf_n &= \alpha \cdot odf_n + (1-\alpha) \cdot sm.odf_{n-1}. \end{aligned} \tag{5}$$

Moreover, for offline OD the normalization of *sm.odf* is meaningful utilizing e.g., the maximum of the *odf* feature vector.

Since not every local maximum of an ODF represents an onset, the threshold function aims at the distinction between relevant and non-relevant peaks (step 6). A fixed value for the threshold is unfavorable since the method could then not react to dynamic changes of the signal. Instead, moving threshold functions are widespread [10]:

$$T_n = \delta + \lambda \cdot mov.fun(|sm.odf_{n-l_T}|, \ldots, |sm.odf_{n+r_T}|),$$

where the parameter *mov.fun* (moving function) is either the ‘median’, ‘mean’ or ‘p-quantile’. l_T and r_T are the numbers of windows to the left and to the right, respectively, of the nth window.

In the following step the tone onsets are localized in the windows where *sm.odf* values exceed the threshold and are a local maximum of a certain window sequence (step 7). Following [21] we also use a third condition: A minimum distance *min.dist* (in number of windows) between the actual window and the window of the previous tone onset $i_{prev.onset}$ should be exceeded. To summarize:

$$O_n = \begin{cases} 1, & \text{if } sm.odf_n > T_n \text{ and} \\ & sm.odf_n = \max(sm.odf_{n-l_O}, \ldots, sm.odf_{n+r_O}) \text{ and} \\ & n > n_{prev.onset} + min.dist, \\ 0, & \text{otherwise.} \end{cases}$$

$\boldsymbol{O} = (O_1, \ldots, O_l)^T$ is the tone onset vector and l_O and r_O are additional parameters, namely the number of windows to the left or right of the actual window, respectively, which define the region for local maxima. The left limits of signal frames with $O_n = 1$ are taken as the time points of the tone onsets.

In [21], it is proposed to report the onsets one window later as actually detected. The authors argue that some features can increase earlier than a human listener would firstly recognize and note a tone onset. For the window length used in [21] this would correspond to a fix time shift of 10 ms. In our work we will consider the parameter *onset.shift* for optimization (step 8).

In contrast to the most papers on the topic, we do not fix window length N and hop size h a priori but optimize them. This means, that parameter settings corresponding to the number of windows (like r_T) could stand for very different time periods depending on N and h. Therefore, all such parameters are re-defined according to the desired time length, N and h. Hence, we will not consider the parameters r_T, l_T, r_O, l_O, and $min.dist$, but the times $t(r_T)$, $t(l_T)$, $t(r_O)$, $t(l_O)$, and $t(min.dist)$.

In what follows, the regions of interests for all parameter are defined. We will consider $N = 512, 1024, 2048$ and 4096 samples as frame length (powers of 2 are chosen in order to apply the fast Fourier transform). The hop size h lies between $N/10$ and N. The following window functions are considered for *window.fun*: 'Uniform', 'Hamming', 'Blackmann' and 'Gauss' (with standard deviation $\sigma = 0.4$). Parameters *spec.filt* and *spec.log* have two settings 'yes' and 'no'. Furthermore, $\ell \in [0.01, 20]$ and the smoothing parameter α varies in $[0, 1]$. Regarding the thresholding, we optimize $\delta \in [0, 10]$ and $\lambda \in [1.1, 2.6]$ for $mov.fun$ = 'median' or 'mean' and fix λ to 1 while optimizing $p \in [0.8, 0.98]$ for $mov.fun$ = 'p-quantile'. The shifting parameter *onset.shift* lies in $[-0.01, 0.02]$ s. Negative values are allowed as it can not be excluded that some features report tone onsets with a certain time delay. For online applications, $t(r_O)$ and $t(r_T)$ are set to 0 s. In the offline case and universally for $t(l_O)$ and $t(l_T)$ these intervals are set to [0, 0.5] s. The region of interest for the parameter $t(min.dist)$ is $[0, 0.05]$ s.

To summarize, there are two application problems to optimize: online and offline OD. Offline OD has a set of 17 parameters while for online OD two parameters are fixed to 0 (i.e., 15 parameters remain for the optimization). In both cases four parameters are categorical and the remaining are numerical.

4.2. Goodness measures for onset detection

In the most onset detection related publications the goodness of onset detection is measured by the F-measure (e.g., see [8]):

$$F = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \quad F \in [0, 1],$$

where TP, FP, and FN stand for the number of *true positive cases*, *false positive cases*, and *false negative cases*, respectively, and $F = 1$ represents an optimal detection. A found tone onset is correctly identified if it is inside a tolerance interval around the true onset. We use here ± 25 ms as the tolerance [21] while ± 50 ms setting is also frequently applied in the literature [7, 8].

Note that the *true negative* cases are not taken into account in the calculation of the F-measure. Another disadvantage is that the distance between true and estimated onsets is only taken into account via the tolerance region. An alternative evaluation measure is therefore the mean (relative) deviation of the estimated onset times from the true ones. This will be called D-measure in what follows. We will optimize the F-measure and use the D-measure only as an additional evaluation feature, e.g., for the clustering in Section 4.4.

4.3. Music data base

The greatest challenge with respect to the creation of a music data base is the necessity of information about the onset times. There are at least two ways to generate audio data with the corresponding onset times. In the literature, music pieces are manually annotated most of the time, leading to only small numbers of annotated pieces and possible annotation errors. The second possibility is the use of the MIDI format, offering the advantage to have many such music pieces at one's disposal. There are different programs available which can generate audio recordings in the so-called wave format from MIDI files using instrument specific signal models. Naturally, a music piece generated synthetically in this way will normally not realistically mimic real music recordings.

The aim of the composition of the data base for our optimization is to cover as many music aspects as possible, e.g., way of generation (annotated real music, synthesized MIDI), type of instrument, degree of polyphony, tempo, music genre, and music style. However, it is not meaningful to combine the labeled (real) music pieces with recordings generated from MIDI format as they differ in the onset annotation principle. While for manually annotated pieces the definition of the perceptual tone onset can be applied (i.e., time point where a human listener can firstly recognize a tone onset, [8]), for MIDI pieces the definition of the physical onset is more appropriate (i.e., time point of the first amplitude rise of the new tone, [8]). Hence, we expect that due to delayed annotation in the first case the optimal parameter set can differ from the second case. For this reason, we consider two data bases which will be referred to as WAV and MIDI data sets.

The WAV data base consists of three frequently used manually annotated data sets: data base introduced in [7] with 23 pieces, in [9] with 92 recordings and in [21] with 206 music pieces. Altogether it consists of 2 750 tone onsets. Many music instruments (like wind or string) and music styles (like European or oriental) are represented in this data set. According to [21], we aggregated true onset times reported within 30 ms to only one tone onset.

The MIDI data base includes the German folk song data base proposed in [28] with 24 music pieces, the MIDI data collection introduced in [5] with 200 pieces and the music epoch data base mentioned in [14] with 22 recordings. Furthermore, 20 pieces were additionally generated for this work. The most MIDI files were converted to wave files using the *MIDI to WAV Converter*[3] while also the *RealConverter* proposed in [25] was used. There are overall 63 067 tone onsets in the MIDI data base.

4.4. Onset detection as instance based problem

Although the necessity of parameter optimization arises in nearly all studies on onset detection, they usually just examine a subset of all possible parameters, while other parameters are set to some fixed empirical values, which were determined in preview studies or are frequently used in the OD community. Merely a few numerical parameters are then optimized via grid search or some generic algorithms. This procedure could lead to a local optimum. We aim,

[3]`http://www.maniactools.com/soft/midi_converter/index.shtml`.

in contrast, to optimize all parameters simultaneously in order to enable lobal optimization.

The goal of OD optimization is to find the optimal values of the algorithm parameters mentioned in Section 4.1 (input $\boldsymbol{x}$) with respect to the mean F-value (output y). The parameter space $\mathcal{X}$ of the offline onset detection problem is the Cartesian product of the regions of interest of $d = 17$ parameters introduced above (in the online case: $d = 15$). For each problem instance (music piece) the target function $f : \mathcal{X} \subset \mathbb{R}^{d1} \times \mathbb{N}_0^{d2} \to [0, 1]$ is defined as the F-value of the associated OD algorithm (characterized by a point $\boldsymbol{x} \in \mathcal{X}$) for this instance, where $d1$ ans $d2$ are the numbers of the numerical and the categorical parameters, respectively. The levels of the categorical parameters are coded with integer numbers (e.g., levels of the parameter *window.fun* are coded as follows: $0 =$ 'Uniform', $1 =$ 'Hamming', $2 =$ 'Blackmann' and $3 =$ 'Gauss').

For the classical MBO approach the OD performance in $\boldsymbol{x}$ is the averaged F-value over the considered data base of k music pieces $\mathcal{I} = (I_1, \dots, I_k)$: $f(\boldsymbol{x}) = 1/k \sum_{j=1}^{k} f(\boldsymbol{x}, I_j)$. As the F-value has to be maximized, we will minimize the negated F-value.

With respect to Algorithm 2 more details have to by clarified regarding the OD application. The same surrogate model and infill criterion as mentioned in Section 2 for the classical MBO are used. Although an instance based problem is given, the target variable is defined as the mean performance over all instances (real performance values for "good" points and estimated values for "bad" points), so that the usual Kriging model and expected improvement criterion can be applied. Categorical parameters are transformed into numeric or dummy variables according to the applied approach (naive or dummy Kriging).

In line 3 all instances should be clustered in $k_{pretest}$ clusters. In [5] we used only the individual F-values of instances in the initial design as classification features. Here, we extend this step by considering also the individual D-values (see Section 4.2). Instances in the same cluster should have similar responses (F- and D-values) for most parameter configurations. In this way we strive to reach the greatest diversity of the music pieces in this pretest subset. The k-means algorithm [30] is used as the clustering approach in line 3.

The size of the pretest subset ($k_{pretest}$) is set to 5% of the training data volume. This corresponds to 10 pieces for the WAV data set and 8 pieces for the MIDI data set (as in the training phase only 2/3 of the whole data set is used, s. Section 5).

5. Research questions and experimental design

The following main research questions will be studied in this work which can be divided in two classes: optimization questions (OQ) and application questions (AQ):

- **OQ1**: Which surrogate model results in better optimization performance: naive or dummy Kriging? (Section 2.3)

- **OQ2**: Does the proposed MBO for instance based problems achieve the same values of the target function as the classical MBO, despite the fewer number of function evaluations? (Sections 2 and 3)
- **AQ1**: Is there a difference in performance between online and offline OD? (Section 4.1)
- **AQ2**: Is there a difference in performance between WAV and MIDI data? (Section 4.3)

This results in $2^4 = 16$ 'optimization strategy - application problem' combinations. In order to show the efficiency of MBO, a random search with the same number of function evaluations (generated via an LHS design) is also conducted for online onset detection. Furthermore, a simple reference strategy for the proposed MBO is also implemented: after the evaluation of the sequential design the optimization in sequential steps is done merely on the $k_{pretest}$ randomly chosen instances.

A very important aspect when comparing many optimization strategies is dealing with over-fitting to the training data. In order to avoid a very good fit on the training data and a much worse performance on new test data, it is essential to use the resampling technique [31]. The most simple resampling technique is the holdout approach where a part of the data (usually 2/3) is used for training and the remaining part for testing. Very popular is the k-fold-cross-validation procedure: Split the data in k disjunct blocks of equal size, conduct the optimization procedure on the training data of $k-1$ blocks and validate the best found parameter setting on the remaining block (test data). In this manner, k goodness values (i.e., for each test part) are produced which either can be analyzed as a vector or averaged to only one measure. In the former case two competitive approaches can be compared according to the distribution of the goodness values. As remarked in [32], the k goodness values are not independent through the block structure of the cross-validation. However, independence is assumed in almost all statistical test.

In order to achieve the independence mentioned above, we replicate the holdout approach 30 times and get, hence, for each optimization strategy a vector of 30 goodness values. In our case, the training data set is used for finding the optimal parameter set of the online or offline OD algorithm while this setting is then applied to the test data. The mean F-value over the music pieces of the test data is then the goodness value of the applied optimization strategy in the respective holdout-replication. In one replication of the holdout approach the same initial design is used for all strategies in order to ensure uniform starting conditions.

As the F-values are not assumed to be normally distributed, the Wilcoxon signed rank test [33, p. 128 ff.] is considered as a non-parametric alternative to the t-test. In accordance with [33] (p. 132) the sample size of 30 observations is sufficient for the desired asymptotic property of the test statistics. Note that although a multiple testing problem occurs here (as some samples have to be applied for testing several times) we waive the methods for holding the global significance level (like Bonferroni-Holm procedure). Instead, the results of the statistical tests will not be over-interpreted but used rather in a descriptive way. The significance level is assumed to be 5%.

The experiments were executed in parallel using the **BatchExperiments** R package [34] on the Linux-HPC cluster system[4] of TU Dortmund University. The classical MBO optimization is conducted using a developing version of **mlrMBO** R package [20]. The OD algorithm as well as MBO for instance based problems are implemented in the *R* programming language [35] and can be provided on request.

6. Results

Here, we successively answer the research questions mentioned in Section 5. Figure 3 illustrates the distribution of the validated F-values for the classical MBO optimization strategy with naive and dummy Kriging as surrogate models. The left five boxplots correspond to the results on the WAV data bank while the right five boxplots consider the MIDI data set. The x-axis provides the OD kind: online or offline. The reference strategy – random search – was applied just for the online OD detection in order to demonstrate the essential advantage of model based optimization.

The first optimization question and both application questions can be analyzed descriptively (s. Figure 3). It seems that there is no big difference in the optimization performance between the two surrogate models (see the first optimization question, **OQ1**). Moreover, naive Kriging shows slightly better F-values than the statistically more sensible dummy Kriging. The Wilcoxon signed rank test for the equality of median was applied to four hypotheses (online and offline OD on both data sets) while all tests show p-values noticeably greater than 0.05. Hence, no significant difference can be stated between the optimization results of naive and dummy Kriging models.

As a further comparison aspect regarding the surrogate models, we will consider the optimization time. Table 1 shows the mean time distribution over the three optimization elements: fitting the surrogate model, EI optimization and function evaluations (see lines 4, 5 and 6 of Algorithm 1, respectively). Time analysis was conducted only for the WAV data set and considers only the sequential steps of MBO (lines 3-8 of Algorithm 1). As the EI optimization algorithm (focus search, see Section 2.5) depends on the number of influencing parameters which is much higher for the dummy Kriging approach, a relevant additional time expense can be noticed for dummy Kriging compared to naive Kriging (3.5 hours more on average for one optimization run). Furthermore, 84.6% of the computational time is needed on average for function evaluations in the case of online OD while for offline OD this ratio is 71%. This illustrates that OD is a typical case of time expensive optimization problems.

[4]`http://lidong.itmc.tu-dortmund.de/ldw/index.php?title=System_overview&oldid=259`.

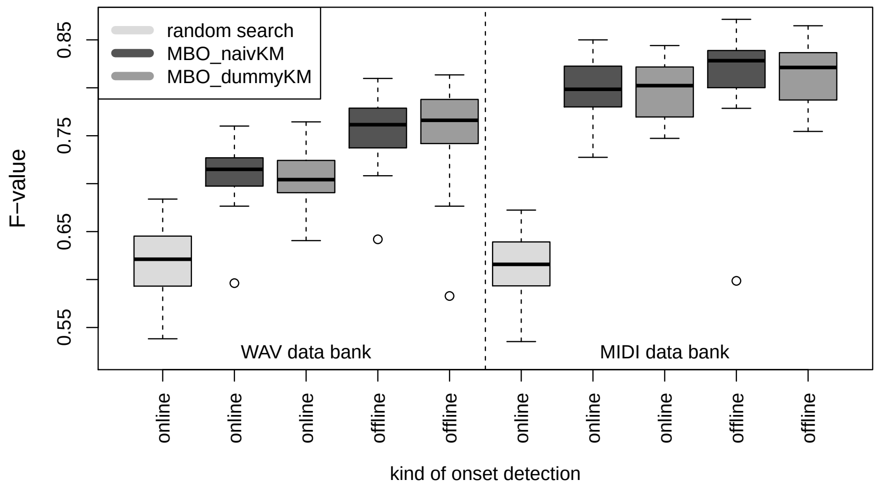

Figure 3: Validated results of the classical MBO with naive and dummy Kriging as surrogate models for online and offline OD application problems. The left five boxplots: results on the WAV data bank; the right five boxplots: results on the MIDI data bank.

Table 1: Mean time in hours for surrogate model fit, EI optimization and function evaluation in one MBO run on the WAV data base. Time for function evaluations on initial design is not considered here. naive: naive Kriging surrogate model, dummy: dummy Kriging.

	model fit		EI optimization		function evaluation	
OD kind	naive	dummy	naive	dummy	naive	dummy
online	1.866	3.602	4.348	6.081	34.214	33.274
offline	3.028	5.251	5.922	7.942	21.876	23.580

Results of the statistical tests and the time analysis in Table 1 indicate that naive Kriging seems to be more beneficial than dummy Kriging because of its simplicity and efficiency. Therefore, further optimization experiments will be conducted just with naive Kriging model. However, this finding might not be generalized to all optimization problems. The good performance of naive Kriging can probably be explained through the size and type of the used initial design. For online OD, e.g., the initial design consists of 75 points, so that each level of the categorical parameter *od.fun* (with 18 levels) occurs approximately four times (according to the construction of LHS designs). This seems to be sufficient for fitting the function landscape appropriately. It would be interesting for further research to study the behavior of naive and dummy Kriging models under different circumstances.

Regarding the first application question (**AQ1**), as expected, offline OD achieves clearly better F-values than the offline OD while this difference seems

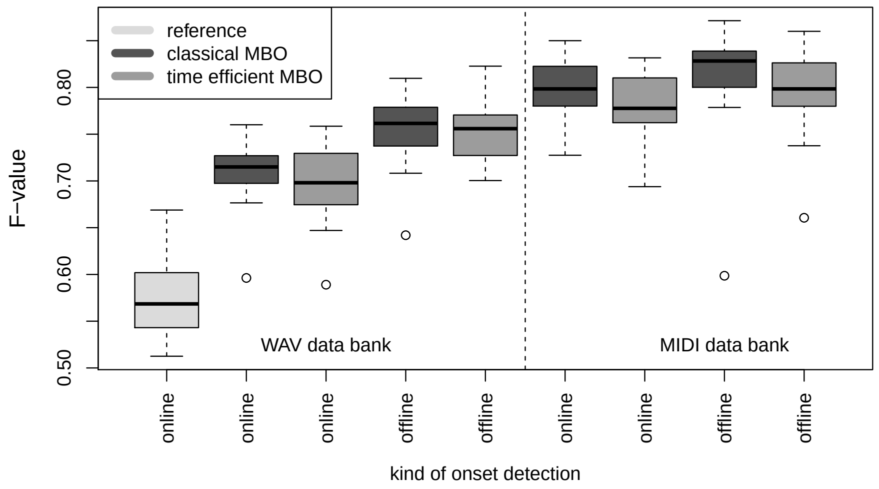

Figure 4: Validated F-values of the classical and proposed MBO applied to the online and the offline OD on the WAV and the MIDI data sets. Surrogate model: naive Kriging.

to be more remarkable on the WAV data base. The p-value of the Wilcoxon signed rank test for online vs. offline OD performance comparison on the WAV data set (for naive Kriging model) is $1.94 \cdot 10^{-08}$ and on the MIDI data sets 0.007. The observed difference is also significant and can be explained due to additional information about the future signal behavior.

Further on, Figure 3 reveals that noticeable better F-values can be observed on the MIDI data set compared to the WAV data (see application question 2, **AQ2**) This difference is highly significant according to the Wilcoxon test both for online and offline OD (p-values are $1.33 \cdot 10^{-15}$ and $9.54 \cdot 10^{-10}$, respectively). A possible explanation could be the artificially regular patterns of tone beginnings in the synthetically generated recordings. Such pieces can not reproduce many musical aspects like development of overtones or legato (smoothly playing of many tones) so that the time points of tone onsets can be easier identified compared to real pieces.

A more difficult problem is how to compare classical MBO with the proposed time efficient instance based approach (optimization question 2, **OQ2**). The proposed MBO can be seen as an approximation of the classical MBO which, on the one hand, is probably a bit worse but, on the other hand, much faster. There are two possible points of view to compare the two strategies. One possibility is based on the same number of iterations and other one is based on the same number of instance evaluations. Which approach is better depends on the run-time costs of the instance evaluations versus the run-time costs of fitting the Kriging model. Taking the above time analysis into account, the more sophisticated comparison seems to be based on the number of instance evaluations. However, to achieve a better generalization to other optimization tasks, we will consider both points of views.

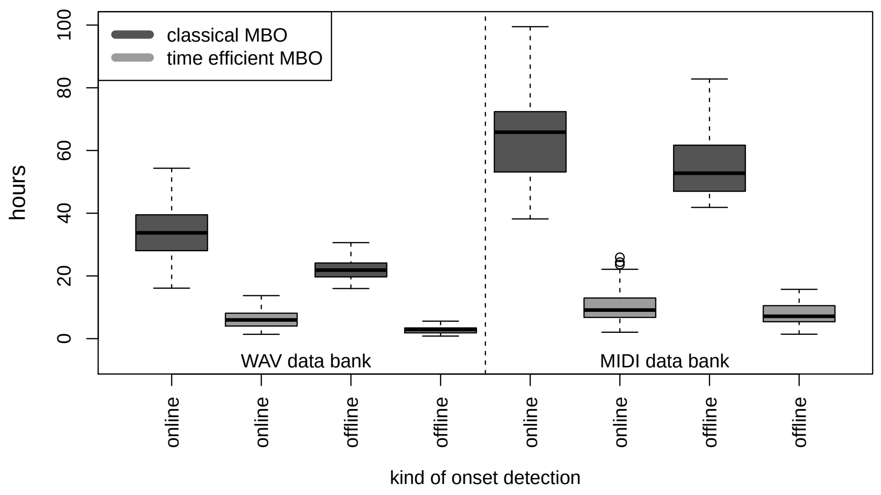

Figure 5: Time (in hours) for function evaluations in sequential steps for the considered optimization stretgies on the WAV (left) and MIDI (right) data base.

Figure 4 illustrates the distribution of the validated F-values for both optimization strategies in case of the same number of iterations (for online and offline OD on WAV and MIDI data sets). Moreover, a simple time efficient reference strategy is applied for online OD on the WAV data base: after complete evaluation of the initial design, $k_{pretest}$ music pieces were chosen randomly so that the optimization in the sequential steps was done just on these instances. As can be seen, the reference strategy shows the worst results. It can be stated for all pairwise comparisons that the proposed instance based approach is slightly or noticeably worse than the classical MBO. The observed difference is significant only on the MIDI data set with p-values of 0.037 for the online OD and 0.042 for the offline OD.

Figure 5 shows the time efficiency of the proposed variant compared to classical MBO. Here, the total computational time in hours on the used computer cluster system for function evaluations in sequential steps is presented over 30 replications of the associated optimization approaches. The mean time saving amounts approx. 23.5 hours on the WAV data set and approx. 51.3 hours on the MIDI data set. This corresponds in both cases to the astonishingly high mean time saving of 84.3%. This saving depends mainly on the number of “good” points, where the complete data set should be evaluated: the more such points, the lower is the saving. However, also the size of the pretest subset influences

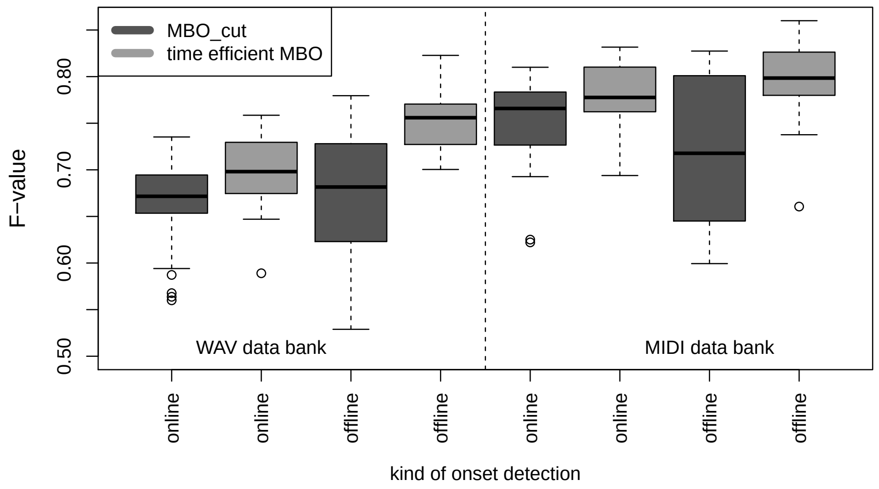

Figure 6: F-values of the proposed time efficitent optimization compared to MBO_cut for the online and the offline OD on the WAV and the MIDI data bases.

the function evaluation time. To remain, in line 6 of Algorithm 2 we proposed a strategy for reducing the subset size. As the result for this application, the number of selected problem instances was reduced from 5% to 3.4% of the training data set on average.

After we compared both MBO strategies on the same number of MBO iterations (Figure 4), let us now compare them based on the same number of music piece evaluations. For this reason, the number of instance evaluations made by an application of the time efficient MBO is divided by the size of the training data set and rounded up. In this manner, the number of equivalent complete data set evaluations is calculated. Afterwards, the optimization path of the associated classical MBO run (in the same validation step and with the same initial design) is cut to this number of iterations and its best already achieved F-value is saved. This approach is abbreviated by MBO_cut.

The comparison of the proposed MBO and MBO_cut in Figure 6 shows a substantial superiority of the proposed approach for both data sets (especially for offline OD). These superiority is highly significant according to the Wilcoxon tests. Although the number of music piece evaluations is nearly the same, the computational time for the instance based optimization might be slightly larger due to modeling overhead. This overhead can be ignored if the computational time for instance evaluations is noticeable larger than the $M'_{pretest}$ and M_{sur} model fitting time. This is the case for our application, as already illustrated in Table 1.

7. Conclusion

The main idea of the proposed time efficient variant of MBO for instance based problems is choosing a representative set of problem instances which can accurately estimate the overall performance. When allowing the same number of instance evaluations, this approach performs significantly better than classical MBO – according to the Wilcoxon signed rank test – for online and offline onset detection. However, this comparison assumes that time for instance evaluations is the only important one while time for surrogate modeling and infill criterion optimization can be neglected. Note, under this condition, the time efficient strategy needs more MBO iterations. For the onset detection application on a large music data base this assumption is fulfilled.

Nevertheless, we also compare both MBO variants concerning the same number of iterations to get a more generalized result also for other applications. Then, classical MBO is slightly better while requiring approximately six times more computational time for instance evaluations. To conclude, the more instances the instance based problem contains and the more time expansive they are, the more preferable the proposed MBO approach becomes.

Additionally, since classical MBO with the popular Kriging surrogate model is limited only to continuous parameters, two extensions for categorical parameters are compared: (1) naively handling them as continuous variables and (2) building dummy variables. No significant difference in the optimization performance can be stated between both extensions while the optimization runs with the naive Kriging model require on average 3.5 hours fewer computational time than with the dummy Kriging model. However, to get a more general statement also for other applications with categorical parameters, further simulation studies are important to investigate the naive Kriging performance.

Other interesting results with respect to the optimization problem are, firstly, the significantly better performance of offline onset detection (due to the using of the future signal information) compared to the online variant. Secondly, synthetically generated recordings (MIDI data) show significantly better results than the real music pieces (WAV data). This might be caused by inevitable simplifications of the sound properties made by the artificial signal generation.

In future work, we aim, on the one hand, to improve the proposed instance based MBO since there are many sensible alternatives both for the selection of the pretest instances and for the choice of the pretest model. For example, also the application of a classification method to select "good" and "bad" points might be sensible. On the other hand, other techniques for optimizing application problems with categorical parameters should be compared to the proposed ones. For instance, in [6] a random forest surrogate model and another definition of model uncertainty are used for handling this problem. But also the usual covariance kernels can be applied by utilizing an appropriate distance measure (like the gover metric, [36]). Finally, we are interested to test the time efficient MBO on further instance based problems.

Acknowledgment

This paper is based on investigations of the projects C2 and B3 of SFB 823, which are kindly supported by the German Research Foundation (DFG).

References

[1] C. Weihs, J. Jessenberger, Statistische Methoden zur Qualitätssicherung und -optimierung in der Industrie, Wiley-VCH, 1999.

[2] J. C. Spall, Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control, Wiley, 2003.

[3] S. Herbrandt, C. Weihs, U. Ligges, M. Ferreira, C. Rautert, D. Biermann, W. Tillmann, Optimization of a simulation for inhomogeneous mineral subsoil machining, in: Proceedings of the European Conference on Data Analysis, Springer International Publishing, 2015.

[4] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global Optimization 13 (4) (1998) 455–492.

[5] N. Bauer, K. Friedrichs, B. Bischl, C. Weihs, Fast model based optimization of tone onset detection by instance sampling, in: A. Wilhelm, H. A. Adalbert (Eds.), Analysis of Large and Complex Data, Springer International Publishing, 2015, p. to appear.

[6] F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: C. A. C. Coello (Ed.), Learning and Intelligent Optimization, Vol. 6683 of Lecture Notes in Computer Science, Springer, 2011, pp. 507–523.

[7] J. P. Bello, L. Daudet, S. A. Abdallah, C. Duxbury, M. E. Davies, M. B. Sandler, A tutorial on onset detection in music signals, IEEE Transactions on Speech and Audio Processing 13 (5-2) (2005) 1035–1047.

[8] S. Dixon, Onset detection revisited, in: Proceedings of the 9th International Conference on Digital Audio Effects (DAFx'09), 2006, pp. 133–137.

[9] A. Holzapfel, Y. Stylianou, A. Gedik, B. Bozkurt, Three dimensions of pitched instrument onset detection, IEEE Transactions on Audio, Speech, and Language Ppocessing 18 (6) (2010) 1517–1527.

[10] C. Rosão, R. Ribeiro, D. Martins De Matos, Influence of peak selection methods on onset detection, in: Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR'12), 2012, pp. 517–522.

[11] V. Picheny, D. Ginsbourger, Noisy kriging-based optimization methods: A unified implementation within the diceoptim package, Computational Statistics & Data Analysis 71 (C) (2014) 1035–1053.

[12] V. Picheny, T. Wagner, D. Ginsbourger, A benchmark of kriging-based infill criteria for noisy optimization, Structural and Multidisciplinary Optimization 48 (3) (2013) 607–626.

[13] N. Bauer, J. Schiffner, C. Weihs, Comparison of classical and sequential design of experiments in note onset detection, in: B. Lausen, D. V. den Poel, A. Ultsch (Eds.), Data Analysis, Machine Learning and Knowledge Discovery, Springer International Publishing, 2013, pp. 501–509.

[14] N. Bauer, J. Schiffner, C. Weihs, Comparison of parameter optimization techniques for a music tone onset detection algorithm, in: Proceedings of the 4th Meeting on Statistics and Data Mining (MSDM), 2013, pp. 28–34.
URL `http://www.tasa-online.com/`

[15] M. D. McKay, R. J. Beckman, W. J. Conover, Comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 21 (2) (1979) 55–61.

[16] D. G. Krige, A statistical approach to some basic mine valuation problems on the witwatersrand, Metallurgical and Mining Society of South Africa 52 (6) (1951) 119–139.

[17] M. Schonlau, Computer experiments and global optimization, Ph.D. thesis, University of Waterloo (1997).

[18] C. E. Rasmussen, C. K. I. Williams, Gaussian processes for machine learning, The MIT Press, 2006.

[19] D. R. Jones, A taxonomy of global optimization methods based on response surfaces, Journal of Global Optimization 21 (4) (2001) 345–383.

[20] B. Bischl, J. Bossek, D. Horn, M. Lang, mlrMBO: Model-Based Optimization for mlr, R package version 1.0 (2015).
URL `https://github.com/berndbischl/mlrMBO`

[21] S. Böck, F. Krebs, M. Schedl, Evaluating the online capabilities of onset detection methods, in: Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR'12), 2012.

[22] J. Cooley, J. Tukey, An algorithm for the machine calculation of complex fourier series, Mathematics of Computation 19 (90) (1965) 297–301.

[23] F. J. Harris, On the use of windows for harmonic analysis with the discrete fourier transform, IEEE 66 (1) (1978) 51–83.

[24] E. Benetos, S. Dixon, Polyphonic music transcription using note onset and offset detection, in: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2011, pp. 37–40.

[25] N. Bauer, J. Schiffner, C. Weihs, Einfluss der Musikinstrumente auf die Güte der Einsatzzeiterkennung, Tech. Rep. 10/12 (2012).

[26] P. Masri, Computer modeling of sound for transformation and synthesis of musical signal, Ph.D. thesis, University of Bristol (1996).

[27] G. Peeters, X. Rodet, A large set of audio feature for sound description (similarity and classification) in the cuidado project, Tech. rep., Ircam (2004).

[28] N. Bauer, K. Friedrichs, D. Kirchhoff, J. Schiffner, C. Weihs, Tone onset detection using an auditory model, in: M. Spiliopoulou, L. Schmidt-Thieme, R. Janning (Eds.), Data Analysis, Machine Learning and Knowledge Discovery, Springer International Publishing, 2014, pp. 315–324.

[29] F. Eyben, S. Böck, B. Schuller, A. Graves, Universal onset detection with bidirectional long short-term memory neural networks, in: J. S. Downie, R. C. Veltkamp (Eds.), Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR'10), International Society for Music Information Retrieval, 2010, pp. 589–594.

[30] J. A. Hartigan, M. A. Wong, Algorithm as 136: A k-means clustering algorithm, Journal of the Royal Statistical Society, Series C (Applied Statistics) 28 (1979) 100–108.

[31] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer New York Inc., New York, 2001.

[32] Y. Bengio, Y. Grandvalet, No unbiased estimator of the variance of k-fold cross-validation, Journal of Machine Learning Research 5 (2004) 1089–1105.

[33] S. Siegel, N. Castellan, Nonparametric statistics for the behavioral sciences, 2nd Edition, McGraw–Hill, Inc., 1988.

[34] B. Bischl, M. Lang, O. Mersmann, BatchExperiments: Statistical experiments on batch computing clusters., R package version 1.2 (2014).
URL `http://CRAN.R-project.org/package=BatchExperiments`

[35] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2014).
URL `http://www.R-project.org/`

[36] J. C. Gower, A general coefficient of similarity and some of its properties, Biometrics 26 (1971) 857–874.