DATA SCIENCE@LMU
STATISTICS + INFORMATICS

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

ESG Data Science

Noisy Bayesian Optimization for Model Selection in Machine Learning

Master's Thesis

Julia Moosbauer

Supervisors:	Prof. Dr. Bernd Bischl
	Janek Thomas
Submission Date:	September 27, 2018

I declare that I have developed and written the enclosed Master's Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Master's Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

Munich, September 27, 2018

Julia Moosbauer

Abstract

Expensive black-box problems occur commonly in practice. In many cases, the underlying function cannot be accessed directly but only a noisy version thereof. One prominent example is model selection in machine learning: performance estimates for model configurations are noisy as they depend on the data that is used for training and validation. The variance in the performance estimates can be reduced if more budget is spent on the evaluation of hyperparameter settings in terms of repetition or cross-validation folds.

This raises two questions: (1) Can intelligent evaluation budget allocation strategies improve the overall optimization result? (2) Can we infer statistical guarantees or even maximize the confidence on the returned solution?

This thesis investigates in replication strategies as extension of sequential model-based optimization for noisy, expensive black-boxes. The contribution of this thesis is threefold: First, existing replication strategies from literature are compared in a benchmark study on artificial test functions and on machine learning tuning problems. Second, a new version of sequential model-based optimization, that aims at reaching a pre-set level of confidence in the returned solution through the implementation of an identification step, is proposed. Third, the new proposed method is applied on the traffic simulation optimization problem CrowdNav[10].

We empirically demonstrate that sophisticated replication strategies can improve the overall optimization result in specific situations, especially for higher dimensional input spaces and high noise levels.

Contents

Table 0.1: Notation

$\mathcal{X} \subset \mathbb{R}^p$	input space of dimensionality p
$\boldsymbol{x} \in \mathcal{X}$	input value, configuration, parameter, design point
$\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\right\} \subset \mathcal{X}^n$	n unique input values
n	number of unique design locations
$\mathcal{Y} \subset \mathbb{R}$	output space
$f : \mathcal{X} \to \mathcal{Y}$	black-box function
$y(\boldsymbol{x}) \in \mathcal{Y}$	(potentially noisy) evaluation of f
$\epsilon(\boldsymbol{x}) \sim \mathcal{N}\left(0, \tau^2(\boldsymbol{x})\right)$	zero-mean Gaussian noise with variance $\tau^2(x)$
$\tau^2(\boldsymbol{x})$	noise variance function
$r(\boldsymbol{x}^{(i)})$, $r^{(i)}$	number of function evaluations at input $\boldsymbol{x}^{(i)} \in \mathcal{X}$
$\boldsymbol{y}^{(i)} := \left(y^{(i,1)}, ..., y^{(i,r^{(i)})}\right)$	$r^{(i)}$ realizations of $y(\boldsymbol{x}^{(i)})$ (replications)
$r := \sum_{i=1}^{n} r^{(i)}$	total number of function evaluations
$D := \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i \in 1,\ldots,r}$	design containing duplicated inputs[1]
$\bar{D} := \left\{\left(\boldsymbol{x}^{(i)}, \bar{y}^{(i)}\right)\right\}_{i \in 1,\ldots,n}$	mean-aggregated design[2]
$\mathcal{H}$	model hypothesis space
$\hat{f}(\boldsymbol{x}) \in \mathcal{H}$	estimated surrogate model for f
$\hat{s}(\boldsymbol{x})$	estimated variance of $\hat{f}(x)$
$\mathcal{I} : \mathcal{X} \to \mathbb{R}$	infill criterion
$\phi : \mathbb{R} \to \mathbb{R}^+, \Phi : \mathbb{R} \to [0, 1]$	standard normal density and distribution function

Without loss of generality objective functions are minimized throughout this work.

[1] Note that $D = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i \in 1,\ldots,r}$ shall be understood as an indexed family containing all evaluations. In case of replication the same input can appear in multiple tuples in D.

[2] The mean-aggregated dataset is denoted by $\bar{D}$. Here, each input appears only once. We denote $t(D)$ the summarized/transformed design. Throughout this work, we consider $t(D)$ be either the identity, i. e. $t(D) = D$, or mean-aggregation, i. e. $t(D) = \bar{D}$.

Chapter 1

Introduction

Medical doctors decide how to dose a drug and measure its effect by employing medical scores or subjective assessment of the patient. The development of robots or electronic devices depends on diverse configuration decisions that impact their functioning. The development of a car requires a huge range of design decisions which possibly impacts safety. This can be assessed through expensive crash tests. In computer science, algorithms are often complex and highly parameterized, and their performance largely depends on the initial configuration of some hyperparameters.

All those examples have three things in common: the relationship between input and output is a sort of a *black-box*, the system is *expensive* to evaluate and often *stochastic*.

For a perfect functioning, all those problems require the right design choice out of a high-dimensional and complex decision space. The relationship between design decision and outcome is often unknown or cannot be described analytically: the underlying mechanism is a *black-box*.

Most of the experiments are *expensive* to evaluate in terms of time, monetary or non-monetary resources: the execution time of an algorithm, the costs for a crash test or the inconveniences for a patient when medication is adjusted, just to name some examples.

In addition, experiments are often *stochastic*: in medicine, there are random factors like the form on the day of the patient, that cannot be controlled. Even computer programs and algorithms are often stochastic.

Automating these design choices has a tremendous impact on a variety of application sectors. Sharari et al. [24] state that, “any significant advances in automated design can result in [...] innovation in a wide area of domains, including advertising, health-care informatics, banking, information mining, life sciences, control engineering, computing systems, manufacturing, e-commerce, and entertainment”.

Yet another field of application has the character of a *stochastic, expensive black-box* problem: model selection in machine learning. Many machine learning algorithms require careful tuning of model hyperparameters. Without automatic approaches for selection of optimal hyperparameters, tuning requires expert experience, rules of thumb

or often brute force search. Through rapid progress in the area of machine learning in recent years, the problem of hyperparameter optimization for machine learning methods has gained importance.

The high complexity of machine learning models makes them often a *black-box* to us. The underlying unknown relationship between hyperparameters of a machine learning model and the output is investigated through training of the model on a training set and subsequent evaluation via an appropriate performance measure.

In accordance with Snoek et al. [25] hyperparameter optimization has a somewhat different flavor than the low-level objectives one often encounters as part of a training procedure: function evaluations are very *expensive* as they involve running the primary machine learning algorithm to completion. For a valid evaluation, advanced validation techniques like cross-validation even require multiple training runs.

There are multiple factors that make the system *stochastic*: The model itself can be of stochastic nature: random forests [6], for example, incorporate bootstrapping in their training process. Loss minimizers, for example stochastic gradient descent, can be stochastic as well. Most importantly, noise is introduced by the data that is used for training and validation. Common model validation techniques like holdout or cross-validation randomly split the data in one or multiple training and test folds. The variance of the estimated performance of a configuration, however, decreases when putting more budget on validation: in general, 10-fold cross-validation will give more reliable results than a 3-fold cross-validation will do.

Sequential model-based optimization [14], which iterates between fitting models and using them to choose which configurations to evaluate next, has become the state-of-the-art optimization strategy for expensive black-box problems.

Many noisy black-box problems like model selection in machine learning naturally raise the question of how much budget needs to be spent on function evaluations. For instance, performing ten times repeated holdout at an inferior configuration can be a waste of budget. Performing a single holdout at a good configuration can yield an unlucky false assessment of the performance which might impact optimization.

Within this work, we investigate the question of evaluation budget allocation in sequential model-based optimization for noisy black-box problems. Chapter 2 will give a short review on sequential model-based optimization emphasizing the problems that arise in noisy settings. In chapter 3 and 4, noise handling replication strategies that are integrated into the Bayesian optimization procedure are presented and evaluated in a benchmark on both synthetic test functions and more realistic machine learning problems. Finally, the developed methods will be employed in a use case from the area of simulation optimization.

Chapter 2

Background and Challenges

2.1 Bayesian Optimization in Noise-free Systems

Problem Statement

In many practical situations we aim at optimizing a system for which an algebraic model and thus especially its derivate is not available. For $\mathcal{X} \subseteq \mathbb{R}^p$ and $\mathcal{Y} \subseteq \mathbb{R}$, we denote $f : \mathcal{X} \to \mathcal{Y}$ the *unknown* function that describes the relationship between an input $x \in \mathcal{X}$ and a system output $y \in \mathcal{Y}$ in terms of some quality criterion. Such systems are referred to as *black-box systems*.

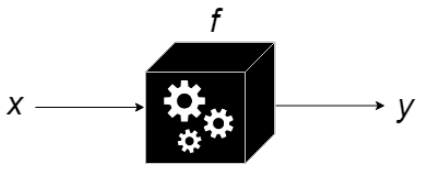


Figure 2.1: A black-box system.

Whereas the analytical form of f is not known, the black-box system can be queried for any value x. Such an evaluation is denoted by $y(\boldsymbol{x})$. If function evaluations are expensive in terms of time or other resources, the number of function evaluations is often limited.

In the noise-free setting, the evaluated value $y(\boldsymbol{x})$ at any location $\boldsymbol{x}$ corresponds to the true function value $f(\boldsymbol{x})$ as exemplarily shown in Figure 2.2.

Solution Approaches

The state-of-the-art approach of minimizing expensive black-box functions is *sequential model-based optimization* (SMBO), also known as *Bayesian optimization*.

The motivation behind is the following: Taking a Bayesian perspective, the unknown function $f(\boldsymbol{x})$ is regarded random and prior believes about its shape are formulated in terms of a prior distribution. First, initial information about the function is gathered for different inputs $\boldsymbol{x}^{(i)}$ through evaluation of the black-box function f, i. e. $y^{(i)} = f(\boldsymbol{x}^{(i)})$,

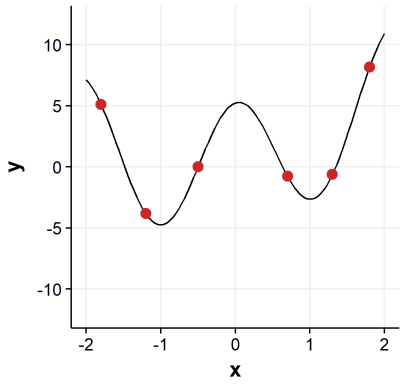


Figure 2.2: The underlying unknown function $f(x)$ is black, the observed values $y(x)$ are red. In the noise-free case the observations correspond to the true function values.

which results in an *initial design* $D = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i\in 1,\ldots,n_{\text{init}}}$. Then, the prior is updated to form the posterior distribution of $f(\boldsymbol{x})$ given D. The posterior distribution, in turn, is used to formulate a criterion $\mathcal{I}(\boldsymbol{x})$, referred to as *infill* or *acquisition criterion*, that is used to determine which points to evaluate next. Iteratively performing these two steps results in a sequential optimization procedure.

Indisputably, the most prominent representative of sequential model-based optimization still yielding state-of-the-art performance is the *Efficient Global Optimization* (EGO) algorithm by Jones et al. [14].

First, the initial design D is determined and evaluated.

Then the optimization process is started: in each iteration, the posterior of f given D is determined. According to the above described Bayesian motivation, the EGO algorithm is based on the assumption of a Gaussian process prior on $f(\boldsymbol{x})$. In its simplest form, a zero-mean Gaussian process prior is placed on $f(\boldsymbol{x})$.

After that, the *expected improvement (ei)* with respect to the current best observed function value $y^{(min)} := \min_{\boldsymbol{x}\in D}\{f(\boldsymbol{x})\}$ is used to propose the most promising point[1]. In combination with the Gaussian process assumption, the expected improvement is analytically tractable and the posterior distribution enters in term of the posterior mean $\hat{f}(\boldsymbol{x})$ and the posterior variance $\hat{s}(\boldsymbol{x})$

[1] In general, the $m \geq 1$ most promising points can be proposed here. Throughout this work the number m is set to 1.

$$\begin{aligned}\mathcal{I}_{ei}(\boldsymbol{x}) &= \mathbb{E}\left[\left(y^{(min)} - \hat{f}(\boldsymbol{x})\right)^{+}\right] \\ &= \left(y^{(min)} - \hat{f}(\boldsymbol{x})\right) + \Phi\left(\frac{y^{(min)} - \hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right) + \hat{s}(\boldsymbol{x})\phi\left(\frac{y^{(min)} - \hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right).\end{aligned} \tag{2.1}$$

As the evaluation of the infill criterion $\mathcal{I}(\boldsymbol{x})$ is usually inexpensive compared to the evaluation of the black-box function, a large number of evaluations of $\mathcal{I}(\boldsymbol{x})$ can be performed and simpler derivative-free optimization methods can be applied.

Finally, the design point with the minimum function value, i. e. $\boldsymbol{x}^*$ with $f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}\in D}\{f(\boldsymbol{x})\}$, is returned.

Note that the Gaussian process regression [21], also referred to as *Kriging*, can be replaced by other regression models. Though slightly moving away from the original Bayesian motivation, the posterior mean $\hat{f}(\boldsymbol{x})$ can be seen as a model approximating the unknown black-box function f and is commonly referred to as *surrogate model* $\hat{f}$. For numeric input spaces $\mathcal{X} \subset \mathbb{R}^p$, Kriging is a recommended choice due to its capability of modelling spatial structures. However, Gaussian processes are not applicable to problems where the input space is categorical or mixed. Hutter et al. [12] replace Gaussian process models by random forest regression models, which are able to handle categorical input variables and also yield variance estimates through out-of-bag estimates.

Whilst the expected improvement is a natural choice for an infill criterion, other infill criteria, which differ in the way they balance exploitation and exploration, can also be used. Usually, the way the posterior mean $\hat{f}(x)$ and the posterior variance $\hat{s}(x)$ are combined defines that balance in an infill criterion. Besides the expected improvement, another common choice to combine $\hat{f}(\boldsymbol{x})$ and $\hat{s}(\boldsymbol{x})$ is the *lower confidence bound* (cb)

$$\mathcal{I}_{cb}(\boldsymbol{x}, \lambda) = \hat{f}(\boldsymbol{x}) + \lambda\hat{s}(\boldsymbol{x}), \quad \lambda > 0. \tag{2.2}$$

To force the optimizer to pure exploitation, the *mean response (mr)* criterion $\mathcal{I}_{mr}(\boldsymbol{x}) = \hat{f}(\boldsymbol{x})$ can be used. The *standard deviation (sd)* criterion $\mathcal{I}_{sd}(\boldsymbol{x}) = \hat{s}(\boldsymbol{x})$ leads to pure exploration.

A general scheme for SMBO is outlined in Algorithm 1.

Algorithm 1 Sequential model-based optimization

1: **input:** infill criterion $\mathcal{I}$, hypothesis space $\mathcal{H}$

2: generate initial design D
3: **while** termination criterion not met **do**
4: **fit** surrogate $\hat{f} \in \mathcal{H}$ on the design data D
5: **propose** the point that optimizes $\mathcal{I}(\boldsymbol{x})$ and update D
6: **end while**

2.2 Bayesian Optimization in Noisy Systems

Problem Statement

In many real-life situations, we cannot access the true function values $f(\boldsymbol{x})$ but only a noisy version thereof

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon(\boldsymbol{x}). \tag{2.3}$$

Here, $\epsilon(\boldsymbol{x})$ is a random variable that represents the noise. For the sake of simplicity, the noise is assumed to be Gaussian throughout this work, i. e.

$$\epsilon(\boldsymbol{x}) \sim \mathcal{N}\left(0, \tau^2(\boldsymbol{x})\right). \tag{2.4}$$

If the noise variance $\tau^2(\boldsymbol{x})$ is constant, i. e. $\tau^2(\boldsymbol{x}) \equiv \tau^2 \in \mathbb{R}_{>0}$, noise is called homoscedastic, otherwise it is called heteroscedastic.

Figure 2.3 shows an example of a function which is only accessible via noisy function evaluations.

Challenges induced by noise

The existence of noise causes misleading information about the underlying function and introduces two challenges that need to be addressed by an optimization procedure:

1. Misleading information affects the quality of the fitted surrogate model and the proposed points which might have a negative impact on the *overall* optimization result

2. Noisy evaluations lead to false assessment of design points and the final best point might not be identifiable in the end (*identification error*)

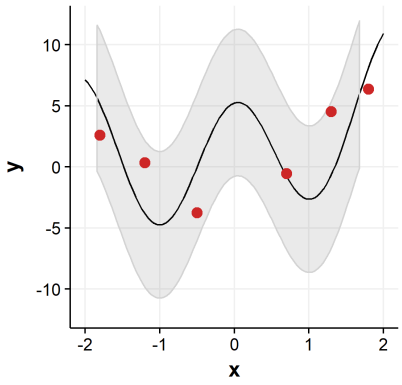


Figure 2.3: The underlying unknown function $f(x)$ is black, the observed values are red. The grey area is the function value plus-minus two times the standard deviation $f(x) \pm 2 \cdot \tau(\boldsymbol{x})$ with $\tau(\boldsymbol{x}) \equiv 3$ (homoscedastic noise).

Those challenges can be emphasized by reconsidering EGO. As Kriging [21] is a strictly interpolating approach, it will return the observed value $\hat{f}(\boldsymbol{x}^{(i)}) = y^{(i)}$ for each design input $\boldsymbol{x}^{(i)} \in D$ which might deviate widely from the true function value.

In addition to that, despite its practical applicability, the choice of the expected improvement criterion is questionable from a theoretical perspective: The expected improvement criterion depends on $y^{(min)}$, which is not available in the noisy case.

Finally, the best configuration in design is not necessarily the point with the best observed output. Bad points could have been overrated while good points could be overlooked through single unlucky evaluations.

Solution Approaches

Two independent lines of work have extended the EGO algorithm to the noisy case: the *Sequential Kriging Optimization* (SKO) algorithm by Huang et al. [11] and the *Sequential Parameter Optimization* (SPO) algorithm by Bartz-Beielstein et al. [1].

The SKO algorithm is an instance of Algorithm 1, but differs from the EGO algorithm in its implementation of the Gaussian process meta-model and its choice of the infill criterion. To accommodate noise, a noisy Gaussian process model, also called nugget-effect Kriging model [21], is used as a surrogate model.

Furthermore, the *augmented expected improvement (aei)*

$$\begin{aligned}\mathcal{I}_{aei}(\boldsymbol{x}) = (T - \hat{f}(\boldsymbol{x}))\Phi\left(\frac{T - \hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right) \\ + \hat{s}(\boldsymbol{x})\phi\left(\frac{T - \hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right) \times \left(1 - \frac{\tau}{\sqrt{\hat{s}^2(\boldsymbol{x}) + \tau^2}}\right)\end{aligned} \tag{2.5}$$

is used instead of the expected improvement criterion. Here, the observed minimum $y^{(min)}$ in the *ei* criterion is replaced by $T = \hat{f}(\boldsymbol{x}^{**})$, which is defined as the predicted value at the effective best solution $\boldsymbol{x}^{**} = \arg\min_{\boldsymbol{x}\in D} \hat{f}(\boldsymbol{x}) + \hat{s}(\boldsymbol{x})$. The second term in Equation 2.5 is a penalty term that increases towards 1 if the function is noise-free. The effective best solution $\boldsymbol{x}^{**}$ is the configuration returned at the end of the optimization process.

The literature contains further attempts to adapt infill criteria for noisy function evaluations. Picheny et al. [19] give an overview over different criteria. One example the so-called *expected quantile improvement (eqi)*

$$\mathcal{I}_{eqi}(\boldsymbol{x}) = (q^{(min)} - \hat{f}_Q(\boldsymbol{x}))\Phi\left(\frac{q^{(min)} - \hat{f}_Q(\boldsymbol{x})}{\hat{s}_Q(\boldsymbol{x})}\right) + \hat{s}_Q(\boldsymbol{x})\phi\left(\frac{q^{(min)} - \hat{f}_Q(\boldsymbol{x})}{\hat{s}_Q(\boldsymbol{x})}\right), \tag{2.6}$$

where $q^{(min)} := \min_{\boldsymbol{x}\in D}\{q(\boldsymbol{x})\}$ with $q(\boldsymbol{x})$ denoting the Kriging quantile $\hat{f}(\boldsymbol{x}) + \Phi(\beta)\hat{s}(\boldsymbol{x})$. $\hat{f}_Q(\boldsymbol{x})$ and $\hat{s}_Q(\boldsymbol{x})$ denote the mean and standard deviation of the Kriging quantile Q updated with $\boldsymbol{x}$ [19].

The SPO method, in contrast, is based on repeated evaluations of design points.

The algorithm proceeds as follows: Each point of the initial design D is evaluated b times. Before entering the sequential optimization loop, the empirical mean function value for each input $\boldsymbol{x}$ (or, more generally, the empirical estimates of a user-defined cost metric) is calculated. The *incumbent* (see Definition 2.1) is determined.

Definition 2.1 (Incumbent)
For a given set of design points $D = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i\in 1,\ldots,r}$, the incumbent is defined as the design input $\boldsymbol{x}^{(inc)}$ with the minimal empirical function value

$$\bar{y}^{(inc)} = \min_{i=1,\ldots,n} \bar{y}^{(i)}.$$

Then, an interpolating Gaussian process model is fitted to the mean-aggregated data $\bar{D} := \left\{\left(\boldsymbol{x}^{(i)}, \bar{y}^{(i)}\right)\right\}$. As infill criterion, the classical expected improvement criterion is used. An *intensification strategy* is explicitly forcing replication: In every iteration, not only the new selected point but also the incumbent is evaluated b times. If the

incumbent is not replaced by a new point, the number of iterations b performed on every (subsequent) point is doubled. Hence, SPO sequentially increases the number of replications.

The main difference between the two approaches is the fact that the SPO procedure explicitly incorporates repeated function evaluations while the SKO approach does not[2].

Hutter et al. [13] experimentally investigate these two sequential model-based procedures in the context of performance optimization of randomized algorithms and found, that the SPO algorithm offered more robust performance than the SKO algorithm. But the implementation of an explicit replication strategy has another major advantage: replication in design points allows inferring statistical guarantees about the performance of different configurations and reduces the risk of "mistakenly" returning overestimated configurations (*identification error*).

Inspired by the idea of an intensification mechanism we formulate the general SMBO WITH INTENSIFICATION in Algorithm 2, which extends Algorithm 1 by integrating an explicit replication strategy[3], that controls the degree to which existing design points are evaluated.

Algorithm 2 Sequential model-based optimization with intensification

1: **input:** infill criterion $\mathcal{I}$, hypothesis space $\mathcal{H}$, replication strategy $\mathcal{R}$

2: generate initial design D
3: **while** termination criterion not met **do**
4: **fit** surrogate $\hat{f} \in \mathcal{H}$ on the (aggregated) design data $t(D)$
5: **propose** the point that optimizes $\mathcal{I}(\boldsymbol{x})$ and update D
6: **intensify** existing design points through a replication strategy $\mathcal{R}$
7: **end while**

We explore sequential model-based optimization procedures that incorporate intensification strategies. The thesis aims to investigate whether proposed replication strategies help managing the uncertainty in the presence of noise and improve the confidence in solutions. Instances of Algorithm 2 are analysed with respect to those replication strategies and other important algorithmic factors like the surrogate model and the infill criterion.

[2]Note that in contrast to the *ei* criterion, the *aei* criterion is in general not 0 for already evaluated points $\boldsymbol{x} \in D$. However, for (theoretically) continuous functions, a point is re-evaluated with zero probability.

[3]The terms replication strategy and intensification strategy will be used synonymously.

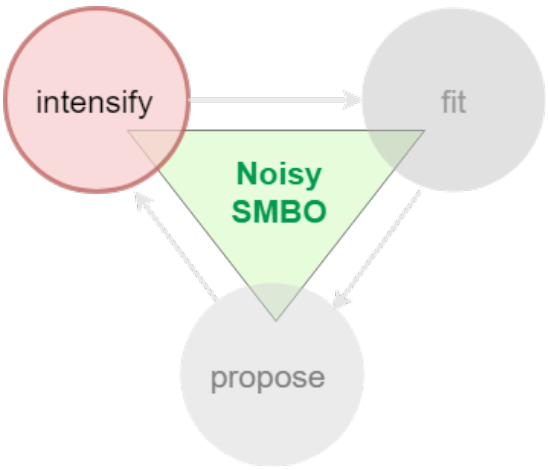


Figure 2.4: Sequential model-based optimization iterates between fitting a surrogate learner on the design data, proposing new promising points and intensifying existing design points.

Chapter 3

Methods

Our primary goal is to find an optimal replication strategy that manages uncertainty during the optimization process and improves confidence in the final solution.

In this section, different intensification strategies, which can be used along with Algorithm 2, are presented. These strategies can be classified into three groups: (1) approaches requiring the specification of a replication budget, (2) approaches not requiring explicit budget specifications, and (3) approaches where the user can prescribe a minimum confidence. The replication decisions are internally determined by the algorithm to match the required level of confidence.

3.1 Replication Strategies with Budget Specification

Fixed Replication Budget

A naive way of integrating replication into SMBO is *fixed* re-evaluation of each point: every point that is proposed by infill optimization is evaluated $b > 1$ times. This replication strategy is sketched in Algorithm 3.

As illustrated in Figure 3.1, the same replication effort is spent on points independent of both their potential of being optimal and the corresponding noise level.

Algorithm 3 *fixed* replication strategy

input: design D, $\boldsymbol{x}^{(prop)}$ proposed input, replication budget b

$j = 0$
while $j < b$ **do**
    $y^{(prop,j)} \leftarrow$ realization of $y(\boldsymbol{x}^{(prop)})$
    $D \leftarrow D \cup \left\{\left(\boldsymbol{x}^{(prop)}, y^{(prop,j)}\right)\right\}$
    $j \leftarrow j + 1$
end while

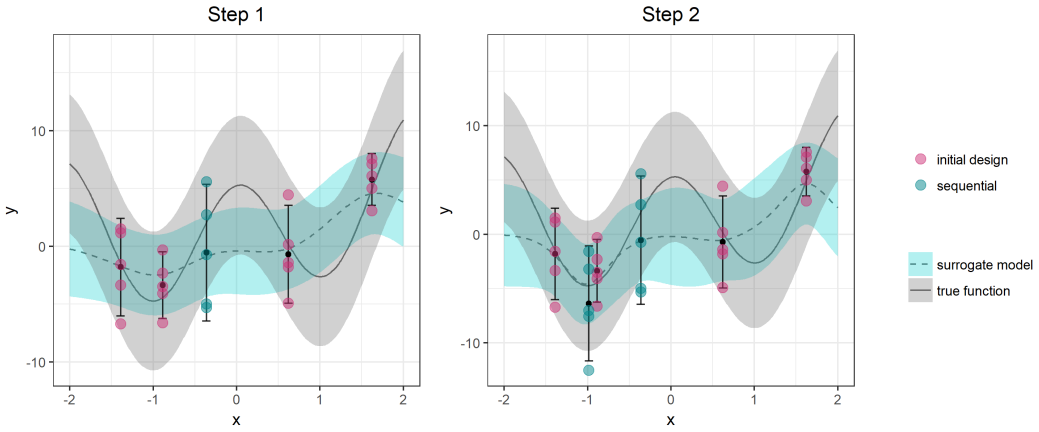


Figure 3.1: Two steps of SMBO with *fixed* intensification starting from an initial Latin hypercube design of size 4 (violet points). Each point is evaluated $b = 5$ times. The surrogate learner is a noisy Gaussian process and is fitted on the non-transformed data. 95% confidence intervals for $\bar{y}^{(i)}$ were calculated based on assuming normality of simulation outputs.

Budget Distribution via Optimal Computing Budget Allocation

Bartz-Beielstein et al. [3] presented the SPO-OCBA algorithm, a modified version of the SPO algorithm and an instance of Algorithm 2. As replication strategy they integrate *Optimal Computing Budget Allocation* (OCBA). OCBA is a ranking and selection (R & S) procedure that was introduced by Chen et al. [7] for intelligent determination of the most efficient replication numbers in the context of optimization of discrete event systems.

SPO-OCBA is implemented within the SPOT toolbox [2]. The toolbox allows the user to choose between noise-free and noisy Gaussian process models and random forests as surrogate model, which is fitted on the non-aggregated data in each iteration. Subsequently, the points that minimize the *mean response (mr)* infill criterion are proposed. The intensification mechanism works as follows: the new proposed point is evaluated at least $b_{init} \geq 2$ times and added to the set of design points D. Then, a predefined budget b_{OCBA} is distributed among all already evaluated design points to approximately maximize the *probability of correct selection (PCS)*.

Definition 3.1 (Probability of Correct Selection)
Given a design D, the probability of correct selection is defined as the probability that the observed best design $\boldsymbol{x}^{(inc)}$ (the incumbent) is actually the best design

$$P(CS) := P\left(f(\boldsymbol{x}^{(inc)}) < f(\boldsymbol{x}^{(i)}), i \neq inc \mid D\right).$$

The aim is now to maximize the probability of correct selection through optimal allocation of a replication budget. Let $r(i)$ be the number of replications that should be allocated to design i. We follow the definition of Chen et al. [7] and formulate the OCBA-PCS Problem 3.2.

Problem 3.2 (OCBA-PCS Problem)

$$\begin{aligned} &\max_{r^{(1)},\ldots,r^{(n)}} P(CS) \\ &\quad \text{s. t. } r^{(1)} + r^{(2)} + \ldots r^{(n)} = r \end{aligned}$$

Note that r is the total number of replications after the current step (that is, the replications that have already been performed plus the b_{OCBA} replications to be distributed in the current step).

Simplifying assumptions (see Assumptions A.1, A.2) yield a simple analytical formulation of the posterior distribution (see Lemma A.3). Chen et al. approximate

the probability of correct selection in Problem 3.2 by using the Bonferroni inequality, which yields *Approximate Probability of Correct Selection (APCS)* (see Definition A.4). They have shown, that the resulting approximate problem (see Problem A.5) has, asymptotically for $r \to \infty$, a closed-form solution

$$\begin{aligned} \frac{\hat{r}^{(i)}}{\hat{r}^{(j)}} &= \left(\frac{\tau_i / \delta_{inc,i}}{\tau_j / \delta_{inc,j}} \right)^2, \qquad i, j \in \{1, 2, ..., n\}, i \neq j \neq inc, \\ \hat{r}^{(inc)} &= \tau_{inc} \sqrt{\sum_{i=1, i \neq inc}^{n} \frac{(\hat{r}^{(i)})^2}{\tau_i^2}}. \end{aligned} \tag{3.1}$$

Here, $\delta_{inc,i} := \bar{y}^{(inc)} - \bar{y}^{(i)}$ denotes the difference of means of evaluations belonging to the i-th design point and the observed best design point $\boldsymbol{x}^{(inc)}$. Equation 3.1 describes the budget ratios, which allows to calculate the optimal number of replications on each point given a fixed budget b_{OCBA}. The budget that is to be allocated in the current step results as the difference of the budget that already has been allocated and the budget that fulfils Rule 3.1.

The OCBA intensification strategy is outlined in Algorithm 4. We consider the OCBA intensification mechanism as one possible plug-in for Algorithm 2. Figure 3.2 shows that replications are distributed among all (already evaluated) design points. The number of allocated replications per location is the higher, the lower the empirical mean and the higher the empirical standard deviation of past observations.

Note that the theoretical assumptions made in OCBA are strong and often violated: First, simulation outputs $f(\boldsymbol{x}^{(i)}), i = 1, ..., n$, are assumed to be independent across designs, which is a discrepancy to the Gaussian process assumption, where the correlation between function outputs is modelled explicitly by a kernel function. Second, noise variances $\tau^2(\boldsymbol{x})$ are assumed to be known, which is usually not the case in practice. Though results could be derived for unknown variances, we stick to the derivations by Chen et al. where τ_i^2 is simply replaced by its empirical version $\hat{\tau}_i^2$ in Equation 3.1. This, in turn, requires that each point is evaluated at least twice. To improve those estimates over time, the variance estimates are updated in each iteration.

Algorithm 4 *ocba* replication strategy

1: **input:** design D, $\boldsymbol{x}^{(prop)}$ proposed input, initial budget b_{init}, OCBA budget b_{ocba}

initial evaluation:

2: $\left\{\left(\boldsymbol{x}^{(prop)}, y^{(prop,j)}\right)\right\}_{j=1,\ldots,b_{init}} \leftarrow$ evaluate new point b_{init} times

3: $D \leftarrow D \cup \left\{\boldsymbol{x}^{(i)}, y^{(prop,j)}\right\}_{j=1,\ldots,b_{init}}$

replication:

4: $\left(\hat{r}^{(1)}, \ldots, \hat{r}^{(n)}\right) \leftarrow$ calculate allocation of budget b_{OCBA} using Rule 3.1

5: **for** $i = 1, \ldots, n$ **do**

6: $\quad \left\{\left(\boldsymbol{x}^{(i)}, y^{(i,j)}\right)\right\}_{j=1,\ldots,\Delta_i} \leftarrow$ perform $\Delta_i = \hat{r}^{(i)} - r^{(i)}$ replications

7: $\quad D \leftarrow D \cup \left\{\left(\boldsymbol{x}^{(i)}, y^{(i,j)}\right)\right\}_{j=1,\ldots,\Delta_i}$ update design

8: **end for**

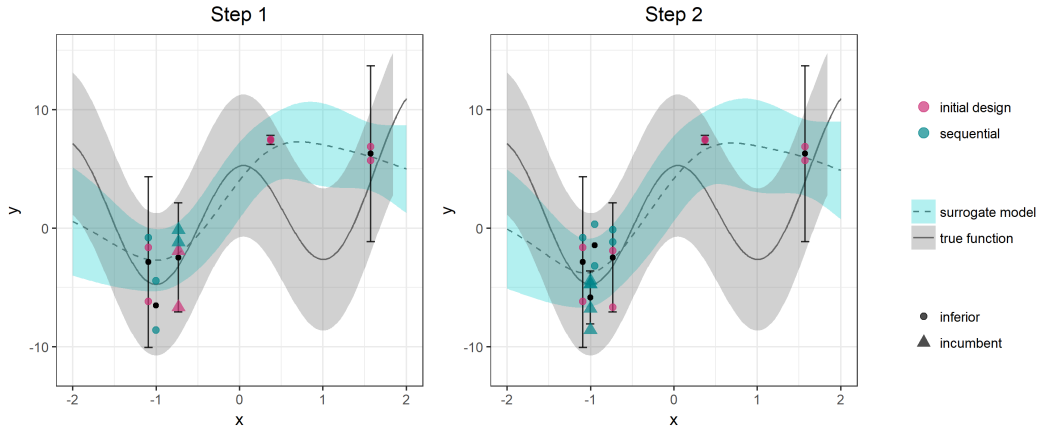


Figure 3.2: Two steps of SPO-OCBA algorithm starting from an initial Latin hypercube design of size 4 (violet points). In each iteration, a new point is evaluated $b_{init} = 2$ times and a budget of $b_{OCBA} = 3$ is distributed all design points according to the OCBA rule. Points with lower mean and higher variance are given more replications. 95% confidence intervals for $\bar{y}^{(i)}$ were calculated based on assuming normality of simulation outputs.

3.2 Replication Strategies without Budget Specifications

While the replication strategies described so far require a predefined budget for replication, the following intensification mechanisms do not.

SPO+-like Incumbent Strategy

Hutter et al. [13] have proposed SPO+ by modifying the intensification strategy of the SPO algorithm.

Before entering to the optimization loop, SPO+ computes the mean observed function values $\bar{y}^{(i)}$ for each of the points evaluated so far to determine the incumbent. In each iteration, an interpolating Gaussian process model is fitted on the mean aggregated and log-transformed data. The configuration minimizing the expected improvement criterion is proposed, evaluated once and added to the set of design points.

The design points are then intensified as follows: A set of candidates is selected to challenge the incumbent configuration $\boldsymbol{x}^{(inc)}$. This set includes the most recently added point $\boldsymbol{x}^{(prop)}$ as well as additional $m \geq 0$ previously evaluated parameter settings $\{\boldsymbol{x}^{(i)}\}_{i \in \{1,\ldots,n\} \setminus inc}$. The challengers are randomly sampled without replacement with probabilities proportional to $\frac{1}{\bar{y}^{(i)}}$. Each one of these points is then challenged against the incumbent. For each of the challengers, we perform runs until either the challenger is (empirically) inferior to the incumbent, i. e. $\bar{y}^{(i)} > \bar{y}^{(inc)}$, or an equal number of evaluations was performed on the new point while still being (empirically) superior to the incumbent, i. e. $r(\boldsymbol{x}^{(i)}) \geq r(\boldsymbol{x}^{(inc)})$ (and still $\bar{y}^{(i)} \leq \bar{y}^{(inc)}$) [1]. In the first case, the challenger is rejected as (probably) inferior. In the latter case, $\boldsymbol{x}^{(i)}$ replaces the incumbent.

Algorithm 5 outlines the intensification strategy, further referred to as *inc+*, which we infer from the SPO+ algorithm. Figure 3.3 illustrates a step of the SPO+ algorithm.

SMAC-like incumbent strategy

Another slight variant of SPO+ has been proposed by Hutter et al. [12]. It is named after its original field of application: *Sequential Model-based Algorithm Configuration (SMAC)*.

As SPO+, SMAC determines the incumbent $\boldsymbol{x}^{(inc)}$ as the point with the minimum mean performance $\bar{y}^{(inc)} = \min_{i=1,\ldots,n} \bar{y}^{(i)}$. In each iteration, a random forest is fitted on the non-aggregated design points. The input $\boldsymbol{x}^{(prop)}$ that minimizes the expected

[1] When starting a race, each challenger is evaluated once. To reduce overhead, the number of runs for the challenger $\boldsymbol{x}^{(i)}$ is doubled each time the challenger is not rejected to reduce overhead.

Algorithm 5 *inc+* replication strategy

input: design D, $\boldsymbol{x}^{(prop)}$ proposed input, incumbent $\boldsymbol{x}^{(inc)}$, challengers

incumbent evaluation

$y^{(inc,r^{(inc)}+1)} \leftarrow$ perform evaluation for incumbent $\boldsymbol{x}^{(inc)}$

$D \leftarrow D \cup \left\{\left(\boldsymbol{x}^{(inc)}, y^{(inc,r^{(inc)}+1)}\right)\right\}$ update design

challenge incumbent

$C \subset \{1, ..., n\} \leftarrow$ sample m configurations

for $k \in C \cup prop$ **do**

  $b \leftarrow 1$

  repeat

    $\left\{\left(\boldsymbol{x}^{(k)}, y^{(k,j)}\right)\right\}_{i=1,...,b} \leftarrow$ perform b evaluations for challenger $\boldsymbol{x}^{(k)}$

    $D \leftarrow D \cup \left\{\left(\boldsymbol{x}^{(k)}, y^{(k,j)}\right)\right\}_{i=1,...,b}$ update design

    $b \leftarrow 2 \cdot b$ double replications

  until $\bar{y}^{(inc)} < \bar{y}^{(k)}$ or $r^{(inc)} \leq r^{(k)}$

end for

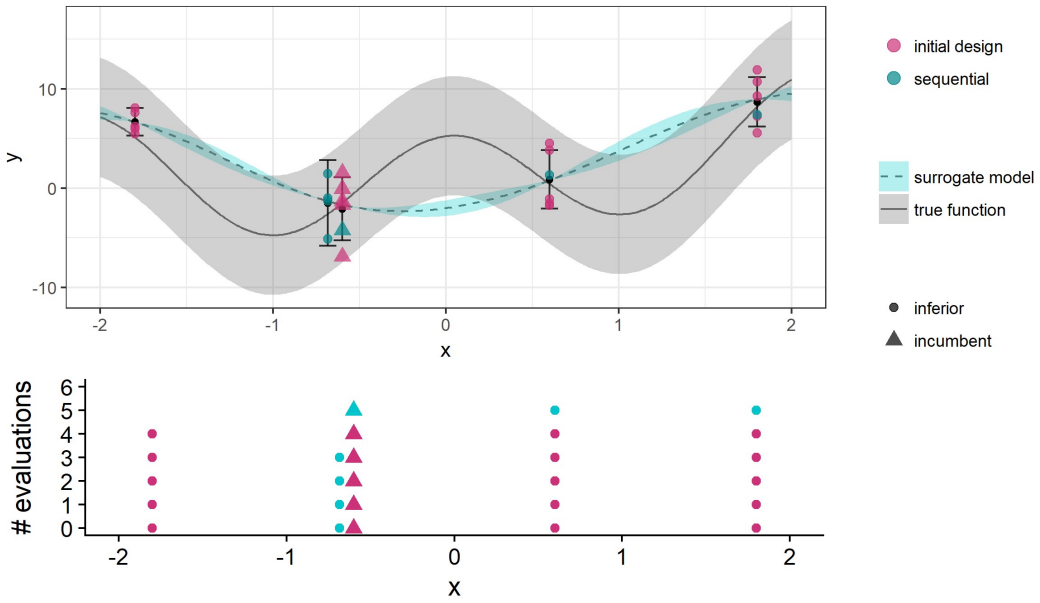


Figure 3.3: One step of the SPO+ algorithm starting from an initial Latin hypercube design of size 4 (violet points). Initial design points are re-evaluated 5 times. The lower figure shows the number of replications that were performed on each configuration. In the current step, the incumbent $x = 0.6$ is evaluated once. The new proposed point $x = 0.68$ as well as $m = 2$ other design points are raced against the incumbent but cannot replace it.

improvement criterion is evaluated and added to the design. This point is then challenged against the incumbent in an analogous manner as in SPO+. SMAC and SPO+ basically differ in the choice of the surrogate learner (which removes the key limitation of being applicable to numerical problems only), their aggregation method and the size of the challenger set [2].

Algorithm 6 outlines the intensification strategy, further referred to as *inc*, which is inferred from the SMAC algorithm. Figure 3.4 illustrates one step of SMAC.

Algorithm 6 *inc* replication strategy

1: **input**: design D, $\boldsymbol{x}^{(prop)}$ proposed input, incumbent $\boldsymbol{x}^{(inc)}$

incumbent evaluation

2: $y^{(inc,r^{(inc)}+1)} \leftarrow$ perform evaluation for incumbent $\boldsymbol{x}^{(inc)}$

3: $D \leftarrow D \cup \left\{ \left(\boldsymbol{x}^{(inc)}, y^{(inc,r^{(inc)}+1)} \right) \right\}$ update design

challenge incumbent

4: $b \leftarrow 1$

5: **repeat**

6: $\quad \left\{ \left(\boldsymbol{x}^{(prop)}, y^{(prop,j)} \right) \right\}_{i=1,\ldots,b} \leftarrow$ perform b evaluation for challenger $\boldsymbol{x}^{(prop)}$

7: $\quad D \leftarrow D \cup \left\{ \left(\boldsymbol{x}^{(prop)}, y^{(prop,j)} \right) \right\}_{i=1,\ldots,b}$ update design

8: $\quad b \leftarrow 2 \cdot b$ double replications

9: **until** $\bar{y}^{(inc)} < \bar{y}^{(prop)}$ or $r^{(inc)} \leq r^{(prop)}$

[2] Note that the intensification mechanism described by Hutter et al. [12] is designed to differentiate different problem instances. The simplified procedure described here results from considering one problem instance only.

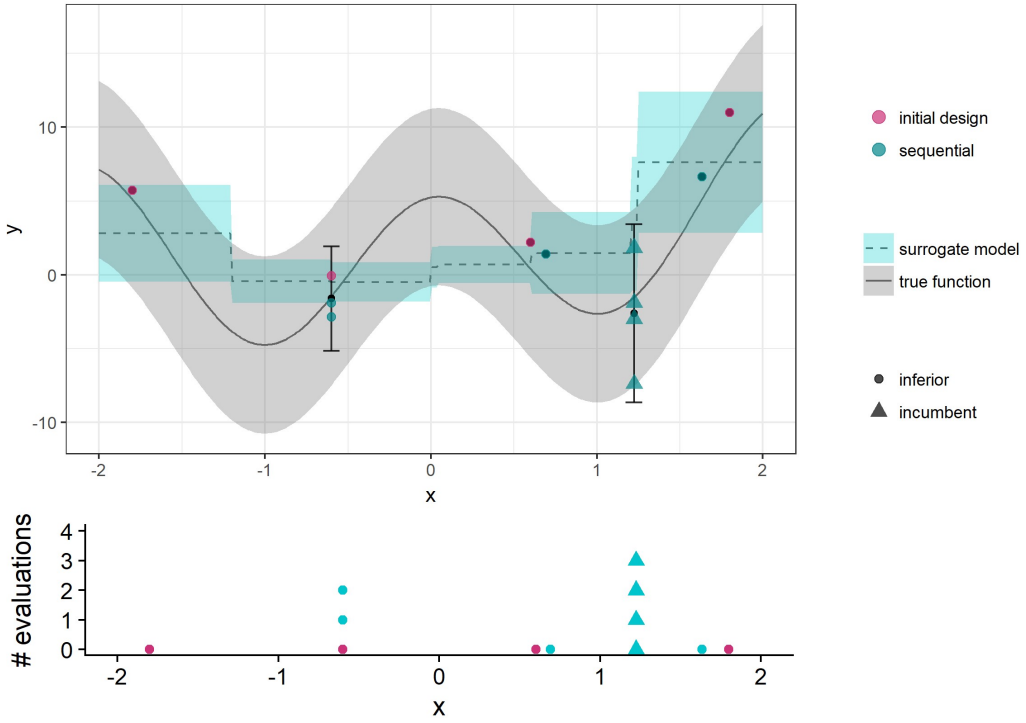


Figure 3.4: One step of the SMAC algorithm starting from an initial Latin hypercube design of size 4 (violet points). The lower figure shows the number of replications that performed on each configuration.

3.3 Replication Strategies with User pre-set Confidence

Replication strategies that require the specification of a budget (see Section 3.1) are difficult to apply in practice: the replication budget needs to be specified by prior knowledge, experience or some well-established default values. The incumbent strategies described in Section 3.2 are superior in this regard: they do not require a user pre-set budget (at worst a hyperparameter like the size of the challenger set that impacts the allocated budget indirectly).

Especially for practical purposes, it might be of great value for the user if he could specify a desired level of confidence in the final solution before starting optimization. The algorithm shall determine internally the number of replications that are necessary to distinguish the best design point from inferior design points with the said level of confidence.

The first question we address is how to express confidences in solutions. One way to come up with guarantees in returned solutions is to carry out a statistical test after

each iteration and reject inferior points. The final solution(s) are the points that have not been rejected with respect to a specified level of significance. Apart from SMBO, racing algorithms [18] have been used for algorithm configuration problems and have gained attention in this field. However, statistical testing will not be pursued in this work, as one easily runs into multiple testing issues.

The probability of correct selection, which was introduced in Section 3.1, is another way to express the statistical guarantees. Instead of prescribing a budget and - to formulate it a little bit offhand - leave it to the algorithm to make the best out of it, the strategy is inverted: minimize the replication budget while ensuring a desired level of probability of correct selection P^*.

Problem 3.3 (OCBA dual)

$$\min_{r^{(1)},\ldots,r^{(n)}} r^{(1)} + r^{(2)} + \ldots r^{(n)}$$
$$\text{s. t. } P(CS) \geq P^*$$

This optimization problem is formally stated in Problem 3.3. According to Chen et al. [8] it is the dual to Problem 3.2. Under the assumptions described in Appendix A, they have shown that its approximate solution corresponds to that of the primal problem.

Note that the problem basically corresponds to the problem of finding a *feasible* points: If the constraint $P(CS) \geq P^*$ is fulfilled, the optimum is $r^{(1)} = ... = r^{(n)} = 0$ and no budget will be allocated. If the constraint is not fulfilled, Chen et al. propose to proceed as follows: In each iteration, a very small budget[3] is allocated to existing design points according to Rule 3.2 until the probability of correct selection of at least P^* is met. The probability of correct selection is calculated in each iteration based on Assumptions A.1 and A.2 using Lemma A.3.

However, the integration of this dual approach poses the following problems: there might be points in the design that are exactly on the same (bad) level and the algorithm tries to distinguish them desperately. Furthermore, for continuous functions, a new design point can be arbitrarily near to an already existing design point. The two problems are illustrated in Figure 3.5.

This behavior, in turn, is less annoying or even desired in the end of the optimization procedure: imagine we already explored the input space sufficiently, and the remaining budget is spent on existing design points to identifiy the final best point.

In this thesis, a new method dividing the SMBO procedure in two phases (*optimization* and a *identification*) is proposed. The first phase aims at optimizing the

[3] We stick to the recommendations of $b_{OCBA} = 3$ by Chen et al..

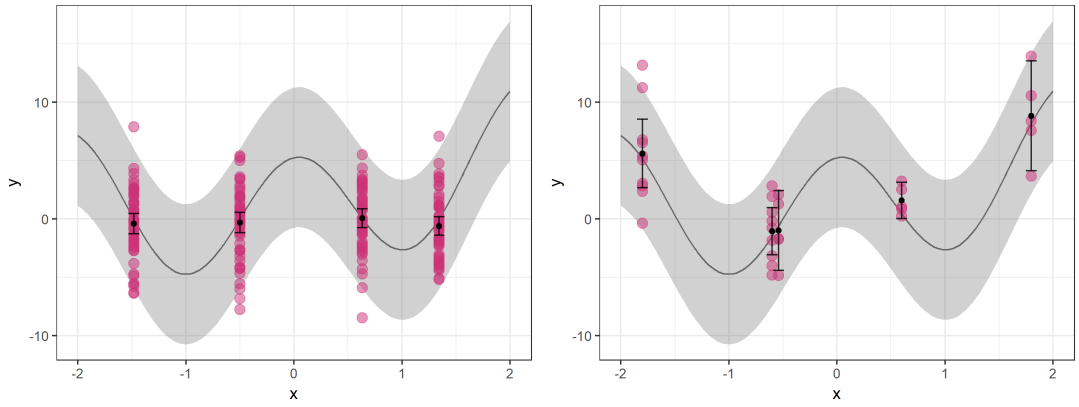


Figure 3.5: Illustration of Problems of the OCBA-dual Approach

function and discovering interesting regions of the function. The second phase aims at identifying the best point among all design points and gives back confidences in the proposed solution. Note that in this second phase, no new points are added to the design. In practice, the second phase can be run until the minimum required probability of correct selection has been reached. However, there is still the risk to run into the convergence issues described above. Thus, we recommend to limit the identification phase by an absolute maximum time limit in case of slow convergence.

Note that the two phases are that modular and could in practice be interrupted and continued at any time by the user. For instance, the optimization phase could be followed by an identification step that doesn't seem to converge. The user can inspect the probability of correct selection reached so far, resume the optimization phase and try to identify the final best point afterwards again.

Algorithm 7 outlines our new proposed approach, SMBO WITH IDENTIFICATION, which is inspired by the OCBA-PCS dual 3.3[4]. The algorithm is examplarily vizualised in Figure 3.6.

[4]Note that this approach still requires budget specifications, which however play a less important role than in Section 3.1 as the total replication budget is determined by the iteration performed. b_{init} should be big enough to ensure sufficient variance estimates for each of the configurations, for $b_{OCBA} = 3$ seems to be a good option.

Algorithm 7 SMBO with identification

input: infill criterion $\mathcal{I}$, hypothesis space $\mathcal{H}$, minimum desired PCS P^*, initial number of evaluations b_{init}

generate initial design D
optimization phase:
while optimization budget not spent **do**
    fit a surrogate $\hat{f} \in \mathcal{H}$ on the (aggregated) data $t(D)$
    propose the point that optimizes $\mathcal{I}(\boldsymbol{x})$ and update D
    intensify new point b_{init} times
end while

identification phase:
calculate PCS
while $PCS < P^*$ and identification budget not spent **do**
    Allocate budget of $b_{OCBA} = 3$ according to Algorithm 4
    calculate PCS
end while

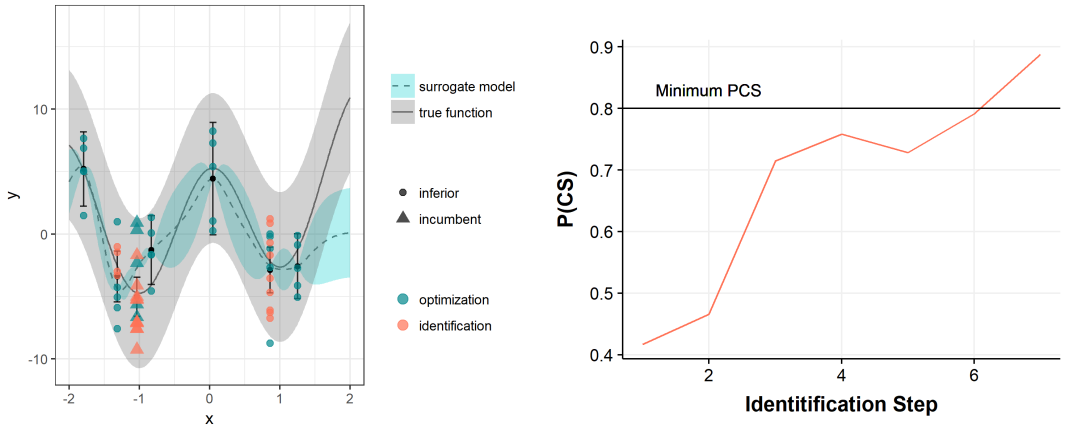


Figure 3.6: In the identification phase of SMBO WITH IDENTIFICATION, budget is allocated on the 7 design points to distinguish inferior from superior design points. In every identification iteration (which corresponds to one iteration in the identification loop in Algorithm 7 and thus to a budget of 3 evaluations), the probability of correct selection increases until it exceeds the minimum required P(CS) of 0.8.

Chapter 4

Experimental Study

Implementation details about this experimental study are presented in Appendix B.

4.1 Experiments with Synthetic Test Functions

The first benchmark is conducted on artificial test functions. As both the underlying black-box function and the structure of the noise are known, this offers us a more in-depth view about the internal behavior of the methods.

This benchmark is divided into three parts: As a starting point, the three original algorithms SPO-OCBA, SMAC and SPO+ are compared against the SKO algorithm that doesn't employ any replication strategy.

After that, the replication strategies we identified from those algorithms are combined with different algorithmic choices of surrogate learners, data aggregation methods and infill criteria in order to systematically find the best variants of SMBO WITH INTENSIFICATION.

Based on insights gained from the first two parts, the new method, SMBO WITH IDENTIFICATION, is compared against SKO and SMBO with replication.

Problem design. All methods are evaluated on four custom test functions $f : \mathcal{X} \to \mathbb{R}$ provided by the `smoof` [5] package: the *Sphere*, the *Rosenbrock*, the *Ackley* and the *Alpine No. 1* function. The functions were chosen in a way to cover a wide range of function properties, from smooth and unimodal to nondifferentiable and multimodal. All functions are scaled to a function standard deviation of 1. Each of the test functions is considered in a low dimensional setting ($dim(\mathcal{X}) = 5$) and a higher dimensional setting ($dim(\mathcal{X}) = 20$). The test functions are presented in Table 4.1.

To simulate a noisy setting, Gaussian noise is added on top of the noise-free test function f, i. e.

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon(\boldsymbol{x}), \qquad \epsilon(\boldsymbol{x}) \sim \mathcal{N}\left(0, \tau(\boldsymbol{x})^2\right). \tag{4.1}$$

The noise level $\tau(\boldsymbol{x})$ is expressed in terms of the proportion of the function standard deviation (which is 1 for all the functions after scaling). For homoscedastic noise, the

Test function		Domain	Properties
Sphere	$f(x) = \sum_{i=1}^{d} x_i^2$	$[-5.1, 5.1]$	unimodal, continuous, differentiable
Rosenbrock	$f(x) = \sum_{i=1}^{d-1} 100 \left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2$	$[-30, 30]$	multimodal, continuous, differentiable
Ackley	$f(x) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i}\right)$ $- \exp\left(\frac{1}{d}\sum_{i=1}^{d} \cos(2\pi x_i)\right)$	$[-32.8, 32.8]$	multimodal, continuous, differentiable
Alpine No. 1	$f(x) = \sum_{i=1}^{d} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	$[-10, 10]$	multimodal, continuous, non-differentiable

Table 4.1: Test functions used within the benchmark and their respective properties.

noise standard deviation function is constant $\tau(\boldsymbol{x}) \equiv \tau$. We set $\tau \in \{0.05, 0.25, 0.5\}$ to investigate in functions with low, medium and high noise levels. For heteroscedastic noise, the noise variance function $\tau^2(\boldsymbol{x})$ is modelled by two different functions: either by the *Sphere* function or the *Rosenbrock* function, which are both scaled to a range of $[0.1, 0.5]$. The problem design is presented in Figure 4.1.

Algorithm	Intensification	Surrogate	Infill Crit.	Aggregation
SKO	*none*	*km.nugget*	*aei*	*none*
SPO-OCBA	*ocba*	*km.nugget*	*mr*	*none*
SMAC	*inc*	*rf*	*ei*	*none*
SPO+	*inc+*	*km*	*ei*	*mean*

Table 4.2: Algorithm specifications for SKO, SPO-OCBA, SPO+, SMAC.

Algorithm design. First, the three algorithms presented in Chapter 3, SPO-OCBA, SPO+ and SMAC are compared against the SKO algorithm. Their specifications are summarized in Table 4.2[1].

Second, we fuse the different intensification mechanisms identified in Chapter 3 with different algorithmic choices we expect to have a high influence on the performance of SMBO: the surrogate model, the infill criterion and the way data is aggregated before fitting the surrogate model.

We will compare three custom choices for a surrogate model: the noise-free Kriging

[1]Note that for $p = 20$ the surrogate models are fitted on the mean aggregated data only to avoid excessive runtimes for Kriging if the design gets too large. Therefore, original methods are slightly modified in this regard which is marked by subscript $_{(m)}$.

model[2] (*km*)[22], the noisy Kriging model (*km.nugget*)[22] and the random forest (*rf*)[17].

As infill criterion we include the classical expected improvement (*ei*), the augmented expected improvement (*aei*), the expected quantile improvement (*eqi*) and the lower confidence bound (*cb*) in our benchmark. For 5-dimensional test functions, data is either not aggregated (*none*) or y-values belonging to the same configuration x are aggregated to their empirical mean (*mean*) before fitting the surrogate model. For $p = 20$, we only consider mean aggregation. The algorithm design is presented in Figure 4.2. Combining all algorithmic factors results in 96 algorithm instances in the 5-dimensional case and 48 in the 20-dimensional case[3].

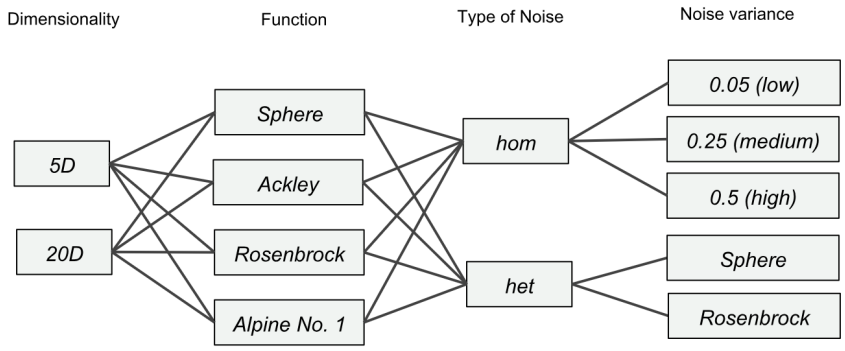


Figure 4.1: Benchmark on Synthetic Test Functions: Problem Design.

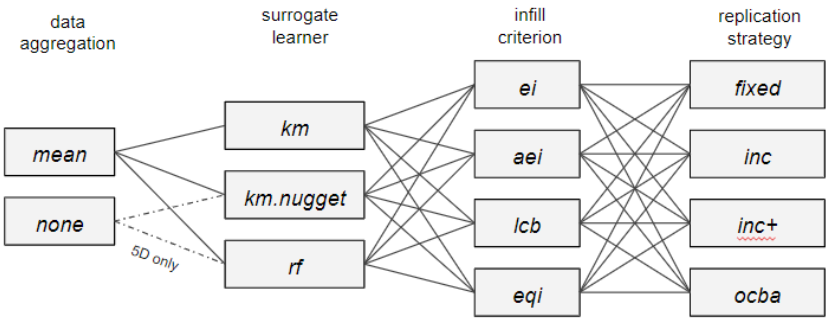


Figure 4.2: Benchmark on Synthetic Test Functions: Algorithm Design.

[2] To ensure numerical stability, a nugget of 1×10^{-7} was added.

[3] Note that from a theoretical perspective the combination of the strictly interpolating noise-free Kriging and no aggregation is not possible. As a small nugget effect was added for numerical stability, this combination is practically still feasible and thus investigated.

Third, we compare the new proposed method, SMBO WITH IDENTIFICATION, to SMBO with fixed replication (which basically corresponds to the *optimization phase* of Algorithm 7) and to the SKO algorithm. As summarized in Table 4.3, the number of initial evaluations per point is set to $b = 5$ for the two variants that employ replication. To allow intercomparability, the identification phase in SMBO WITH IDENTIFICATION terminates after 100 function evaluations (risking that the probability of correct selection is not reached due to this short identification phase).

To keep the number of experiments small, we investigate the 5-dimensional problems only. Furthermore, the surrogate learner (*km.nugget*), the infill criterion (*cb*) and aggregation method (*mean*) are kept constant here[4].

Algorithm	**Initial replications** per config.	**Optimization budget** max. evals	**Identification budget** max. evals
SKO	$b = 1$	500	0
SMBO W/ INTENSIFICATION	$b_{init} = 5$	500	0
SMBO W/ IDENTIFICATION	$b_{init} = 5$	400	100

Table 4.3: Algorithms that are compared for a first validation of SMBO WITH IDENTIFICATION.

Algorithmic factor		**Hyperparameters**	**Reference**
replication strategy	fixed	$b = 5$	-
	ocba	$b_{init} = 2$, $b_{OCBA} = 3$	[3]
	inc+	$m = 5$	[13]
	inc	-	-
surrogate learner	*km*	Matérn-3/2 kernel	`mlrMBO` [4]
	km.nugget	Matérn-3/2 kernel	
	rf	500 trees	
infill criterion	ei	-	`mlrMBO` [4]
	aei	-	
	eqi	$\beta = 0.7$	
	cb	$\lambda = 1$	

Table 4.4: Hyperparameter settings of different choices.

Note that some of the different algorithmic choices presented depend on higher-level parameters. To keep the number of experiments to a minimum, those are set to a reasonable default, overtaken either from recommendations of the original authors or from common implementations (see Table 4.4). Other algorithmic choices like the type

[4]The nugget-effect Kriging model, the lower confidence bound criterion and mean aggregation were chosen after inspection of first results for the benchmark above and classified as superior to other methods (see results below)

or the size of the initial design, the number of points proposed by the infill criterion or the infill optimizer are also set to a default: The initial design is a Latin hypercube design of size $5 \times p$, the number of proposed points per iteration is set to $m = 1$. The infill criterion is optimized by focus search [4]. We expect these choices not to distort results. For SKO, SPO-OCBA, SMAC and SPO+ choices might deviate from choices made by the original authors. In favor of a consistent assessment and a uniform implementation, we neglect those discrepancies.

Execution of Experiments. Experiments are repeated 20 times. Every experiment terminates if a total number of $100 \times p$ function evaluations has been reached and returns the incumbent configuration. For SMBO WITH IDENTIFICATION, the last $20 \times p$ function evaluations are spent for identification.

Though usually time is the limiting factor for noisy black-box optimization, it would be misleading in this artificial setup as the model fitting will determine runtime instead of the function evaluations. Algorithms that perform more replications would perform much more function evaluations than strategies that only perform one iteration in each step like SKO. We allow each algorithm to perform a fixed number of function evaluations, which means that the number of model iterations (i. e. the number of model fits) depends on the replication strategy. We leave it to the algorithm to determine an "intelligent" ratio between function evaluations and model iterations.

Evaluation of Results. Experiment results will be evaluated with respect to two criterions:

- *Overall error* $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^*)$: The difference between the true noise-free optimum value $f(\boldsymbol{x}^*)$, $\boldsymbol{x}^* := \arg\min_{x \in \mathcal{X}} f(\boldsymbol{x}^*)$ (theoretical best result), and the noise-free function value $f(\boldsymbol{x}^{(\boldsymbol{inc})})$ at the incumbent configuration.
- *Identification error* $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^{**})$: The error that arises from false identification of the final best point, that is the difference between the noise-free function value at the incumbent configuration $f(\boldsymbol{x}^{(\boldsymbol{inc})})$ and the actual best point in the design $f(\boldsymbol{x}^{(**)})$, $\boldsymbol{x}^{**} = \arg\min_{x \in D} f(x)$.

Note that the identification error is a part of the *overall error* and arises if the incumbent configuration does not correspond to the actual best point in design. In this artificial setting we can quantify both errors as the true underlying function is known.

Results. In Table 4.5 the average ranks for SKO, SPO-OCBA, SMAC and SPO+ with respect to the overall error are presented. The SKO is superior in most 5-dimensional problems (see Figure 4.4a). The optimization paths for the SKO decrease faster, but the especially for high noise levels the differences to other methods are small

(a) 5-dimensional test functions

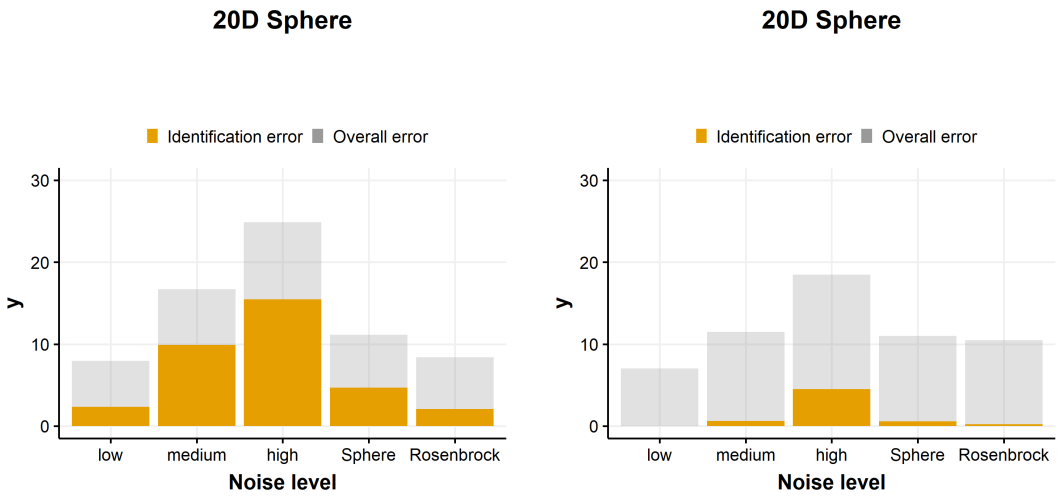
Algorithm	Avg. rank
SKO	**1.61**
SPO-OCBA	2.08
SMAC	2.68
SPO+	2.88

(b) 20-dimensional test functions

Algorithm	Avg. rank
$\text{SKO}_{(m)}$	1.85
$\textbf{SPO-OCBA}_{(m)}$	**1.76**
$\text{SMAC}_{(m)}$	3.09
SPO+	2.45

Table 4.5: Average ranks for the original algorithms SKO, SPO-OCBA, SMAC and SPO+ w. r. t. the overall error $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^*)$. Results were ranked in each replication and then averaged over the replications and problem instances.

(see Figure C.1). On most 20-dimensional problems, the SPO-OCBA outperforms SKO (see Figure C.3a). We see that in most cases a major part of the overall error made by SKO can be explained by false identification of the final best point. The identification errors for the original methods are presented in Figures C.2a and C.4a. Figure 4.3 emphasizes this observation showing the mean identification error across all experiments for SKO compared to SPO+ (which reduces the identification error through replication) for the 20-dimensional *Sphere* function in proportion to the mean overall error.



(a) SKO **(b)** SPO+

Figure 4.3: Mean identification error $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^{**})$ (orange) in proportion to the mean overall error $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^*)$ (grey) across experiments for the 20-dimensional *Sphere* function.

Results obtained from combining each of the four replication strategies (*fixed*, *ocba*, *inc*, *inc+*) with different surrogate models, infill criteria and aggregation methods are investigated through a global rank analysis (see Tables C.1 and C.2).

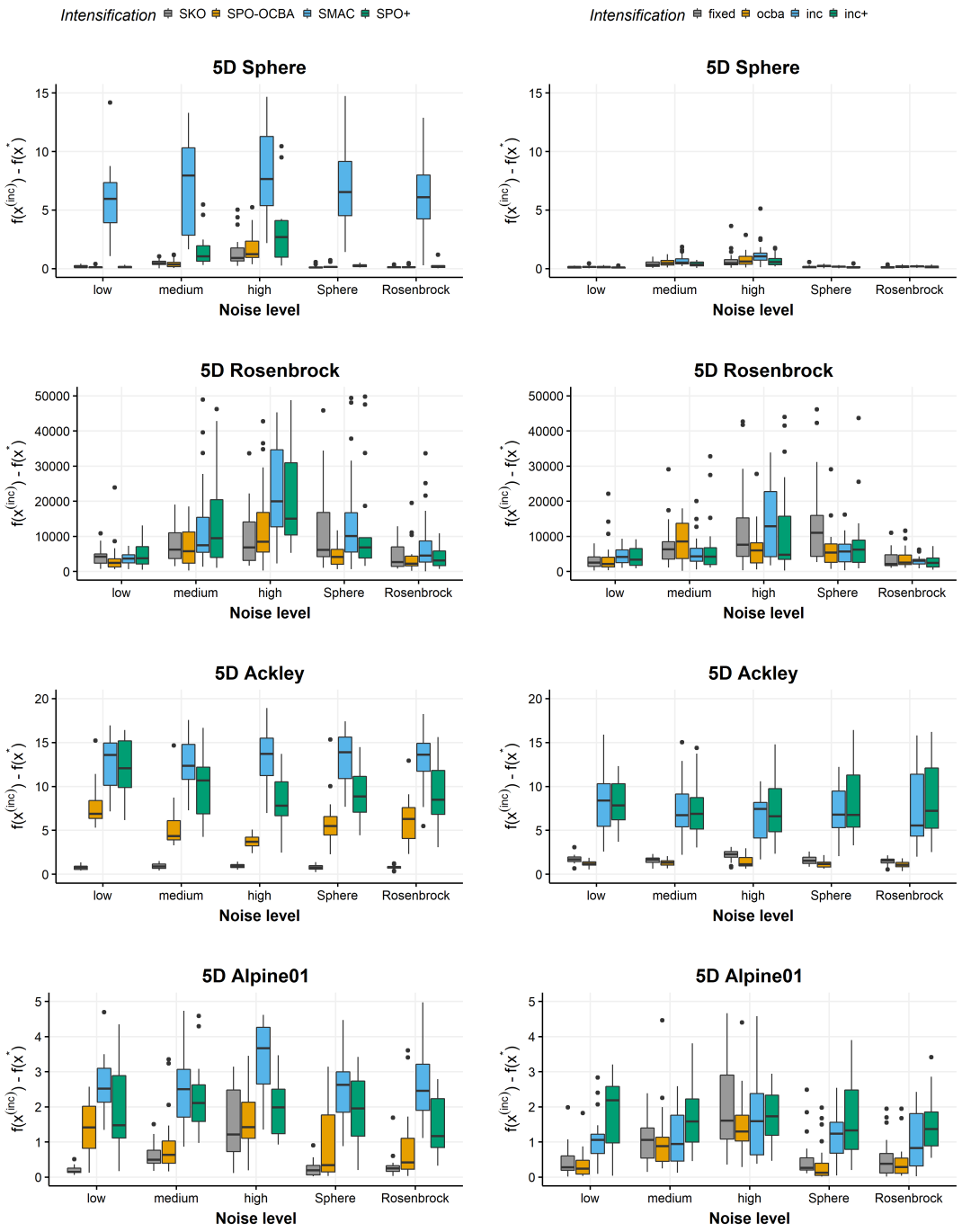
For each of the four replication strategies, the noisy Kriging model systematically outperforms the noise-free Kriging model and the random forest. Furthermore, results were better when the surrogate models were fitted on the aggregated data. From the experiments that have been performed we cannot infer a clear superiority of one of the infill criteria.

For a better visual comparison of the four replication strategies we consider each of them in its "optimized" configuration, i. e. with a noisy Kriging model as surrogate learner and *mean* aggregation (see Table 4.6). Assuming that the effect of the infill criterion is limited, it is set to the overall best performing one (*cb*).

Figure 4.4b shows the performance of the four final candidates for $p = 5$. Though the performance considerably improved through an appropriate specification of the surrogate learner and the aggregation method, the methods still seem to be inferior to the SKO algorithm. While the SKO algorithm seems to be able to handle the noise for the lower dimensional problems, it is inferior to *ocba* in the higher dimensional setting $p = 20$ (see Figure C.3). The respective identification errors are shown in Figures C.2b and C.4b. Optimization paths for problems with high noise levels are presented in Figure C.5.

intensification	surr.	crit.	agg.
fixed	km.nugget	cb	mean
ocba	km.nugget	cb	mean
inc	km.nugget	cb	mean
inc+	km.nugget	cb	mean

Table 4.6: The four replication strategies *fixed, ocba, inc* and *inc+* in their optimal configurations w. r. t. surrogate learner and aggregation. The infill criterion was set to the overall best performing (*cb*).



(a) Original methods (see Table 4.2) **(b)** Optimized strategies (see Table 4.6)

Figure 4.4: Overall error $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^*)$ from experiments on the 5-dimensional test functions.

Finally, we analyse the experiments performed with the new proposed method in order to get a first impression of whether the identification phase in the end of the optimization procedure can improve the overall optimization result. Figure 4.5 shows the optimization paths for high noise levels on the four test functions. SMBO WITH IDENTIFICATION spends the last 100 function evaluation on existing design points only in order to identify the final best point in design. We see that the SMBO WITH IDENTIFICATION successfully reduces the identification error (see Figure C.7).

We see that the optimization path for SKO decreases faster than the other methods as it fits a surrogate model after each evaluation (see Figure 4.5). For the less difficult *Sphere* and *Rosenbrock* functions, the SKO hardly improves after about 300 evaluations and seems to converge. The methods that employ replication, however, achieve a better result here. For the more complex *Ackley* and *Alpine No. 1* function, we see that the SKO algorithm outperforms the other methods for a given budget of 500 iterations. The SMBO WITH IDENTIFICATION seems to further decrease after 500 iterations whilst the others don't. Further work will who if the SMBO WITH IDENTIFICATION could be superior to the other methods when extending the overall budget and thus also the optimization phase.

Besides that, we are interested in the expressiveness of the returned probability of correct selection. Theoretically, a high *P(CS)* should imply a low identification error. The minimum required probability of correct selection P^* has not been reached for the smooth *Sphere* and *Rosenbrock* functions. If functions are very smooth, it can be very hard to distinguish close points, which might be the case for the *Sphere* and the *Rosenbrock* function. Here, a level of 0.75 is unrealistic to achieve for larger designs. In contrast, for the more complex *Alpine No. 1*, the probability of correct selection has been achieved in some of the runs. Considering identification error vs. the probability of correct selection (that actually has been reached) for the Alpine No. 1 function (see Figure 4.6), we may assume that a high probability of correct selection indicates a small identification error. This assumption needs to be validated or rejected in future work.

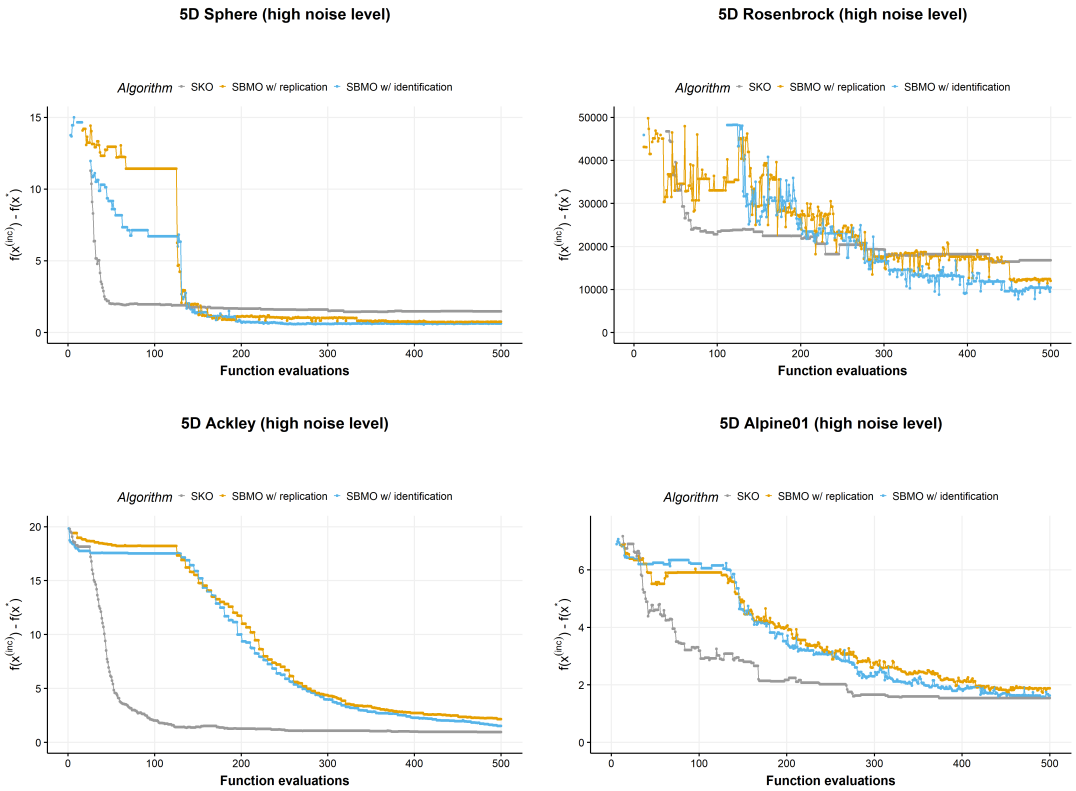


Figure 4.5: Aggregated optimization paths of the SKO, SMBO WITH REPLICATION and SMBO WITH IDENTIFICATION.

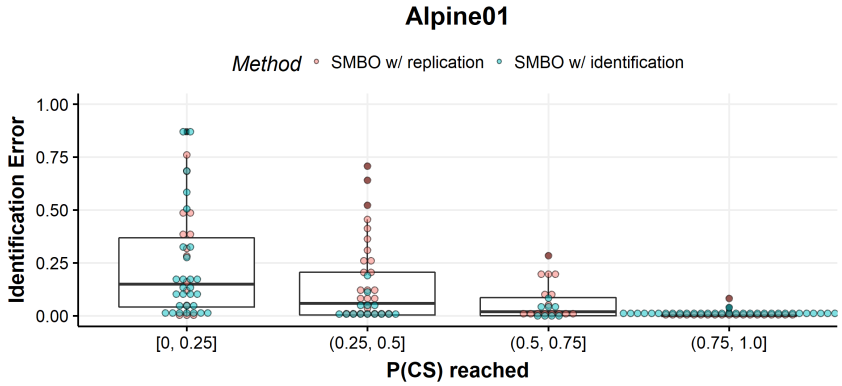


Figure 4.6: Identification error made vs. probability of correct selection reached by the respective methods on the *Alpine No. 1* function with high noise level (note that the probability of correct selection can be calculated as long as variance estimates for the different configurations are available).

4.2 Machine Learning Experiments

The second benchmark situation is a practical setting from machine learning, where sequential model-based optimization is used for hyperparameter tuning. We investigate the tuning of the `xgboost` algorithm [9], which is a scalable machine learning system for tree boosting yielding state-of-the-art performance in many applications. As the performance of the algorithm heavily depends on a large number of hyperparameters for optimization and regularization, the algorithm is a relevant and representative example for this benchmark.

Problem Design. We compare the performance of the tuning algorithms on a subset of the datasets that are used by Thornton et al. [26] for evaluation of the AutoWEKA framework. The chosen datasets contain binary and multiclass classification problems and differ in size, dimensionality and types of the features. The datasets are summarized in Table 4.7.

Algorithm design. First, SMBO with intensification is used for hyperparameter tuning on all of the problems in Table 4.7. We compare the four algorithm versions that have been identified in the first benchmark on artificial test functions (see Table 4.6). In addition, experiments are performed for random forests as surrogate models as well.

Second, we also investigate SMBO with identification for a subset of the problems

Dataset	# Classes	# Training obs.	# Test obs.	# Numeric features	# Factor features
Abalone	28	2923	1254	7	1
Car	2	1209	519	0	6
Dexter	2	420	180	20000	0
GermanCredit	2	700	300	7	2
KR-vs-KP	2	2237	959	0	37
Madelon	2	1820	780	500	0
Secom	2	1096	471	591	0
Semeion	10	1115	478	256	0
Waveform	3	3500	1500	40	0
Wine Quality	7	3425	1469	11	0
Yeast	10	1038	446	8	0

Table 4.7: Summary of the AutoWEKA datasets used for the machine learning benchmark.

in Table 4.7 in order to get a first impression of whether an identification step in the end can improve the final tuning result. Here as well, we use a noisy Kriging model (*km.nugget*) as surrogate learner, the lower confidence bound (*cb*) as infill criterion and *mean* aggregation for all three methods.

Further algorithmic choices are set to the same defaults as for the benchmark on synthetic test functions (see Table 4.4).

Execution of Experiments. Within the SMBO procedure, a function evaluation for a given configuration $\boldsymbol{x}$ corresponds to one randomized[5] holdout iteration (with split rate $\frac{4}{5}$). Each experiment is repeated 20 times and terminates if either a maximum number of 500 evaluations is reached during tuning or a time budget of 15 hours is exceeded. For the comparison of SMBO with identification to SMBO with intensification and SKO, we use the more practically oriented termination criterion of a maximum runtime of one hour. For SMBO with identification, the last 10 minutes were used for identification.

`xgboost` is trained via optimization of the logistic loss for the binary and the softmax loss for multiclass classification problems respectively. Factor features are dummy encoded. The hyperparameter space we tune over is presented in Table 4.8. Note that we tune over the `nrounds` hyperparameter which could have been determined through early stopping as well. Performing replications would result in multiple (potentially different) estimates for the `nrounds` parameter for one and the same configuration, which needs to be aggregated across replications. This raises the interesting question

[5]Note that randomized holdout introduces more noise than if we would have defined fixed folds within the tuning. It is questionable if this kind of noise can help to avoid potential overfitting or if it rather disturbs the optimizer. This question is not covered by this thesis.

of whether replications can help to find a good `nrounds` parameter.

Name	Range	$\log_2$ scale
`nrounds`	$\{1, 2, ..., 5000\}$	no
`eta`	$[0.01, 0.2]$	no
`gamma`	$[-7, 6]$	yes
`max_depth`	$\{3, ..., 20\}$	no
`colsample_bytree`	$[0.5, 1]$	no
`colsample_bylevel`	$[0.5, 1]$	no
`lambda`	$[-10, 10]$	yes
`alpha`	$[-10, 10]$	yes
`subsample`	$[0.5, 1]$	no
`scale_pos_weight`	$[-10, 10]$	yes

Table 4.8: `xgboost` hyperparameter spaces that are tuned over in the benchmark.

Evaluation of Results. We evaluate the performance of the tuned learners in terms of the *mean misclassification error* on a test set that was excluded from training (*mmce*). As test set we use the one that is pre-defined by AutoWEKA and was also used by other authors for the purpose of evaluation.

Results. Results for experiments employing SMBO WITH REPLICATION are visualised in Figure 4.7 for the binary problems (for the multiclass problems see Figure C.8). The ranks across all the data problems are summarized in Table C.3.

From the results on the binary problems, we see that no strategy clearly outperforms the others. Further, random forests as surrogate learners yield comparable performances. Notably, we see that replication strategies yield better results than classical SMBO with no integrated replication strategy.

The results for SMBO WITH IDENTIFICATION are presented in Figure 4.8. The respective ranks across all the problems show that for the experiments conducted, SMBO WITH IDENTIFICATION are superior to the strategies without identification step. We see that the variability of the experiments conducted with SMBO with identification step is smallest among the three methods. Future work will show if SMBO WITH IDENTIFICATION can yield substantial benefits in machine learning tuning.

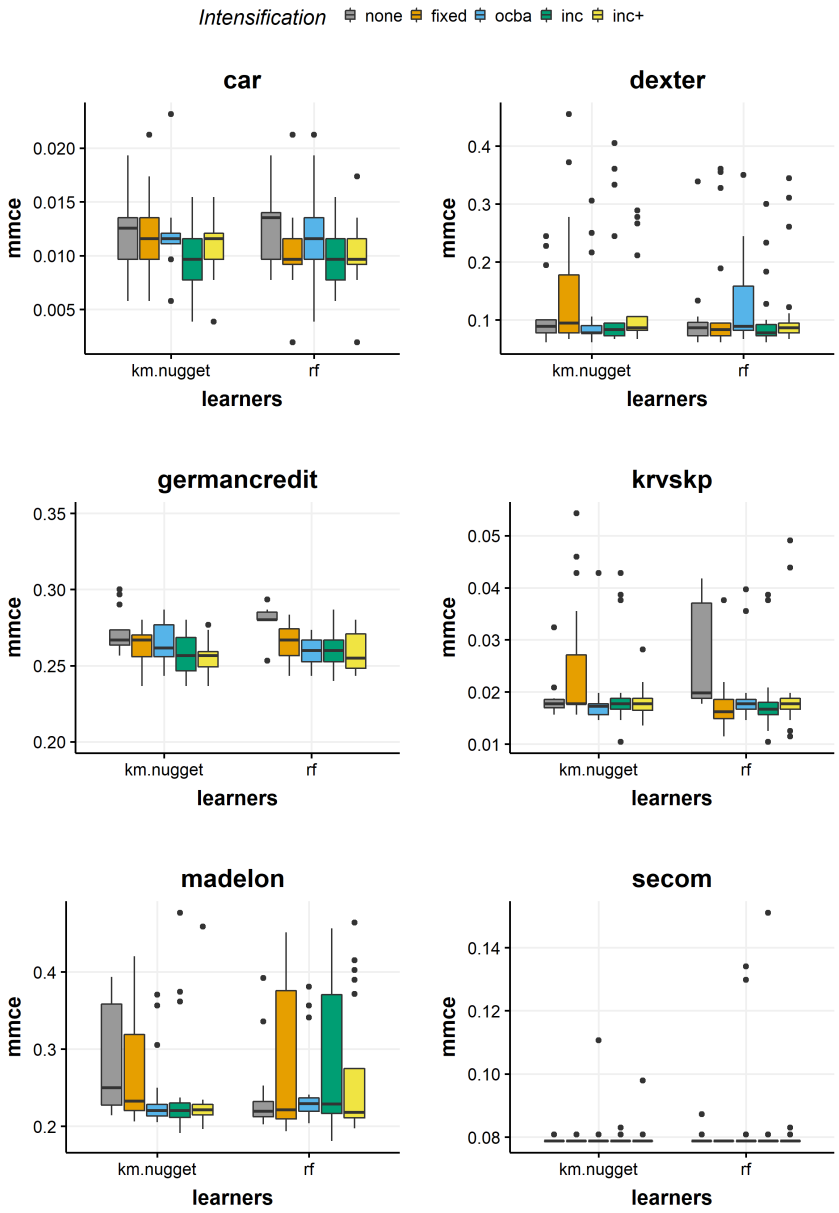


Figure 4.7: Test performance (*mmce*) of the final `xgboost` model after tuning for the different SMBO versions on the binary problems.

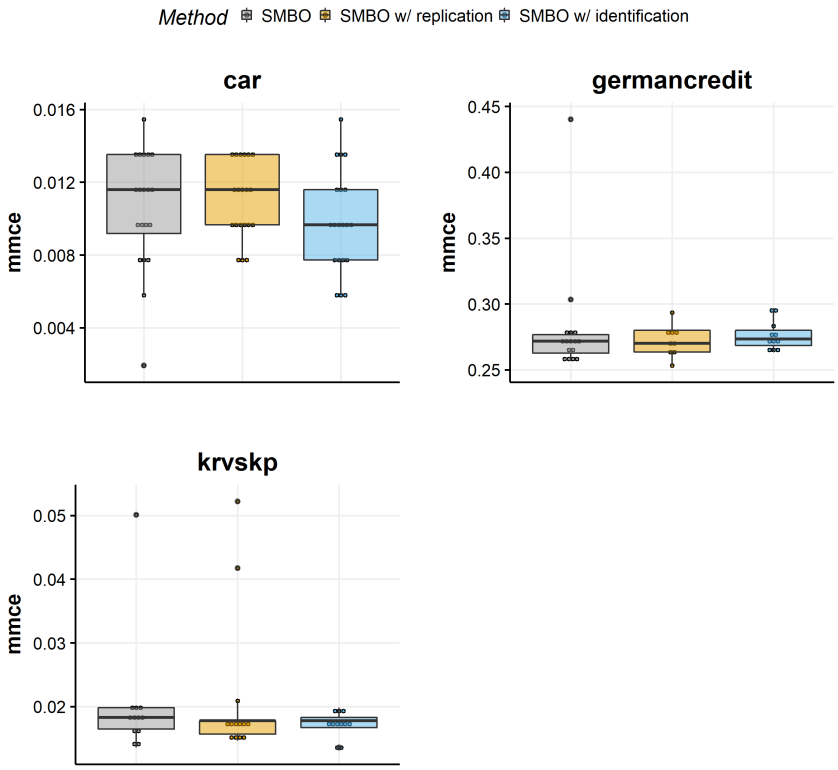


Figure 4.8: Test performance (*mmce*) of the final `xgboost` model after tuning by SMBO, SMBO WITH INTENSIFICATION and SMBO WITH IDENTIFICATION respectively on three of the AUTOWEKA datasets.

intensification	rank
SMBO	2.26
SMBO WITH REPLICATION	2.06
SMBO with identification	**1.62**

Table 4.9: Average ranks for the tuning results of the SMBO, SMBO WITH REPLICATION and SMBO WITH IDENTIFICATION on the datasets CAR, GERMANCREDIT and KRVSKP. Results were ranked in each replication and then averaged over the replications and problems.

Chapter 5

CrowdNav: A Sample Application

When investigating the effect of different design choices on a real-life system, experiments might be not feasible, either for technical or financial reasons. Here, computer simulations can be a useful tool to imitate and analyse those systems. Due to their complexity, simulation experiments often fulfil the characteristics of black-boxes.

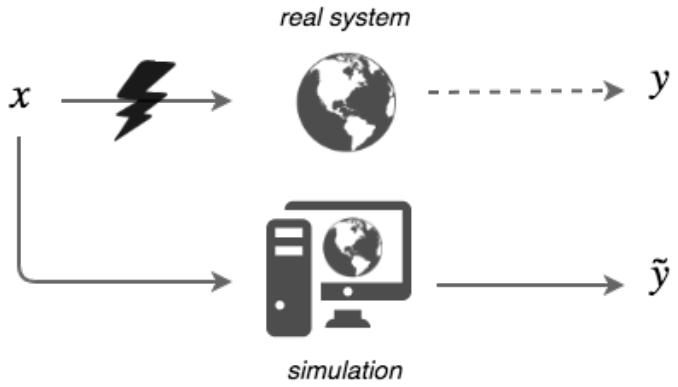


Figure 5.1: When experiments cannot be performed in the real-life, those systems are imitated via computer simulations.

The goal of simulation optimization is to find the optimal input value with respect to some quality criterion using simulations. Each time the objective function, i. e. the underlying relation between input value x and quality criterion y, is evaluated, a simulation is run instead.

In this work, we consider a traffic navigation expemlar: Deploying a new navigation algorithm in real cars in a city for test purposes is difficult if not impossible and brings high risks in case of fail. In order to test such systems, one often uses traffic simulations instead. One such example is CrowdNav [23][10] : the goal is to optimize the duration of car trips in a city by adapting parameters of a *smart* routing algorithm, that is based on static traffic information (e.g. the length and the speed limit of a street) but also on dynamic traffic information (e.g. current traffic volume). The simulation

is based on SUMO (Simulation of Urban Mobility)[15] and TraCi (Traffic Control Interface)[27]. CrowdNav consists of a number of cars traveling in a virtual city following the itineraries of their drivers and of a centralized navigation service (see 5.2). A fraction is guided by the smart navigation service, whilst the others are controlled by the standard built-in routing algorithm of SUMO.

The smart routing service is controllable via multiple hyperparameters, which are summarized in table 5.1 For instance, randomness is introduced by route randomization in order to avoid giving the same routes to a large number of drivers. The data freshness threshold determines when dynamic traffic information is regarded expired.

Figure 5.2: The graphical user interface of CrowdNav: Cars are driving in a virtual city, some of them are guided by the smart navigation system (red).s

Name	Description	Range
route randomization	degree to which random noise is introduced introduced to avoid giving same routes	$[0, 0.3]$
exploration percentage	controls the ratio of smart cars used as explorers	$[0, 0.3]$
static info weight	controls the importance of static information on routing	$[1, 2.5]$
dynamic info weight	controls the importance of dynamic information on routing	$[1, 2.5]$
exploration weight	controls the degree of exploration of the explorers	$[5, 20]$
data freshness threshold	threshold for considering observed traffic-related data as stale and disregard it	$[100, 700]$
re-routing requency	controls how often the route would be invoked to re-route a smart car	$[10, 70]$

Table 5.1: CrowdNav hyperparameter space.

The goal is to optimize the quality of CrowdNav with respect to the average trip overhead. The overhead for each trip is defined as

$$y = \frac{\text{actual duration of a trip}}{\text{theoretical duration of a trip}} \geq 1. \tag{5.1}$$

The system has the character of a *noisy, expensive black-box*: the simulation is complex and the relationship of input parameters $\boldsymbol{x}$ and trip overhead y is not known in advance. Further, each simulation has to run some time which makes evaluations expensive. Ultimately, function evaluations are noisy: the trip overhead doesn't only depend on the route proposed by the router, which itself has stochastic components (e. g. the rout randomization factor), but particularly on other cars and the traffic which is generated stochastically. Noise can be smoothed out the longer we run the simulation.

This application raises again the question of how much budget in terms of simulation runtime should be spent on each configuration.

For observing data produced by CrowdNav for different configurations, we use the Real-Time Experimentation (RTX) tool [23]. In this thesis, existing work by Gerostathopoulos et al. [10] was extended by the integration of SMBO with intensification, which is employed to find the optimal hyperparameters for the *smart* navigation system. As surrogate model, a random forest is fitted on the non-aggregated data as random forests turned out to be more stable in the scope of this application. New points were proposed by the lower confidence bound criterion. In order to get good results in a reasonable amount of time we tune over two hyperparameters only: the route randomization factor and the data freshness threshold. In this example, one evaluation corresponds to one trip. We use an initial Latin hypercube design of size 10. Every time a new parametric configuration is set in the router, the first 2000 trips are omitted from analysis as it takes some time until the system adapts (burn-in rate). Every configuration is evaluated 1000 times. After 80 optimization iterations the identification phase is started with $b_{OCBA} = 10000$ evaluations. The algorithm terminates, if a minimum $P(CS)$ of 0.7 or a maximum number of 20 identification iterations is reached.

Figure 5.3 shows the 20 best configurations and the 90% percent confidence intervals around the mean trip overhead for the corresponding configuration. Though it seems that the identification step in the end allows a clear distinction between the best point and inferior points, this is questionable when looking at the configurations in the 2-dimensional hyperparameter space (see figure 5.4). We see that good points are very close which can have two reasons: either the effects observed are random and appear because noise has not been smoothed out sufficiently or the true underlying mechanism

in fact very unsmooth. In future work, it still needs to be clarified if higher-level hyperparameters like the burn-in rate or the initial evaluation budget of 1000 are sufficient to capture the noise level. The performance of sequential-based optimization in this application needs to be validated in a more extensive experimental study.

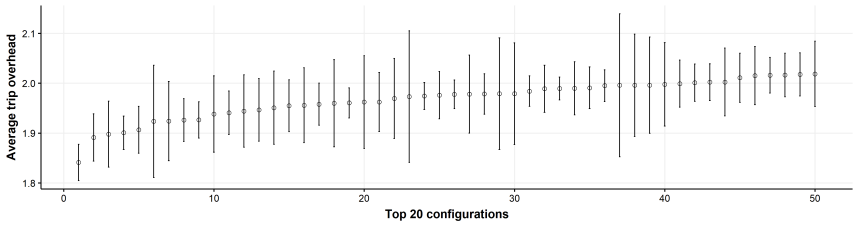


Figure 5.3: Resulting top 20 configurations after using SMBO WITH IDENTIFICATION as optimizer for CROWDNAV. The figure shows the average trip overhead for the top 20 configurations. 90% confidence intervals for $\bar{y}^{(i)}$ were calculated based on assuming normality of simulation outputs.

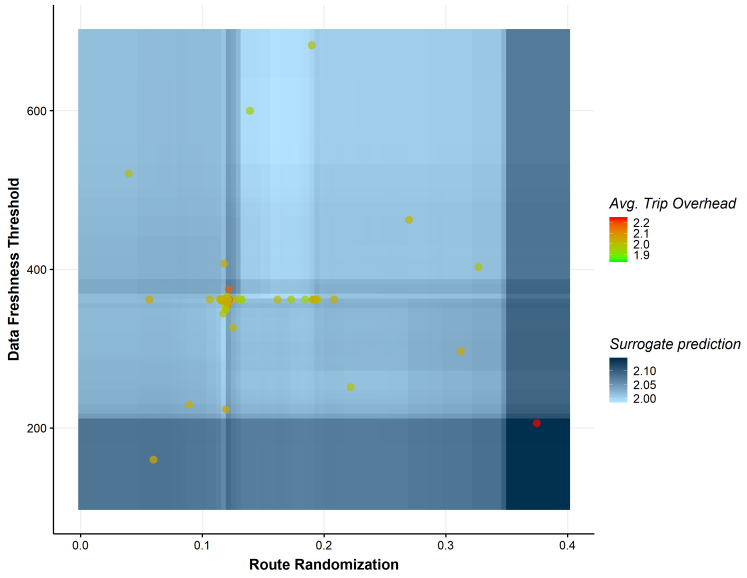


Figure 5.4: The two-dimensional hyperparameter space that was tuned over in this application and the configurations evaluated in the course of hyperparameter optimization.

Chapter 6

Conclusion

In this work we formulated the general SMBO WITH INTENSIFICATION algorithm which extends sequential model-based optimization by a replication strategy. We have identified four replication strategies from literature: the simple *fixed* strategy, replication budget allocation through *ocba* and *inc* and *inc+* that iteratively perform races against the *incumbent* configuration.

In an experimental study, the replication strategies have been compared: whereas experiments on artificial test functions have shown that replication strategies can improve the optimization result considerably in the 20-dimensional case, the SKO algorithm that employs no replication strategy at all could not been outperformed by those methods employing replication. Based on the benchmark on artificial test functions, the *ocba* replication strategy together with a noisy Gaussian process surrogate learner was the best performing instance of SMBO WITH INTENSIFICATION.

Results of deploying SMBO WITH INTENSIFICATION in machine learning tuning problems suggest that replication strategies can improve the tuning outcome, but results vary between datasets and thus do not allow clear conclusions or recommendations for practical purposes. In future work, these methods will be investigated on more machine learning problems.

From experiments performed on the artificial test functions, we analysed the final error made by the respective optimization algorithms in more depth: often, a fraction of the overall error is attributable to false identification of the final best point in the end. Motivated by the OCBA theory, we proposed a new version of sequential model-based optimization that tackles this type of error in a final identification step and provides the user with a sort of confidence in this returned solution: the probability of correct selection. While a first experimental investigation of the performance of this algorithm for optimization of artificial test functions and in machine learning hyperparameter tuning yields promising results, this method needs to be evaluated in more depth in future work.

We emphasize that the point in time, when optimization is stopped, and the identification phase is started has an impact on the final outcome. In our experiments, we simply defined a budget for the optimization phase. When it was exceeded, the identification phase was started. Two suboptimal scenarios can occur: First, the optimization phase

is quit too early and the identification phase is triggered when the function was not optimized sufficiently. In this case, multiple suboptimal configurations are compared and the identification phase won't yield useable results. Second, the optimization phase takes too long. In this case, the identification phase could have been entered earlier and budget could have been saved. In future work, we will investigate how to find this "sweet spot" of switching from optimization to identification. Furthermore, we might look into in better notions of confidences in solutions.

Finally, we employed the new proposed method on a black-box optimization problem for parametric traffic simulation optimization to tackle the question of how much simulation budget should be spent on single configurations. In future work, this we will analyse the functioning of SMBO WITH IDENTIFICATION in a more extensive benchmark study.

Appendix A

Optimal Computing Budget Allocation

Assumption A.1 (Normal simulation outputs)
For inputs $x^{(i)}, i = 1, ..., n$, simulation outputs $y(\boldsymbol{x}^{(i)})$ are normally distributed around their true function value $f(\boldsymbol{x}^{(i)})$, with known variance $\tau_i^2 := \tau^2(\boldsymbol{x}^{(i)})$, i. e.

$$y(\boldsymbol{x}^{(i)}) \sim \mathcal{N}\left(f(\boldsymbol{x}^{(i)}), \tau_i^2\right).$$

For each input $x^{(i)}$, outputs are drawn independently. Furthermore, outputs are independent across designs, that is $y(\boldsymbol{x}^{(i)})$ and $y(\boldsymbol{x}^{(j)})$ are independent for all $i \neq j$.

Assumption A.2 (Conjugate uninformative normal prior)
The function values $f(\boldsymbol{x}^{(i)})$, $i = 1, ..., n$, are independent and have the conjugate normal prior distribution with mean η_i and variance $\nu_i^2 > 0$, i. e.

$$f(\boldsymbol{x}^{(i)}) \sim \mathcal{N}\left(\eta_i, \nu_i^2\right).$$

Furthermore, the prior is assumed uninformative, i. e. $\nu_i^2 \to \infty$.

Lemma A.3
Under assumptions A.1 and A.2, the posterior distribution is

$$f(\boldsymbol{x}^{(i)}) \mid D \sim \mathcal{N}\left(\frac{\tau_i^2\eta_i + r^{(i)}\nu_i^2\bar{y}^{(i)}}{\tau_i^2 + r^{(i)}\nu_i^2}, \frac{\tau_i^2\nu_i^2}{\tau_i^2 + r^{(i)}\nu_i^2}\right)$$

For $\nu_i^2 \to \infty$, this becomes

$$f(\boldsymbol{x}^{(i)}) \mid D \overset{a}{\sim} \mathcal{N}\left(\bar{y}^{(i)}, \frac{\tau_i^2}{r^{(i)}}\right) \quad \text{for } \nu_i \to \infty.$$

Definition A.4
The approximate probability of correct selection is defined as

$$APCS = 1 - \sum_{i=1}^{n} P\left(f(\boldsymbol{x}^{(inc)}) > f(\boldsymbol{x}^{(i)}) \mid D\right)$$

Problem A.5 (OCBA-APCS Problem)

$$\max_{r^{(1)},\ldots,r^{(k)}} APCS$$
$$\text{s. t. } r^{(1)} + r^{(2)} + \ldots r^{(n)} = r$$

Theorem A.6
Given a total number of simulation samples r to be allocated to n competing designs whose performance is depicted by random variables with means $f(\boldsymbol{x}^{(i)}), i = 1, ..., n$, and finite variances τ_i^2 respectively, the Approximate Probability of Correct Selection (APCS) is asymptotically maximized when

$$\begin{aligned} \frac{r^{(i)}}{r^{(j)}} &= \left(\frac{\tau_i / \delta_{inc,i}}{\tau_j / \delta_{inc,j}} \right)^2, \qquad i, j \in \{1, 2, ..., m\}, i \neq j \neq inc \\ n^{(inc)} &= \tau_{inc} \sqrt{\sum_{i=1, i \neq inc} \frac{(r^{(i)})^2}{\tau_i^2}}. \end{aligned} \tag{A.1}$$

Appendix B

Implementation in `mlrMBO`

All algorithms, experiments and graphics have been implemented in `R` [20]. Only the experiments for the CrowdNav example have been integrated using `python` as programming language. The intensification strategies presented in Chapter 3 are implementend in the `R` package `mlrMBO` [4] and can be accessed under the development branch `https://github.com/mlr-org/mlrMBO/tree/feature_noisy_incumbent_ocba`. The benchmark experiments have been implemented with the use of the `R` package `batchtools` [16]. For graphics, the `R` package `ggplot2` [28] was used.

Code, results, graphics and the thesis are accessible via the bitbucket repository `https://bitbucket.org/ju_moosbauer/thesis-moosbauer`.

- `/benchmarks/` contains all code for the execution and analysis of the experiments
- `/RTX/` contains the source code of RTX; the contribution of this thesis can be found under `/crowdnav-rtx/RTX/rtxlib/executionstrategy/MlrStrategy.py` (for installation of CrowdNav and RTX, please refer to the documentations under `https://github.com/Starofall/CrowdNav` and `https://github.com/Starofall/RTX`)
- `/result/` contains all benchmark results and respective graphics
- `/thesis/` contains the this thesis in `.tex` and `.pdf` format
- `/mlrMBO/` contains the current version of the `feature_noisy_incumbent_ocba` branch of the `R` package `mlrMBO`

Appendix C

Detailed Benchmark Results

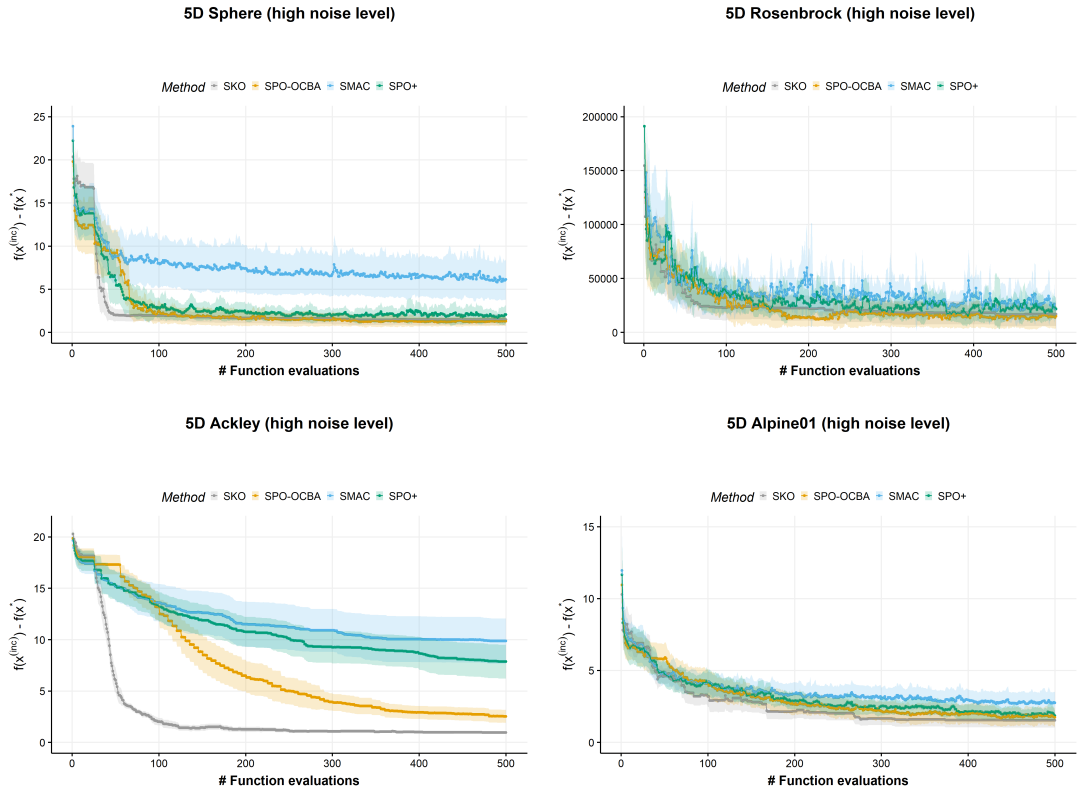
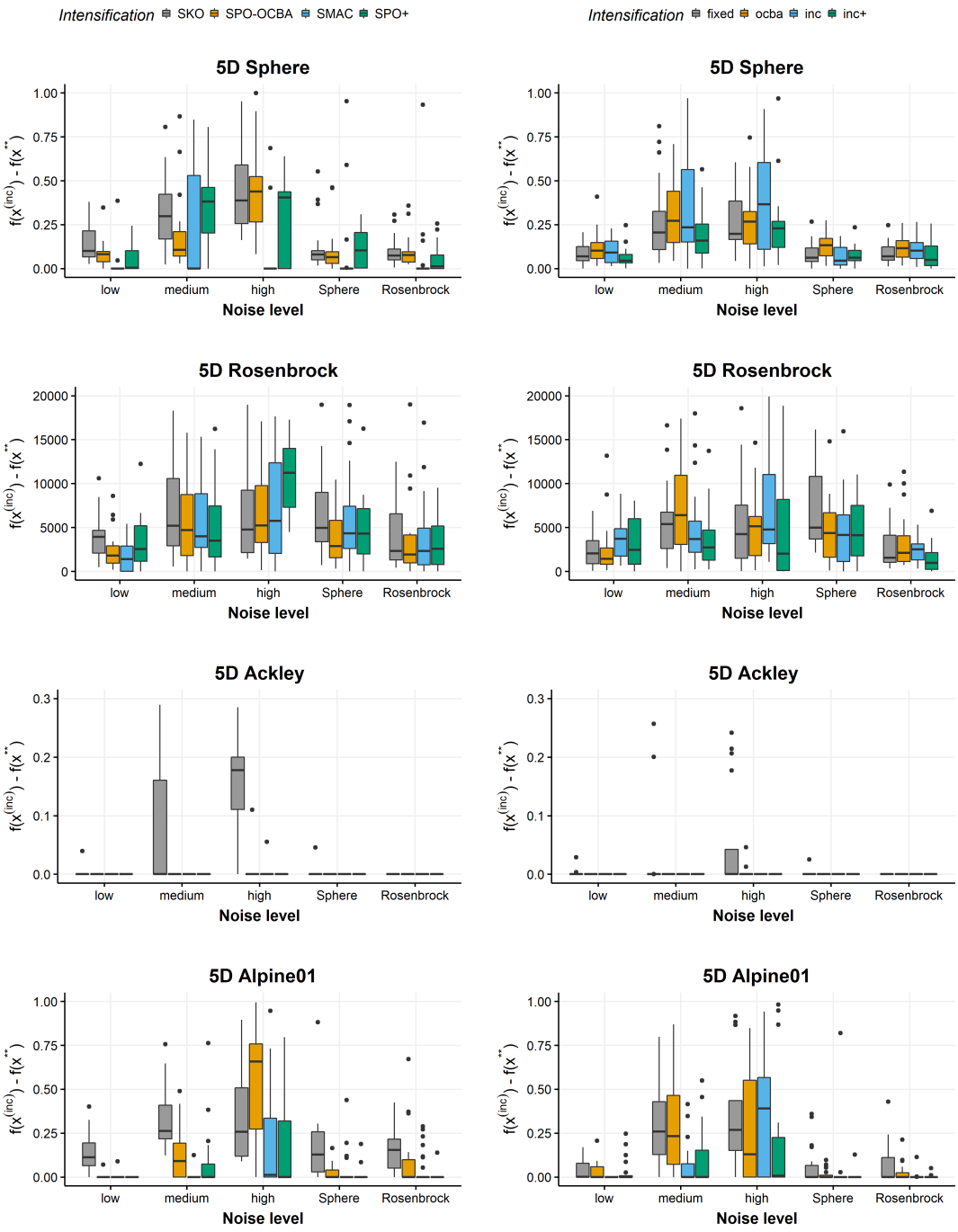
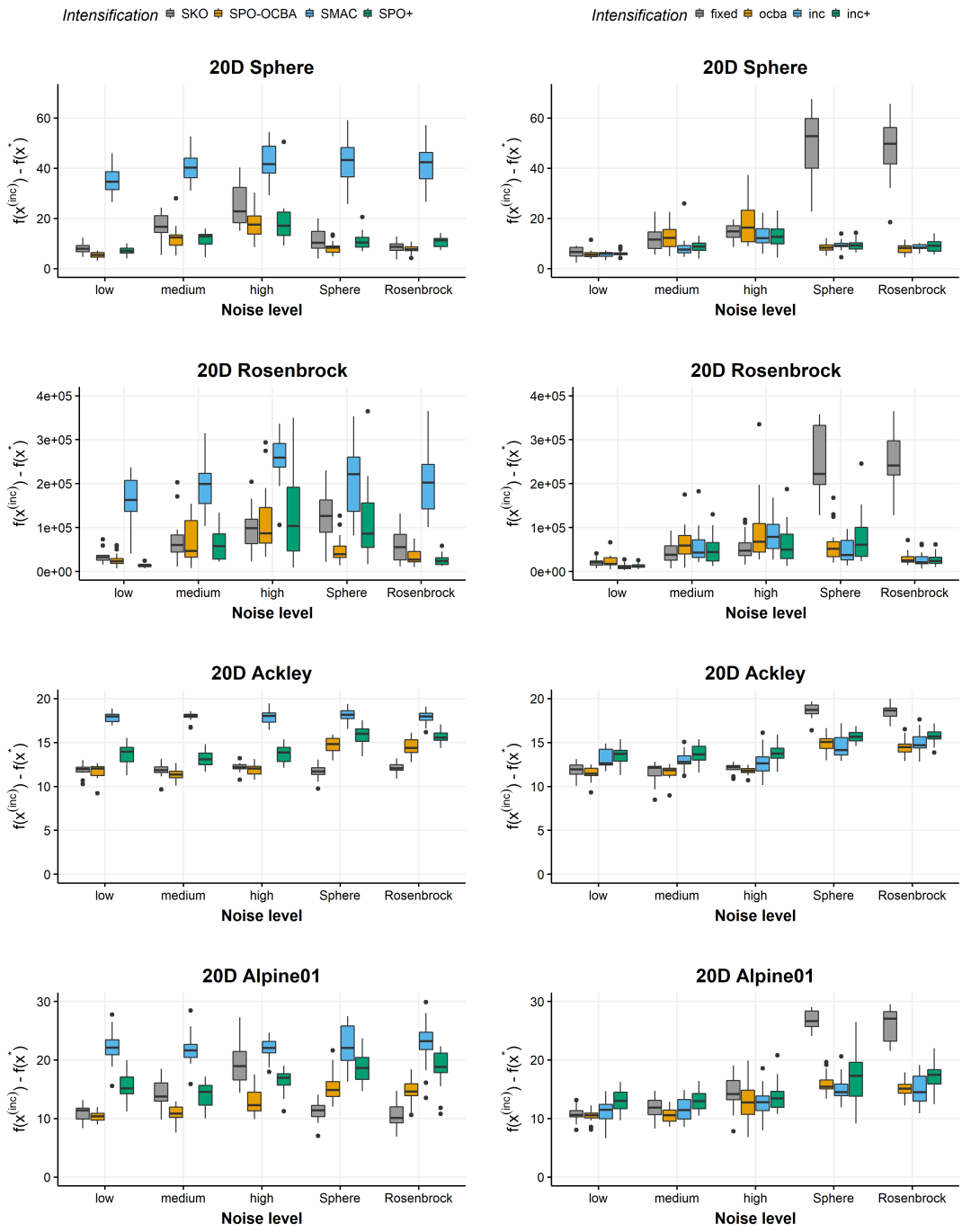


Figure C.1: Aggregated optimization paths of the original algorithms (see table 4.2) on experiments with the $5D$ test functions with high noise levels. The figure shows the mean error accross the experiments (line) and the corresponding 90-percent confidence band.



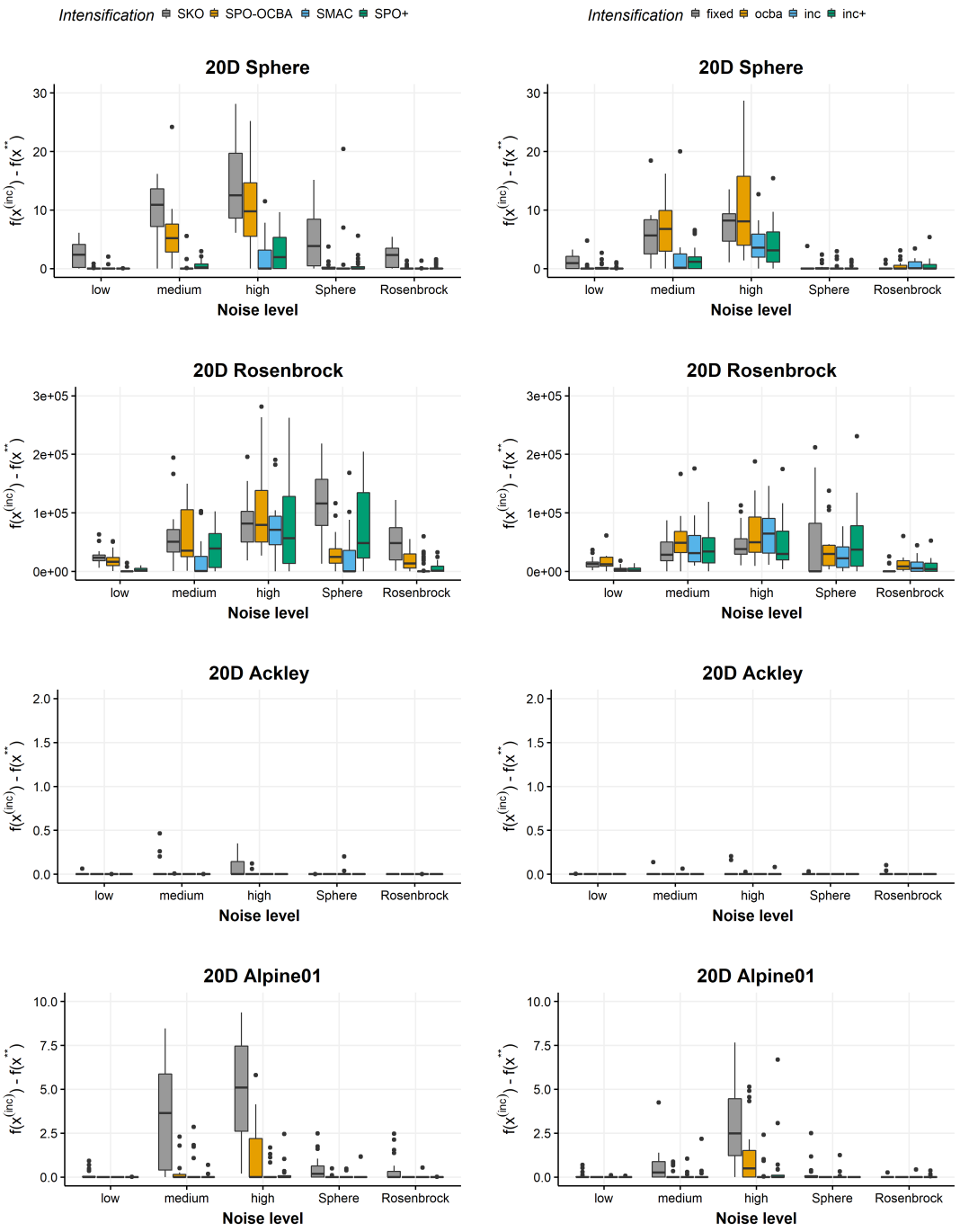
(a) Original methods (see table 4.2) **(b)** Optimized strategies (see table 4.6)

Figure C.2: Identification error $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^{**})$ for the experiments on the $5D$ test functions.



(a) Original methods (see table 4.2) **(b)** Optimized strategies (see table 4.6)

Figure C.3: Overall error $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^*)$ for the experiments on the $20D$ test functions.



(a) Original methods (see table 4.2)

(b) Optimized strategies (see table 4.6)

Figure C.4: Identification error $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^{**})$ for the experiments on the $20D$ test functions.

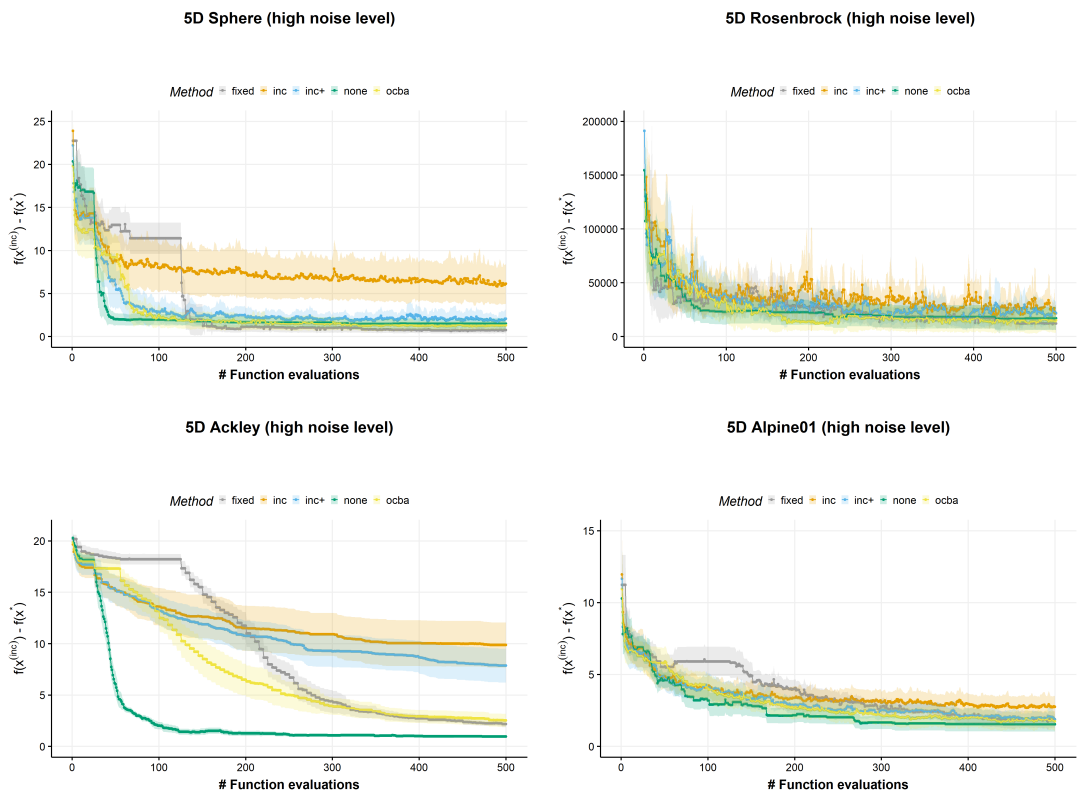


Figure C.5: Aggregated optimization paths of the different intensification strategies in an improved configuration (see table 4.6) on experiments with the $5D$ test functions with high noise levels. The figure shows the mean error accross the experiments (line) and the corresponding 90-percent confidence band.

		intensification	surr.	crit.	agg.	rank
SKO		**none**	**km.nugget**	**aei**	**none**	**18.46**
	*	**ocba**	**km.nugget**	**cb**	**mean**	**19.41**
		ocba	km.nugget	eqi	mean	20.04
		fixed	km.nugget	aei	mean	20.90
	*	**fixed**	**km.nugget**	**cb**	**mean**	**21.25**
		ocba	km.nugget	aei	mean	22.53
		ocba	km.nugget	ei	mean	22.78
		fixed	km.nugget	ei	mean	23.35
		fixed	km.nugget	eqi	mean	25.50
		ocba	km	cb	mean	28.59
		ocba	km.nugget	mr	none	31.79
		ocba	km	ei	mean	33.57
		ocba	km	aei	mean	33.59
		ocba	km	eqi	mean	33.87
		ocba	km	eqi	none	34.59
		fixed	km	cb	mean	36.00
		inc	km.nugget	aei	mean	36.00
		ocba	km.nugget	cb	none	36.35
		fixed	km	aei	mean	37.33
		inc	km.nugget	eqi	mean	37.62
	*	**inc**	**km.nugget**	**cb**	**mean**	**37.94**
		ocba	km	cb	none	38.10
		inc	km.nugget	ei	mean	38.39
		ocba	km.nugget	ei	none	38.56
		ocba	km	aei	none	39.34
		fixed	km	eqi	mean	39.41
		fixed	km.nugget	cb	none	40.10
		fixed	km	ei	mean	40.15
SPO-OCBA		**ocba**	**km**	**mr**	**none**	**40.31**
		inc	km.nugget	cb	none	41.40
		inc+	km.nugget	eqi	mean	42.07
	*	**inc+**	**km.nugget**	**cb**	**mean**	**42.43**
		ocba	km	ei	none	42.49
		fixed	km.nugget	ei	none	42.79
		ocba	km.nugget	aei	none	42.86
		inc	km.nugget	ei	none	42.97
		inc+	km.nugget	ei	mean	43.07
		fixed	km	cb	none	43.29
		inc+	km.nugget	aei	mean	45.22
		inc	km	cb	mean	45.22
		ocba	km.nugget	eqi	none	45.36
		inc	km	cb	none	45.67
		inc	km	ei	mean	46.10
		inc	km.nugget	aei	none	46.54
		fixed	km.nugget	aei	none	46.82
		inc	km	aei	mean	47.10
		inc	km	ei	none	47.41
		inc	km	eqi	mean	47.43
		inc	km	eqi	none	47.45
SMAC		**inc**	**rf**	**ei**	**none**	**47.70**
		⋮	⋮	⋮	⋮	⋮
		⋮	⋮	⋮	⋮	⋮

	intensification	surr.	crit.	agg.	rank
	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮
	fixed	km	eqi	none	47.71
	inc	km	aei	none	48.32
	ocba	rf	cb	none	48.72
	fixed	km	aei	none	48.88
	fixed	km	ei	none	49.19
	inc	km.nugget	eqi	none	49.67
	fixed	km.nugget	eqi	none	50.78
	inc+	km	cb	none	50.89
	ocba	rf	cb	mean	51.58
	inc+	km.nugget	cb	none	51.61
	inc+	km	aei	none	51.88
	inc+	km	eqi	mean	52.29
SPO+	**inc+**	**km**	**ei**	**mean**	**52.51**
	inc+	km	aei	mean	52.66
	inc+	km.nugget	ei	none	52.68
	inc+	km	ei	none	53.34
	inc+	km	cb	mean	53.44
	ocba	rf	mr	none	54.70
	inc+	km.nugget	aei	none	54.78
	inc+	km	eqi	none	55.39
	fixed	rf	cb	none	55.68
	inc	rf	cb	none	55.90
	ocba	rf	eqi	mean	56.84
	fixed	rf	cb	mean	57.13
	inc	rf	cb	mean	57.88
	ocba	rf	eqi	none	59.47
	ocba	rf	aei	mean	60.08
	inc+	km.nugget	eqi	none	60.19
	ocba	rf	ei	none	60.35
	ocba	rf	aei	none	60.75
	ocba	rf	ei	mean	61.60
	fixed	rf	aei	none	63.18
	fixed	rf	ei	none	64.45
	fixed	rf	ei	mean	64.97
	fixed	rf	eqi	none	65.10
	inc	rf	eqi	none	65.94
	fixed	rf	eqi	mean	65.97
	fixed	rf	aei	mean	65.98
	inc	rf	aei	none	66.51
	inc	rf	ei	mean	67.22
	inc	rf	aei	mean	67.49
	inc	rf	eqi	mean	68.28
	inc+	rf	cb	mean	70.08
	inc+	rf	cb	none	70.69
	inc+	rf	aei	none	70.97
	inc+	rf	aei	mean	71.85
	inc+	rf	ei	mean	72.01
	inc+	rf	eqi	none	72.52
	inc+	rf	ei	none	73.22
	inc+	rf	eqi	mean	75.04

Table C.1: Average ranks for the overall error on the $5D$ test functions for the different algorithm versions. Results were ranked in each replication and then averaged over the replications and problem instances. Maximum rank is thus $n_{repl} \times n_{surr} \times n_{crit} \times n_{agg} + 2 = 4 \times 3 \times 4 \times 2 + 2 = 98$ (+2 for original versions of SKO and SPO-OCBA results).

	intensification	surr.	crit.	agg.	rank
*	**ocba**	**km.nugget**	**cb**	**mean**	**12.19**
	ocba	km.nugget	mr	mean	12.60
*	**inc**	**km.nugget**	**cb**	**mean**	**13.85**
	ocba	km.nugget	aei	mean	14.16
	ocba	km.nugget	ei	mean	15.12
	ocba	km.nugget	eqi	mean	15.16
	inc	km.nugget	ei	mean	15.50
	none	km.nugget	aei	mean	15.65
	inc	km.nugget	aei	mean	16.00
	none	km.nugget	cb	mean	16.88
	inc	km.nugget	eqi	mean	17.10
	ocba	km	cb	mean	17.70
*	**inc+**	**km.nugget**	**cb**	**mean**	**18.33**
	inc	km	cb	mean	18.55
	inc	km	ei	mean	18.62
	ocba	km	ei	mean	18.88
	inc+	km.nugget	ei	mean	19.22
	ocba	km	eqi	mean	19.23
	inc	km	aei	mean	19.33
	ocba	km	aei	mean	19.34
	inc+	km.nugget	aei	mean	19.88
	inc+	km.nugget	eqi	mean	20.10
	inc	km	eqi	mean	20.36
	none	km.nugget	eqi	mean	20.93
	inc+	km	cb	mean	21.99
	inc+	km	ei	mean	22.12
	inc+	km	aei	mean	22.37
	inc+	km	eqi	mean	23.45
	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮

	intensification	surr.	crit.	agg.	rank
	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮
	none	km	cb	mean	23.61
*	**fixed**	**km.nugget**	**cb**	**mean**	**24.94**
	fixed	km.nugget	aei	mean	25.79
	fixed	km.nugget	ei	mean	26.09
	fixed	km	cb	mean	27.22
	inc	rf	cb	mean	27.40
	fixed	km.nugget	eqi	mean	27.63
	none	km	eqi	mean	27.71
	ocba	rf	cb	mean	28.66
	fixed	km	eqi	mean	29.26
	fixed	km	ei	mean	29.48
	inc	rf	eqi	mean	30.05
	fixed	km	aei	mean	30.11
	inc+	rf	cb	mean	30.28
	ocba	rf	eqi	mean	31.59
	inc	rf	aei	mean	32.17
	fixed	rf	cb	mean	32.40
	inc	rf	ei	mean	33.40
	inc+	rf	eqi	mean	33.46
	ocba	rf	aei	mean	33.93
	inc+	rf	aei	mean	34.34
	ocba	rf	ei	mean	34.41
	fixed	rf	eqi	mean	34.79
	none	rf	cb	mean	35.07
	inc+	rf	ei	mean	35.90
	fixed	rf	aei	mean	35.90
	fixed	rf	ei	mean	36.82
	none	rf	eqi	mean	40.68

Table C.2: Average ranks for the overall error on the $20D$ test functions for the different algorithm versions. Results were ranked in each replication and then averaged over the replications and problem instances. Maximum rank is thus $n_{repl} \times n_{surr} \times n_{crit} \times n_{agg} + 2 = 4 \times 3 \times 4 \times 1 + 2 = 50$ (+2 for original versions of SKO and SPO-OCBA).

intensification	**surrogate**	**rank**
spoplus	km.nugget	4.77
smac	rf	4.98
ocba	km.nugget	4.99
fixed	rf	5.07
smac	km.nugget	5.10
spoplus	rf	5.26
fixed	km.nugget	5.48
none	km.nugget	5.77
ocba	rf	5.78
none	rf	5.84

Table C.3: Average ranks for the tuning results (*mmce*) of the different SMBO versions with intensification. Results were ranked in each replication and then averaged over the replications and problems.

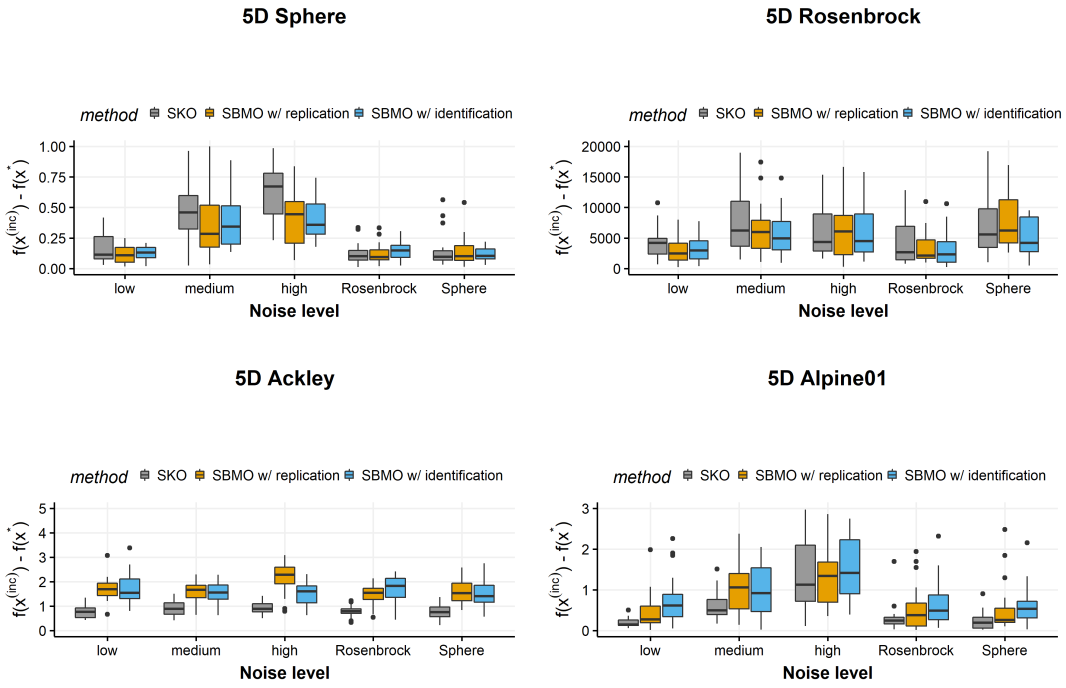


Figure C.6: Overall error $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^*)$ for the experiments on the $5D$ test functions comparing SKO, SMBO WITH REPLICATION and SMBO WITH IDENTIFICATION.

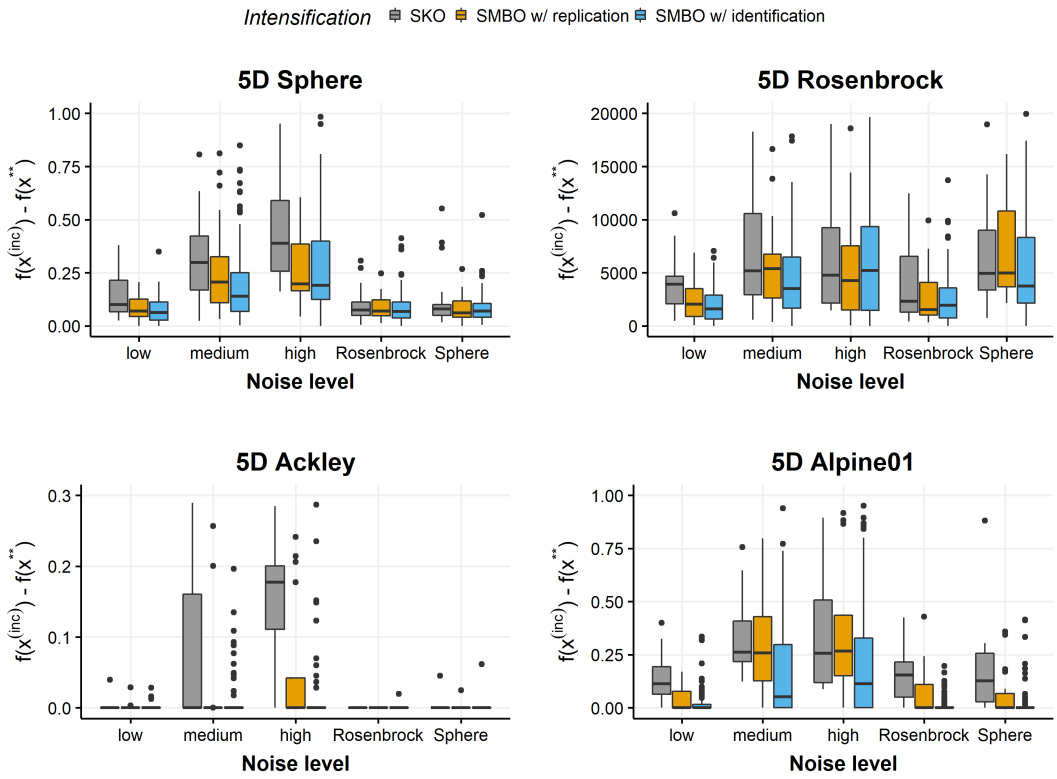


Figure C.7: Identification error $f(\boldsymbol{x}^{(inc)}) - f(\boldsymbol{x}^{**})$ for the experiments on the $5D$ test functions comparing SKO, SMBO WITH REPLICATION and SMBO WITH IDENTIFICATION.

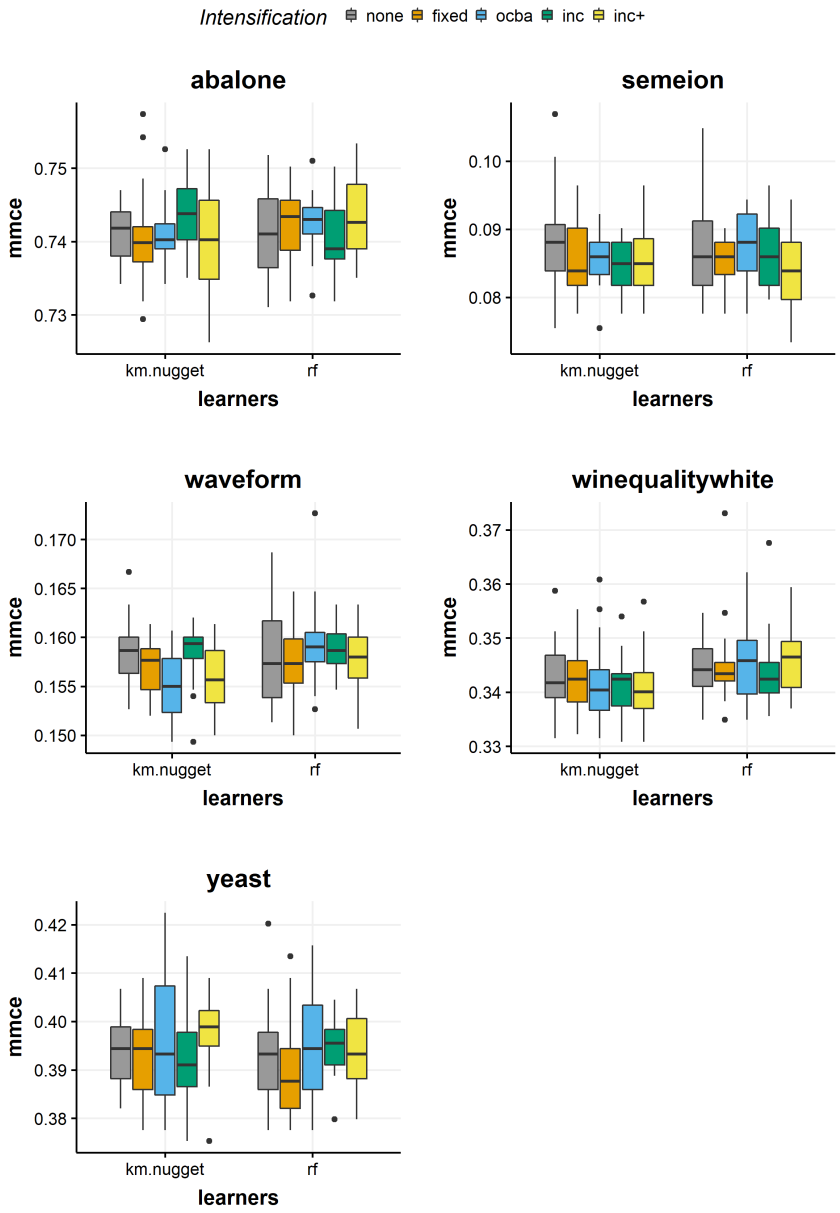


Figure C.8: Test performance (*mmce*) of the tuned `xgboost` model for the different SMBO versions on the multiclass problems.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. “Sequential Parameter Optimization”. In: vol. 1. IEEE, 2005, pp. 773–780. ISBN: 978-0-7803-9363-9.

[2] T. Bartz-Beielstein. “SPOT: An R Package For Automatic and Interactive Tuning of Optimization Algorithms by Sequential Parameter Optimization”. In: *arXiv:1006.4645 [cs, math, stat]* (June 2010). arXiv: 1006.4645 [cs, math, stat].

[3] T. Bartz-Beielstein, M. Friese, M. Zaefferer, B. Naujoks, O. Flasch, W. Konen, and P. Koch. “Noisy Optimization with Sequential Parameter Optimization and Optimal Computational Budget Allocation”. In: ACM Press, 2011, p. 119. ISBN: 978-1-4503-0690-4.

[4] B. Bischl, J. Richter, J. Bossek, D. Horn, J. Thomas, and M. Lang. “mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions”. In: *arXiv:1703.03373 [stat]* (Mar. 2017). arXiv: 1703.03373 [stat].

[5] J. Bossek. “Smoof: Single- and Multi-Objective Optimization Test Functions”. In: *The R Journal* (2017).

[6] L. Breiman. “Random Forests”. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565.

[7] C.-h. Chen and L. H. Lee. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. System engineering and operations research v. 1. OCLC: ocn456170891. Singapore ; Hackensack, NJ: World Scientific, 2011. ISBN: 978-981-4282-64-2.

[8] C.-H. Chen, J. Lin, E. Yücesan, and S. E. Chick. “Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization”. In: *Discrete Event Dynamic Systems* 10.3 (July 2000), pp. 251–270. ISSN: 0924-6703.

[9] T. Chen and C. Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. New York, NY, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2.

[10] I. Gerostathopoulos, C. Prehofer, and T. Bures. "Adapting a System with Noisy Outputs with Statistical Guarantees". In: *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*. SEAMS '18. Gothenburg, Sweden: ACM, 2018, pp. 58–68. ISBN: 978-1-4503-5715-9.

[11] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. "Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models". In: *Journal of Global Optimization* 34.3 (Mar. 2006), pp. 441–466. ISSN: 0925-5001, 1573-2916.

[12] F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Sequential Model-Based Optimization for General Algorithm Configuration". In: *Learning and Intelligent Optimization*. Ed. by C. A. C. Coello. Vol. 6683. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523. ISBN: 978-3-642-25565-6 978-3-642-25566-3.

[13] F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy. "An Experimental Investigation of Model-Based Parameter Optimisation: SPO and Beyond". In: ACM Press, 2009, p. 271. ISBN: 978-1-60558-325-9.

[14] D. R. Jones, M. Schonlau, and W. J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13.4 (Dec. 1998), pp. 455–492. ISSN: 1573-2916.

[15] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. "Recent Development and Applications of SUMO - Simulation of Urban MObility". In: *International Journal On Advances in Systems and Measurements* 5.3&4 (Dec. 2012), pp. 128–138.

[16] M. Lang, B. Bischl, and D. Surmann. "batchtools: Tools for R to work on batch systems". In: *The Journal of Open Source Software* 2.10 (Feb. 2017).

[17] A. Liaw and M. Wiener. "Classification and Regression by randomForest". In: *R News* 2.3 (2002), pp. 18–22.

[18] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle. "The Irace Package: Iterated Racing for Automatic Algorithm Configuration". In: *Operations Research Perspectives* 3 (2016), pp. 43–58. ISSN: 22147160.

[19] V. Picheny, T. Wagner, and D. Ginsbourger. "A Benchmark of Kriging-Based Infill Criteria for Noisy Optimization". In: *Structural and Multidisciplinary Optimization* 48.3 (Sept. 2013), pp. 607–626. ISSN: 1615-1488.

[20] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2014.

[21] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive computation and machine learning. OCLC: ocm61285753. Cambridge, Mass: MIT Press, 2006. ISBN: 978-0-262-18253-9.

[22] O. Roustant, D. Ginsbourger, and Y. Deville. "**DiceKriging** , **DiceOptim** : Two *R* Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization". In: *Journal of Statistical Software* 51.1 (2012). ISSN: 1548-7660.

[23] S. Schmid, I. Gerostathopoulos, C. Prehofer, and T. Bures. "Self-Adaptation Based on Big Data Analytics: A Model Problem and Tool". In: IEEE, May 2017, pp. 102–108. ISBN: 978-1-5386-1550-8.

[24] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1 (Jan. 2016), pp. 148–175. ISSN: 0018-9219, 1558-2256.

[25] J. Snoek, H. Larochelle, and R. P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *arXiv:1206.2944 [cs, stat]* (June 2012). arXiv: 1206.2944 [cs, stat].

[26] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms". In: *Proc. of KDD-2013*. 2013, pp. 847–855.

[27] A. Wegener, M. Piórkowski Michałand Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux. "TraCI: An Interface for Coupling Road Traffic and Network Simulators". In: *Proceedings of the 11th Communications and Networking Simulation Symposium*. CNS '08. Ottawa, Canada: ACM, 2008, pp. 155–163. ISBN: 1-56555-318-7.

[28] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4.