Surrogates for Hierarchical Search Spaces: The Wedge-Kernel and an Automated Analysis

Daniel Horn
TU Dortmund University
Dortmund, Germany
daniel.horn@tu-dortmund.de

Jörg Stork
TH Köln
Cologne, Germany
joerg.stork@th-koeln.de

Nils-Jannik Schüßler
TU Dortmund University
Dortmund, Germany
nilsjannik.schuessler@tu-dortmund.de

Martin Zaefferer
TH Köln
Cologne, Germany
martin.zaefferer@th-koeln.de

ABSTRACT

Optimization in hierarchical search spaces deals with variables that only have an influence on the objective function if other variables fulfill certain conditions. These types of conditions complicate the optimization process. If the objective function is expensive to evaluate, these complications are further compounded. Especially, if surrogate models are learned to replace the objective function, they have to be able to respect the hierarchical dependencies in the data. In this work a new kernel is introduced, that allows Kriging models (Gaussian process regression) to handle hierarchical search spaces. This new kernel is compared to existing kernels based on an artificial benchmark function. Thereby, we face a typical algorithm design problem: The statistical analysis of benchmark results. Instead of just identifying the best algorithm, it is often desirable to compute algorithm rankings that depend on additional experimental parameters. We propose a method for the automated analysis of such algorithm comparisons. Instead of investigating all parameter constellations of our artificial test function at once, we apply a cluster algorithm and analyze rankings of the algorithms within each cluster. This new method is used to analyze the above mentioned benchmark of kernels for hierarchical search spaces.

CCS CONCEPTS

• **Mathematics of computing → Nonparametric statistics**; • **Theory of computation → Mathematical optimization**; • **Computing methodologies → Kernel methods**; *Gaussian processes.*

KEYWORDS

surrogate model-based optimization, hierarchical search spaces, conditional variables, kernel

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6111-8/19/07.
https://doi.org/10.1145/3321707.3321765

ACM Reference Format:
Daniel Horn, Jörg Stork, Nils-Jannik Schüßler, and Martin Zaefferer. 2019. Surrogates for Hierarchical Search Spaces: The Wedge-Kernel and an Automated Analysis. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3321707.3321765

1 INTRODUCTION

This article deals with specific optimization problems that combine two core challenges: an expensive objective function and a hierarchical search space. Expensive objective functions imply that the evaluation of a candidate solution requires some significant amount of resources (e.g., time, money). One solution is to employ surrogate models that replace the expensive objective function.

Hierarchical search spaces are caused by conditions between variables: some variables only affect the objective function if other variables satisfy some condition. Hence, an objection function in a hierarchical search space can always be written as

$$f(\boldsymbol{x}) = g\left(\boldsymbol{x}^{(\mathrm{nc})}\right) + \begin{cases} h\left(\boldsymbol{x}^{(\mathrm{nc})}, \boldsymbol{x}^{(\mathrm{c})}\right) & \text{if condition holds} \\ 0 & \text{else} \end{cases},$$

where $\boldsymbol{x}^{(\mathrm{c})}$ is the vector of all hierarchical and $\boldsymbol{x}^{(\mathrm{nc})}$ the vector of all remaining variables, g is a function that only depends on $\boldsymbol{x}^{(\mathrm{nc})}$, while h depends on $\boldsymbol{x}^{(\mathrm{c})}$ and the interactions between $\boldsymbol{x}^{(\mathrm{nc})}$ and $\boldsymbol{x}^{(\mathrm{c})}$.

A typical example arises in the parameter tuning of evolutionary algorithms, as illustrated in Fig. 1. Here, the step size σ of a Gaussian mutation operator only impacts the performance, if the operator is chosen to be used in the algorithm. Hence, $\boldsymbol{x}^{(\mathrm{c})}$ consists of only σ, while $\boldsymbol{x}^{(\mathrm{nc})}$ contains both the choice of the mutation operator and the population size.

Hierarchical search spaces often introduce discontinuities in the objective function, which cannot be handled well by standard surrogate models. Hence, surrogate models that respect hierarchical dependencies in the data are required. Moreover, the hierarchical structure is often known a priori and can thus be used in the modelling process. To that end, we focus on kernel-based models, and describe a new kernel for hierarchical search spaces, the Wedge-kernel.

We outline some advantages and disadvantages of the Wedge-kernel in comparison to alternative kernels by Hutter and Osborne [9] and Zaefferer and Horn [17] based on their design and simple

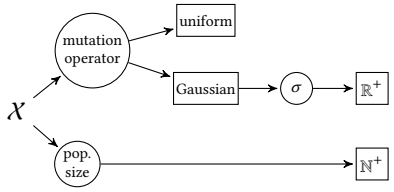

Figure 1: The search space of an algorithm tuning problem, with a graph structure to denote the hierarchical dependency.

theoretical considerations. But we need experiments to judge how such considerations impact algorithm performance. To that end, we perform experiments with an artificial benchmark function, which is taken from Zaefferer and Horn [17]. This leads to a common problem in the design and understanding of algorithms: The statistical analysis of experimental benchmark results. Quite a lot of different statistical tests and methods are used to approach this problem in current publications, while no single one prevails. Moreover, most of the used methods have at least some drawbacks. Therefore, we tackle this issue in a new way following two main ideas: Firstly, we use non-parametric multi-comparison tests to determine differences between algorithms. Secondly, we make these comparisons for specific subgroups of our experimental test instances. These groups are automatically determined with a clustering approach. The intent of these groups is to identify situations in which the algorithms behave differently.

We aim at answering the following research questions:

(1) How does the Wedge-kernel perform in comparison to alternative kernels for hierarchical search spaces?
(2) Does the automated analysis give meaningful additional information?
(3) How does the automated analysis differ from the analysis based on theoretical groups by Zaefferer and Horn [17]?

This paper is organized as follows: In Sections 2 and 3, introductions to model-based optimization and to Kriging models are given. Section 4 presents existing kernels for hierarchical search spaces as well as the proposed Wedge-kernel. In Section 5, the experimental set-up is given. Section 6 introduces the proposed method for the automated analysis of benchmark results. In Section 7, the results from the experiments are described with the methods from Section 6. Finally, the paper concludes in Section 8.

2 SURROGATE MODEL-BASED OPTIMIZATION

Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a real-valued black-box function, i.e. no information on f is available except for the shape of its parameter space $\mathcal{X} = \mathcal{X}^{(1)} \times \ldots \times \mathcal{X}^{(d)} \subset \mathbb{R}^d$. For each variable x_i, box constraints are given, i.e. $\mathcal{X}^{(i)} = [l_i, u_i]$. In optimization, the goal is to find a vector $\tilde{\boldsymbol{x}} \in \mathcal{X}$ with $\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) = \left\{\boldsymbol{x} | f(\boldsymbol{x}') > f(\boldsymbol{x}) \ \forall \boldsymbol{x}' \in \mathcal{X}\right\}$. Further, it is assumed that f is expensive to evaluate. Hence, only a small budget of function evaluations is available. In this situation, surrogate model-based optimization (SMBO) algorithms have been shown to be effective.

In SMBO, f is substituted by a surrogate model $\hat{f}$. At first, SMBO evaluates an initial design in order to fit an initial surrogate model $\hat{f}$. Afterwards, $\hat{f}$ is alternately searched for promising vectors to be evaluated with f itself and refined with the new information gained from these evaluations. Hereby, a vector $\boldsymbol{x}$ is called promising if $\hat{f}(\boldsymbol{x})$ gives a small mean prediction $\hat{y}(\boldsymbol{x})$ or a high uncertainty $\hat{s}^2(\boldsymbol{x})$. One popular SMBO algorithm is Efficient Global Optimization that was proposed by Jones et al. [10]. Details on existing SMBO variations are given, e.g., by Bischl et al. [1].

In this paper, Latin Hypercube Sampling (LHS) is used for the initial design, the Expected Improvement (EI) [11] to search for promising points and Differential Evolution (DE) to optimize the EI. These are common choices in SMBO. The central aspect of SMBO is the surrogate model, since its quality governs the success of the optimization process. Therefore, a specific model is described next.

3 KRIGING

Many SMBO algorithms use Kriging (also called Gaussian process regression) to model the collected data. A detailed description of Kriging in the context of SMBO is given by Forrester et al. [6]. Importantly, Kriging incorporates data based on a correlation function, or kernel. One standard kernel is the exponential kernel $\mathrm{k}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\theta \cdot \mathrm{d}(\boldsymbol{x}, \boldsymbol{x}'))$. Here, $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$ are data samples, $\theta \in \mathbb{R}^+$ a kernel parameter, and $\mathrm{d}(.,.)$ a distance function, e.g., the Euclidean distance $\mathrm{d}_{\mathrm{Euc}}(.,.)$. The Kriging predictor is:

$$\hat{y}(\boldsymbol{x}) = \hat{\mu} + \boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{K}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}),$$

where the kernel matrix $\boldsymbol{K}$ contains the correlations between all training samples $\boldsymbol{X}$, $\mathbf{y}$ are the corresponding observations, $\hat{\mu}$ is the estimate of the mean of the Gaussian process, $\mathbf{1}$ is a vector of ones and $\boldsymbol{k}(\boldsymbol{x})$ is a vector of correlations between training samples $\boldsymbol{X}$ and the predicted sample $\boldsymbol{x}$.

A key distinction from other models is that Kriging may not only provide a predicted value $\hat{y}(\boldsymbol{x})$, it also provides an estimate of the variance, or uncertainty of that prediction, with

$$\hat{s}^2(\boldsymbol{x}) = \hat{\sigma}^2 \left(1 - \boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{K}^{-1} \boldsymbol{k}(\boldsymbol{x})\right),$$

where $\hat{\sigma}$ is the process variance. The parameters of the kernel function, as well as $\hat{\mu}$ and $\hat{\sigma}$ are determined by Maximum Likelihood Estimation (MLE). The uncertainty estimate $\hat{s}^2(\boldsymbol{x})$ can be employed to balance exploration and exploitation in SMBO algorithms. It may push the search into regions that are promising (good predicted value $\hat{y}(\boldsymbol{x})$) yet not well explored (large uncertainty $\hat{s}^2(\boldsymbol{x})$). This is usually achieved by computing infill criteria that incorporate both values. Infill criteria are used to decide which candidate solution is currently the most promising to be evaluated by the algorithm. The EI of candidate solutions is a frequently employed infill criterion, e.g., in the Efficient Global Optimization (EGO) algorithm [10].

Due to that, we also employed Kriging in the experiments described in this paper. However, all the introduced kernels could as well be used with other kernel-based models, such as support vector machines or radial basis function networks.

4 KERNELS FOR HIERARCHICAL SEARCH SPACES

A standard Kriging model does not incorporate information about a variable's activity state. The idea of this and previous articles [9, 13, 17] is to integrate that knowledge into the kernel function, thus producing better models and improved optimization performance.

Like Hutter and Osborne [9], we use the function $\delta_i(\boldsymbol{x})$ to denote the condition under which the i-th variable becomes active: if x_i is inactive then $\delta_i(\boldsymbol{x}) = \text{false}$, or else, if x_i is active then $\delta_i(\boldsymbol{x}) = \text{true}$. In the following, when we talk about kernels, we will always imply a kernel of the form $\mathrm{k}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\sum_{i=1}^{d} \mathrm{d}_i(\boldsymbol{x}, \boldsymbol{x}'))$. The different kernels are distinguished only by their choice of distance $\mathrm{d}_i(.,.)$. All configurable parameters of the various choices for $\mathrm{d}_i(.,.)$ will be determined by MLE. In the most simple case, we refer to the Gauss kernel as the standard kernel (*Stan-kernel*), with $\mathrm{d}_i(\boldsymbol{x}, \boldsymbol{x}') = \theta_i |x_i - x_i'|$. Note, that a value for x_i is available even if $\delta_i(\boldsymbol{x}) = \text{false}$ holds, it just does not have an impact on f. Hence, $\mathrm{d}_i(\boldsymbol{x}, \boldsymbol{x}')$ can be computed if $\delta_i(\boldsymbol{x}) = \text{false}$ holds.

4.1 The Arc-kernel

According to Hutter and Osborne [9], a kernel for hierarchical search spaces should respect that:

a) If the i-th variable is inactive in both samples $\boldsymbol{x}, \boldsymbol{x}'$, that is, $\delta_i(\boldsymbol{x}) = \delta_i(\boldsymbol{x}') = \text{false}$, then the distance in that dimension should be zero. With respect to dimension i, both samples have the same influence on the objective function: none.
b) If the i-th variable is active in both samples, that is, $\delta_i(\boldsymbol{x}) = \delta_i(\boldsymbol{x}') = \text{true}$, then the distance should be a function of x_i and x_i', e.g. the euclidean distance function.
c) If the i-th variable is active in exactly one sample, that is, $\delta_i(\boldsymbol{x}) \neq \delta_i(\boldsymbol{x}')$, then the samples are incomparable in that dimension, hence the distance should be a constant.

The Arc-kernel respects these assumptions, due to a mapping function $\mathrm{h}_i(\boldsymbol{x})$, which maps each conditional variable x_i to a two-dimensional Euclidean space. All inactive samples are mapped to a single point, which guarantees a). Active samples are mapped to a two-dimensional arc around that single point. Here, a parameter $\theta_i \in \mathbb{R}^+$ defines the radius of the arc, while $\rho_i \in [0, \pi]$ defines the angle. On the arc, distances depend on the location, ensuring that b) is respected. The distance between the single point and points on the arc is a constant, the radius, which ensures c). The geometric interpretation of $\mathrm{h}_i(\boldsymbol{x})$ is shown in Fig. 2. For continuous variables $x_i \in \mathbb{R}$, the mapping function $\mathrm{h}_i(.)$ is

$$\mathrm{h}_i(\boldsymbol{x}) = \begin{cases} \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}, & \text{if } \delta_i(\boldsymbol{x}) = \text{false}, \\ \theta_i \begin{bmatrix} \sin\left(\rho_i \frac{x_i}{u_i - l_i}\right) & \cos\left(\rho_i \frac{x_i}{u_i - l_i}\right) \end{bmatrix}^{\mathrm{T}}, & \text{otherwise.} \end{cases}$$

A standard kernel can operate in the mapped space, and its distance function can be expressed with $\mathrm{d}_{\mathrm{Arc}\,i}(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{d}_{\mathrm{Euc}}(\mathrm{h}_i(\boldsymbol{x}), \mathrm{h}_i(\boldsymbol{x}'))$. Since an exponential kernel with Euclidean distance is definite, the mapping function ensures that the Arc-kernel is also definite. Definiteness is often required for models like Kriging. In the context of Kriging, definiteness is, e.g., discussed by Zaefferer and Bartz-Beielstein [16]. The distance employed in the Arc-kernel is hence

$$\mathrm{d}_{\mathrm{Arc}\,i}(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{d}_{\mathrm{Euc}}(\mathrm{h}_i(\boldsymbol{x}), \mathrm{h}_i(\boldsymbol{x}')) = \begin{cases} 0, & \text{if } \delta_i(\boldsymbol{x}) = \delta_i(\boldsymbol{x}') = \text{false}, \\ \theta_i, & \text{if } \delta_i(\boldsymbol{x}) \neq \delta_i(\boldsymbol{x}'), \\ \theta_i \sqrt{2 - 2\cos\left(\rho_i \frac{x_i - x_i'}{u_i - l_i}\right)}, & \text{if } \delta_i(\boldsymbol{x}) = \delta_i(\boldsymbol{x}') = \text{true}. \end{cases}$$

4.2 Imputation Kernel

According to point c) in Section 4.1, the Arc-kernel is based on the idea that two samples that have different activity in some variable x_i, x_i' are not directly comparable. Due to that assumption, the Arc-kernel is designed to measure a constant distance for that variable.

This idea may result into a conflict. For example, let us assume that we took three samples $\boldsymbol{x}, \boldsymbol{x}'$, and $\boldsymbol{x}''$ from the search space with $\delta_i(\boldsymbol{x}) = \text{false}$ and $\delta_i(\boldsymbol{x}') = \delta_i(\boldsymbol{x}'') = \text{true}$. Let us further assume that the inactive x_i has some arbitrary value, and $x_i' + h = x_i''$ with some perturbation h. In that case, the Arc-kernel would evaluate the distance between x_i, x_i' and x_i, x_i'' to be the same constant: $\mathrm{d}_{\mathrm{Arc}\,i}(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{d}_{\mathrm{Arc}\,i}(\boldsymbol{x}, \boldsymbol{x}'') = \theta_i$. At the same time, due to the perturbation h, the objective function values actually differ between $\boldsymbol{x}'$ and $\boldsymbol{x}''$: $f(\boldsymbol{x}') \neq f(\boldsymbol{x}'')$. This observation leads to a revision of point c) from Section 4.1:

c.2) If the i-th variable is active in exactly one sample, that is, $\delta_i(\boldsymbol{x}) \neq \delta_i(\boldsymbol{x}')$, then the distance in that dimension should depend *exclusively* on the active variable.

To that end, Zaefferer and Horn [17] proposed the Imp-kernel. The Imp-kernel compares the active variable against an *imputed* value ρ_i, i.e.,

$$\mathrm{d}_{\mathrm{Imp}\,i}(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} 0, & \text{if } \delta_i(\boldsymbol{x}) = \delta_i(\boldsymbol{x}') = \text{false} \\ \theta_i \mathrm{d}_i(\rho_i, x_i'), & \text{if } \delta_i(\boldsymbol{x}) = \text{false} \neq \delta_i(\boldsymbol{x}') \\ \theta_i \mathrm{d}_i(x_i, \rho_i), & \text{if } \delta_i(\boldsymbol{x}) = \text{true} \neq \delta_i(\boldsymbol{x}') \\ \theta_i \mathrm{d}_i(x_i, x_i'), & \text{if } \delta_i(\boldsymbol{x}) = \delta_i(\boldsymbol{x}') = \text{true}, \end{cases}$$

The kernel parameter ρ_i may have different bounds than x_i. Just as the Arc-kernel, the Imp-kernel is definite, as it can also be interpreted in terms of a mapping function $\mathrm{h}(.)$. For $\mathrm{d}_{\mathrm{Imp}\,i}(\boldsymbol{x}, \boldsymbol{x}')$, the mapping function is

$$\mathrm{h}_i(\boldsymbol{x}) = \begin{cases} x_i, & \text{if } \delta_i(\boldsymbol{x}) = \text{true} \\ \rho_i, & \text{otherwise.} \end{cases}$$

Unlike the Arc-kernel, this maps to a one-dimensional space, as depicted in Fig. 2.

4.3 Variants

Three additional variants of the above kernels are considered in the experiments, but not discussed further. More details on these variants can be found in the descriptions of Zaefferer and Horn [17]. Hence, we only briefly summarize these three kernel variants.

The Ico-kernel is a variant of the Arc-kernel, implementing the same behavior but without the mapping function $\mathrm{h}(\boldsymbol{x})$. Thus, it is not definite. The IcoCor-kernel is the Ico-kernel corrected via an eigenspectrum transformation, to mitigate the lack of definiteness. Finally, the ImpArc-kernel is a linear combination of the Imp- and Arc-kernels.

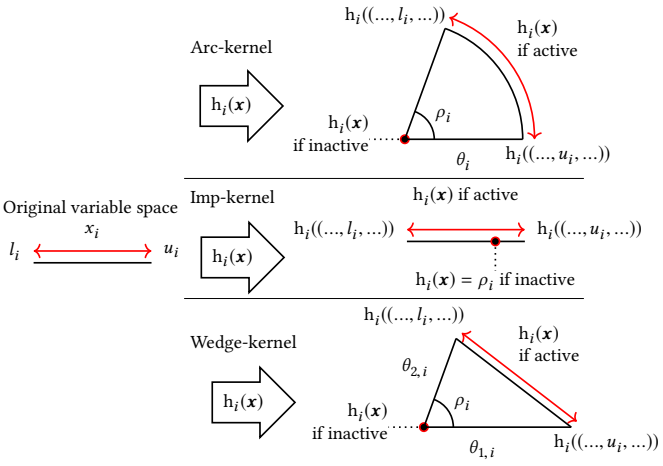

Figure 2: Overview of the different geometric interpretations of the mapping functions h, for the Arc-, Imp-, and Wedge-kernel.

Another kernel for hierarchical search spaces that we did not consider in our experiments is proposed by Hung et al. [8]. Essentially, their kernel only considers distances between hierarchical variables of two samples if the variables are active in both samples. As a consequence, their distance is zero in all other cases, whereas the distances in the Arc- or Imp-kernel are only zero when both samples are inactive. A zero-distance between an active and inactive variable can be problematic, as the overall distance will only be due to the triggering variables (i.e., the branching factors as denoted by Hung et al.).

4.4 Wedge-Kernel

As the earlier investigation by Zaefferer and Horn showed, the Imp-kernel performs well in some situations, but still has a clear disadvantage in other situations, where the Arc-kernel works better [17]. For instance, consider the case where an active sample assumes the same value as the imputed value, i.e., $\delta_i(\boldsymbol{x}) = \text{true}$ and $x_i = \rho_i$. Clearly, the distance between this active sample and an inactive sample $\boldsymbol{x}'$ ($\delta_i(\boldsymbol{x}') = \text{false}$) is zero, with $\mathrm{d}_{\mathrm{Imp}_i}(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{d}_i(\rho_i, \rho_i) = 0$. A resulting model is forced to be smooth when transitioning from active to inactive regions of the search space (when $x_i \approx \rho_i$). If this fits to the ground truth, the Imp-kernel works well. But there may as well be cases where this transition will involve a discrete jump. This jump will not be modeled well with the Imp-kernel.

Hence, we propose the Wedge-kernel (also discussed in [15]). Just like the Imp-kernel, the Wedge-kernel will evaluate a distance that depends on the active sample if it compares an active and inactive sample (see c.2) in Section 4.2). In fact, the Imp-kernel is a special case of the more general Wedge-kernel. But unlike the Imp-kernel, the Wedge-kernel can model discrete jumps, because it can enforce non-zero distances between active and inactive samples (if configured accordingly).

Essentially, the Wedge-kernel maps to a triangular shape, with

$$\mathrm{h}_i(\boldsymbol{x}) = \begin{cases} \begin{bmatrix} 0 & 0 \end{bmatrix}^\mathrm{T}, \\ \quad \text{if } \delta_i(\boldsymbol{x}) = \text{false}, \\ \begin{bmatrix} \theta_{1,i} + v\left(\theta_{2,i}\cos(\rho_i) - \theta_{1,i}\right) & v\theta_{2,i}\sin(\rho_i) \end{bmatrix}^\mathrm{T}, \\ \quad \text{otherwise}, \end{cases}$$

where $v = (x_i - l_i)/(u_i - l_i)$. The parameters $\rho_i \in [0, \pi]$, $\theta_{1,i} \in \mathbb{R}^+$, and $\theta_{2,i} \in \mathbb{R}^+$ specify the shape of the triangle (angle, and two side lengths). The Wedge-kernel simplifies to the Imp-kernel if one of the following is true: $\theta_{1,i} = 0$, $\theta_{2,i} = 0$, $\rho_i = 0$, or $\rho_i = \pi$. The geometric shape of the mapping is shown in Fig. 2, and can be compared to those of the Arc- and Imp-kernel.

With this mapping function $\mathrm{h}_i(.)$, and the squared Euclidean distance, the Wedge-kernel becomes

$$\mathrm{d}_{\mathrm{Wedge}_i}(\boldsymbol{x}, \boldsymbol{x}') = \|\mathrm{h}_i(\boldsymbol{x}) - \mathrm{h}_i(\boldsymbol{x}')\|_2^2 = \begin{cases} 0, \\ \quad \text{if } \delta_i(\boldsymbol{x}) = \delta_i(\boldsymbol{x}') = \text{false}, \\ \left(\theta_{1,i} + v'\left(\theta_{2,i}\cos(\rho_i) - \theta_{1,i}\right)\right)^2 + \left(v'\theta_{2,i}\sin(\rho_i)\right)^2, \\ \quad \text{if } \delta_i(\boldsymbol{x}) = \text{false} \neq \delta_i(\boldsymbol{x}'), \\ \left(\theta_{1,i} + v\left(\theta_{2,i}\cos(\rho_i) - \theta_{1,i}\right)\right)^2 + \left(v\theta_{2,i}\sin(\rho_i)\right)^2, \\ \quad \text{if } \delta_i(\boldsymbol{x}) = \text{true} \neq \delta_i(\boldsymbol{x}'), \\ \left(\frac{x_i - x_i'}{u_i - l_i}\right)^2 \left(\theta_{1,i}^2 + \theta_{2,i}^2 - 2\theta_{1,i}\theta_{2,i}\cos(\rho_i)\right), \\ \quad \text{if } \delta_i(\boldsymbol{x}) = \delta_i(\boldsymbol{x}') = \text{true}, \end{cases}$$

with $v' = (x_i' - l_i)/(u_i - l_i)$. As discussed above, the Wedge-kernel may avoid drawbacks of both the Arc- and the Imp-kernel. Yet, it also has a disadvantage: the triangular shape of the mapping requires an additional third parameter, which may increase the difficulty of the model training procedure. On the other hand, this is still one parameter less than the ImpArc-kernel.

5 EXPERIMENTAL SET-UP

In the experiments, the results of Zaefferer and Horn [17] are reproduced, using the same test functions, algorithms, models, and parameterizations. That is, the test function

$$f(\boldsymbol{x}) = (x_1 - d)^2 + \begin{cases} 0 & \text{if } x_1 \leq c \\ (x_2 - 0.5)^2 + b & \text{else} \end{cases}$$

is used. The optimum of the test function is mostly located at $x_1 = d$ and the parameter c controls the size of the active region of x_2. If $c > d$ holds, the optimum is located in the inactive region of x_2, as visualized in the left plot of Fig. 3. The right plot shows the contrary situation where the optimum is in the active region of x_2.

Certainly, this test function is fairly simple and does not reflect many practical test cases. The sphere it is based on can be easily modeled by a Gaussian process and, therefore, SMBO should find the global optimum within a few function evaluations. This aspect is respected in our experiments by limiting the budget to only 10 function evaluations.

In addition to its simplicity, the test function focuses on the aspect we want to study here: The influence of hierarchical search spaces. The effect of the single hierarchical variable is easy to understand and to visualize, as can be seen in Fig. 3. The behavior of c and d is easy to understand and can later easily be connected

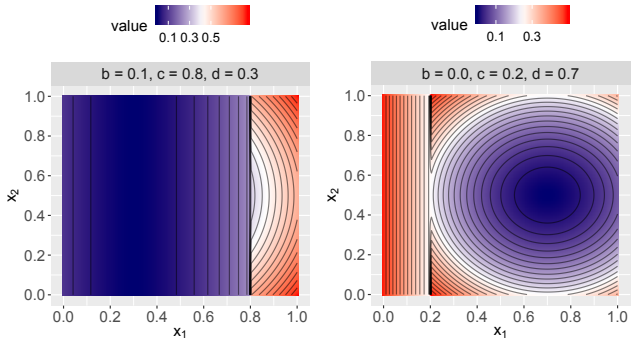

Figure 3: Illustration of two instances of the test function.

with the performances of the kernels. Moreover, the addition of the parameter b allows to investigate the effect of imputation: If $b = 0$ holds, the kernel can learn to impute $x_2 = 0.5$ for all settings with $x_1 \leq c$. Hence, the test function is well suited to understand the performances of the kernels in a hierarchical setting.

The test function is instantiated for all combinations of $b = \{0, 0.1\}$, $c = \{0.2, 0.4, 0.6, 0.8\}$, and $d = \{0.1, 0.3, 0.4, 0.7, 0.9\}$, resulting in 40 distinct test situations. In addition to the six kernels tested by Zaefferer and Horn (Arc, Ico, IcoCor, Imp, ImpArc, Stan) the Wedge-kernel is also examined. Each experiment is repeated 100 times with 100 unique random seeds (one per replication).

The remaining parameters of the optimization are taken unchanged: The budget is limited to 10 objective function evaluations, while 3 evaluations are reserved for the initial design. The Kriging model is trained with the `CEGO` R package [14] using nugget effect and re-interpolation. The EI criterion is optimized using Differential Evolution from the `DEoptim` R package [12] with 10 000 evaluations per iteration. Finally, the difference between the best found and the optimal function value is returned.

Zaeffer and Horn analyzed the results following the ideas of Demšar [4] by applying Friedman and Nemenyi tests. Moreover, they identified five disjoint subgroups of test situations and applied the analysis to each subgroup individually. The subgroups differ in two theoretical properties of the test function: Firstly, whether imputation can be profitable ($b = 0$ vs. $b \neq 0$) and secondly, the position of the optimum in the *active* or *inactive* region of x_2:

- T1: $d < c$ and $b = 0$ (imputation *profitable*, optimum *inactive*),
- T2: $d < c$ and $b > 0$ (imputation *unprofitable*, optimum *inactive*),
- T3: $d > c$ and $b = 0$ (imputation *profitable*, optimum *active*),
- T4: $d > c$, $b = 0$ and $b < (c - d)^2$ (imputation *unprofitable*, optimum *active*),
- T5: $d > c$, $b = 0$ and $b > (c - d)^2$ (imputation *unprofitable*, optimum at $x_1 = c$).

Fig. 3 shows two of these situations: T2 in the left and T3 in the right plot. It can be seen that the function in the right plot is smooth at $(d, 0.5)$, i.e. at the border of the active area, while it is a jump discontinuity at $(d, 0.5)$ in the left plot. This shows that imputation of $x_2 = 0.5$ for $x_1 < d$ leads to a smooth function if $b = 0$, while this is not possible for $b > 0$.

6 STATISTICAL ANALYSIS OF BENCHMARK RESULTS

Research in the fields of machine learning or evolutionary computation is often focused on the development of new algorithms, or the extension to existing ones. Often, such research is based on experiments that are intended to show whether a proposed algorithm variant is to some extent superior to existing algorithms. Results of such experiments have to be analyzed with statistical methods. Typically, the performance of K algorithms is compared in M different test situations and I replications are performed for each experiment, i.e. a performance value $y_k^{(m,i)}$ is available for each combination of $k = 1, ..., K$, $m = 1, ..., M$, $i = 1, ..., I$. Here, for a specific replication i, the same random seed should be used for each algorithm in order to increase comparability.

This article is a relevant example for this type of publications: The Wedge-kernel is proposed as an extension to the SMBO algorithm. On a set of $M = 40$ test situations obtained by varying three experimental parameters of a test function, the Wedge-kernel is compared with six existing SMBO variants ($K = 7$). $I = 100$ replications are performed, while the same initial design is used for each experiment of the i-th replication.

In 2006, Janez Demšar published recommendations on statistical methods for comparing algorithms over multiple data sets [4]. His approach is divided into three steps: In step one, the performance values are averaged over the replications, such that a single performance value $y_k^{(m)}$ for each combination of algorithm and test situation is available. In step two, the global hypothesis $H_0^{(glo)} : E(y_1) = ... = E(y_K)$ is checked, whether any two algorithms (also called: *groups*) differ in the test situations (also called: *blocks*). If a significant difference is detected in step two, all $\frac{K(K-1)}{2}$ post-hoc hypotheses $H_0^{(k_1,k_2)} : E(y_{k_1}) = E(y_{k_2})$ $\forall k_1, k_2 \in \{1, ..., K\}$ with $k_1 < k_2$ are checked in step three.

Demšar proposes to follow a non-parametric approach and uses the Friedman test for $H_0^{(glo)}$ and corresponding post-hoc Nemenyi tests for $H_0^{(k_1,k_2)}$. Let $r_k^{(m)} \in \{1, ..., K\}$ be the rank of the k-th algorithms in the m-th test situation with respect to $y_k^{(m)}$. The Friedman test surveys whether the average ranks $R_k = \frac{1}{M} \sum_m r_k^{(m)}$ differ too much. For $H_0^{(glo)}$ the test statistic

$$\frac{12M}{K(K+1)} \sum_{k=1}^{K} \left(R_k - \frac{K+1}{2} \right)^2$$

is asymptotically χ^2-distributed with $K - 1$ degrees of freedom, if K and M are large enough. The post-hoc Nemenyi test measures, whether the pairwise differences between the average ranks are too large. The corresponding test statistic for $H_0^{(k_1,k_2)}$

$$\sqrt{\frac{12M}{K(K+1)}} \left| R_{k_1} - R_{k_2} \right|,$$

follows a Studentized range distribution. The Nemenyi test already includes a correction for multiple testing.

The approach proposed by Janez Demšar has several disadvantages. At first, Demšar's procedure only decides which algorithms

significantly differ from each other. But the more important information would be a partial order on the algorithms. At second, the method cannot handle the I replications. Instead, it calculates an average performance value for each test situation. In this way, a good part of the variance from the original results is ignored. At third, it assumes that the algorithms behave identically over all M test situations. However, it is likely that the test situation has an influence on the algorithm performance. Hence, it is to be expected that the order of the algorithm can be diverse in different subgroups of test situations. Demšar's approach can be enhanced with respect to all three disadvantages.

Partial Order: A partial order on the algorithms is already given by the p-values of the pairwise Nemenyi tests: An algorithm k_1 is considered to be better than an algorithm k_2 if $H_0^{(k_1,k_2)}$ is rejected with respect to the test level α and $R_{k_1} < R_{k_2}$ holds. This partial order can be illustrated with a directed graph (DG). Each node in the DG corresponds to one of the algorithms in the experiment. The edge (k_1, k_2) is added to the DG if k_1 is better than k_2. In order to increase the readability of the DG, we remove edges that are represented by the transitive nature of a partial order: If the edges (k_1, k_2) and (k_2, k_3) are added to the DG, the edge (k_1, k_3) is omitted.

Using the replications: Instead of calculating mean values over the replications, they can be added as an additional variable to form the blocks for the Friedman test. Hence, ranks $r_k^{(m,i)}$ based on the original performance values $y_k^{(m,i)}$ are used. By using the same random seed in each experiment of the i-th replications it is ensured that the corresponding performance values are comparable and reasonable ranks can be computed. Afterwards, average ranks $R_k = \frac{1}{M}\sum_m \sum_i r_k^{(m,i)}$ are computed and the Friedman and Nemenyi tests are performed as usual.

Identifying subgroups of test situations: After performing the analysis on all test situation together, it can be tried do identify test situations where the algorithms behave similarly. This is a clustering problem: Finding groups such that the observations in each group differ as little as possible while the groups differ as much as possible. Here, each test situation is interpreted as an observation, the associated features are the performances of the K algorithms in each replication. Since non-parametric tests are performed afterwards, not the original performance values $y_k^{(m,i)}$ but the ranks $r_k^{(m,i)}$ are used. Hence, the vector $\left(r_1^{(m,1)}, ..., r_K^{(m,1)}, r_1^{(m,I)}, ...r_K^{(m,I)}\right)$ is used as the feature vector for the m-th test situation. Now, standard clustering methods can be used, as for example k-means clustering. Afterwards, Friedman and Nemenyi tests are applied to each found cluster and the order of algorithms is visualized by a directed graph.

The complete approach is summarized in Algorithm 1 and implemented in an R package[1]. It supports all clustering indices and methods of the `NbClust` package [2] and uses the `igraph` package [3] for visualizing the test results.

Algorithm 1 Summary of the approach for the statistical comparison of benchmark results.

Require: Performance values $y_k^{(m,i)}$ of K algorithms in M test situations with I replications.
  Calculate mean ranks $R_k = \frac{1}{M}\sum_m \sum_i r_k^{(m,i)}$
  Test the global hypothesis $H_0^{(glo)}$ using a Friedman test
  if Friedmann test is significant **then**
    Test post-hoc hypotheses $H_0^{(k_1,k_2)}$ using Nemenyi tests
    Visualize test results with an directed graph
  Identify subgroups using a cluster analysis
  for all subgroups **do**
    Test the global hypothesis $H_0^{(glo)}$ using a Friedman test
    if Friedmann test is significant **then**
      Test post-hoc hypotheses $H_0^{(k_1,k_2)}$ using Nemenyi tests
      Visualize test results with an directed graph

7 RESULTS

In this section, the test procedure described in Section 6 is applied to the results obtained from the experiments in Section 5. First, all test situations are considered. With a p-value that is close to zero (10^{-296}) the Friedman test indicates that there are performance differences between the kernels. Figure 4 shows the partial order on the kernels obtained from the Nemenyi tests. As expected, the Stan-kernel performs worst. This is consistent with [17] and shows that it is necessary to incorporate the hierarchical structure in the kernel function. The imputation-based kernels outperform kernels based on the assumptions of the arc embedding. This is probably an artifact of the test function, since imputation can be profitable in half of the test cases while differences may disappear in some of the remaining test cases. Finally, the Wedge-kernel significantly outperforms the other tested kernels.

Second, the five groups of test situations derived from the theoretical considerations by [17] are analyzed. The Friedman test shows significant differences for all subgroups. The DGs for each situation are visualized in the left column of Fig. 5. The results are mostly consistent with [17]: In situations T1 and T3 with $b = 0$, i.e. imputation could be profitable, the Imp-kernel scores the first place. This clarifies that the Imp-kernel may still be desirable to use, despite being a special case of the Wedge-kernel (e.g., if prior knowledge suggests that it fits to a certain problem). The Imp-kernel is followed by the Wedge-kernel in both situations, although the difference is only significant in T3. In situations T2 and T5 with $b > 0$, i.e. imputation should not be profitable, the Imp-kernel scores third and second to last place. Contrarily, the Wedge-kernel still reaches the second place, only superseded by the Ico- and the IcoCor-kernel

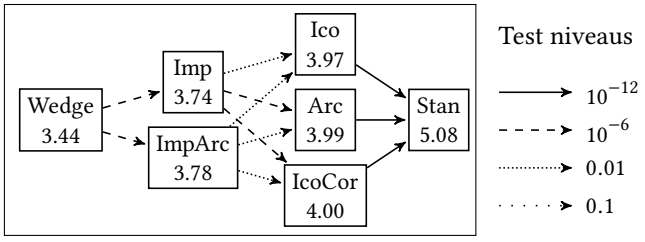

Figure 4: Directed graph showing the partial order of algorithms as well as the R_k over all test situations.

[1] see https://github.com/danielhorn/SABeR/

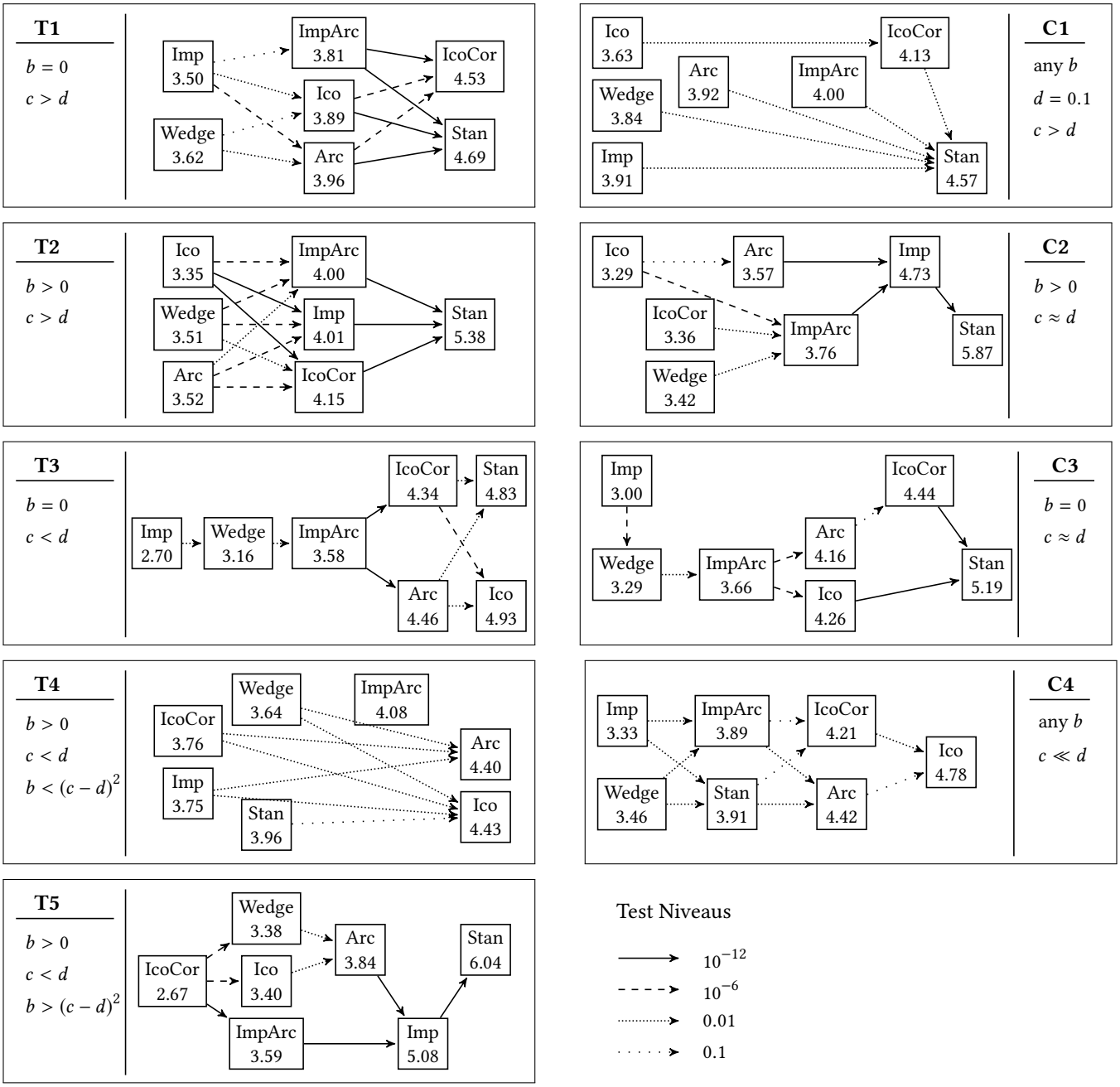

Figure 5: Directed graphs of algorithm orders in the different groups of test situations. In the left column, T1-T5, the groups from the theoretical analysis, in the right column, C1-C4, the groups from the cluster analysis are shown.

respectively. In situation T4, where the optimum is in the active region of x_2 and imputation is not profitable, not many differences between the kernels can be observed.

Third, groups of test situations are calculated automatically using a cluster analysis. Here, the optimal cluster method and amount of clusters is determined using the Dunn index [5]. As an additional constraint it was ensured that each cluster contains at least four test situations. The final partitioning was achieved using k-means clustering with four clusters. Fig. 6 shows the distribution of the experimental parameters within those clusters. Although the k-means algorithm had no access to the experimental parameters, it was able to identify connected clusters. Both the cluster C2 and C3 contain observations with $c \approx d$, i.e., the position of the optimum and the jump discontinuity are close to each other. While C2 mostly

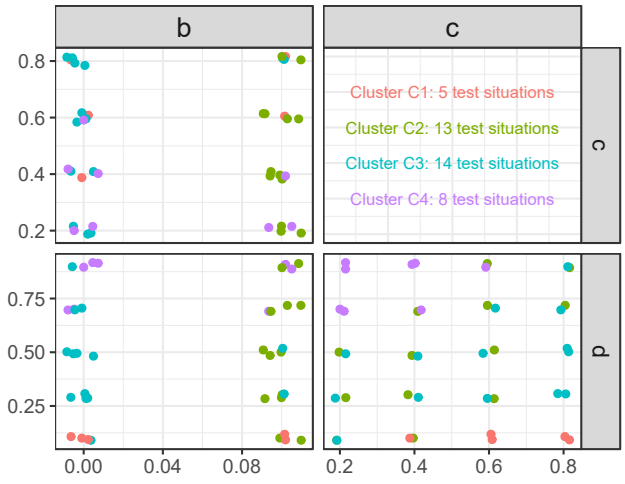

Figure 6: Distribution of experimental parameters b, c, and d in the four clusters.

consists of test situations with $b = 0$, all test situations in C3 ensure $b > 0$. In C1 and C4 both $b = 0$ and $b > 0$ are equally represented. In C1, $c \gg d$ and $d = 0.1$ hold, while $d \gg c$ holds in C4.

In the right column of Fig. 5, the results of the Nemenyi tests in these four clusters are visualized. In C1, not many differences between the kernels can be observed except that the Stan-kernel is clearly outperformed. Since in C1 the optimum is at $x_1 = 0.1$ in the inactive region of x_2, it does not seem to be important in which way the activity of x_2 is modeled in order to find the global optimum and hence most kernels perform equally well. However, it is still important that the activity is modeled in some way since the Stan-kernel is clearly outperformed. In C2, imputation should not be profitable since $b > 0$ and hence the imputation based Imp- and ImpArc-kernels perform rather badly. The remaining kernels perform equally well except for a slight advantage of the Ico-kernel over the Arc-kernel. In C3, where imputation should be possible, the Imp-kernel performs by far the best. Both Wedge and ImpArc also perform reasonably well and outperform the remaining kernels that can not execute an imputation. In C4, the Stan-kernel performs reasonably well and scores the fourth place. However, in this group of test situations with $d \gg c$, the test function is mostly just a two-dimensional sphere function. Therefore, modeling the hierarchical structure is not important. In all clusters, the Wedge-kernel is not significantly worse than the best kernel, except for cluster C3.

Comparing the theoretical groups T1-T5 and the computed groups C1-C4, some similarities and some conceptual differences can be observed (c.f. Fig. 7). Both of them use $b = 0$ versus $b > 0$ to discriminate whether imputation can be successful or not and both of them also investigate the difference $(c - d)$. Instead of just looking at the sign of $(c - d)$ as in the theoretical groups, the computed groups respect its magnitude. Moreover, contrary to Zaefferer and Horn [17], the relation between b and $(c - d)^2$ is not important.

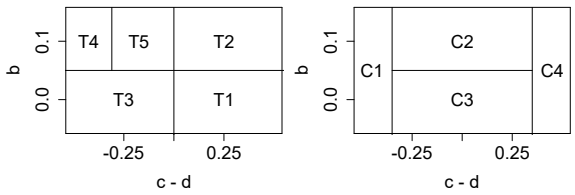

Figure 7: Conceptional visualization of theoretical (left) and computational (right) groups.

8 CONCLUSION

We proposed a new kernel for hierarchical search spaces, especially for the use within Kriging models in sequential model-based optimization approaches. We adopted a simple test function and benchmarked the optimization performance of the new kernel and existing kernels. In the statistical analysis of our benchmark we followed the guidelines given be Janez Demšar [4]. However, his guidelines gave space for several improvements. We proposed a new approach for the statistical analysis of benchmark results, including a cluster analysis of the results and a visualization using directed graphs. Our research questions can be answered as follows:

(1) Performance of the Wedge-kernel. The Wedge-kernel significantly outperforms other kernels averaged over all test situations. If only subgroups of test situations are considered, the Wedge-kernel may be outperformed by another kernel in each subgroup. In terms of the cluster analysis, the difference is only significant in one case: If imputation has the potential to be beneficial, the Imp-kernel is preferable. The Wedge-kernel combines the strength of the Arc- and the Imp-kernel, and avoids their weaknesses.

(2) Automated analysis meaningful. Our results show, that the analysis of subgroups of test situations is important. Although differences between algorithms may be significant considering all test situations, the order of these algorithms can differ in certain subgroups. This can be seen in subgroup C4, where the Stan-kernel outperforms three of the remaining kernels, although it is clearly outperformed if all test situations are considered together.

(3) Differences between automated and theoretical analysis. Naturally, the order of the algorithms does not differ that much between the two analysis methods. However, the found subgroups show some differences. Most outstanding is the fact, that b only has an influence on the algorithm order if the difference between c and d is small. Hence, modeling of the jump discontinuity at $x_1 = c$ is only important, if the optimum is close to it.

We aim at improving the experiments in our future work. Especially, while the test function is transparent and illustrative it is also fairly simple. Tests with more complex functions are desirable. Regrettably, no established test functions for hierarchical parameter spaces do exist. Good test functions should cover different function properties like multi-modality, non-separability, or ruggedness. One promising approach would be to follow the idea of the WFG test functions for multi-objective optimization [7]. The WFG test functions are produced in a constructive manner. Adapting them to hierarchical problems may allow producing cheap-to-compute test functions for hierarchical search spaces with adequate properties.

REFERENCES

[1] Bernd Bischl, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. 2017. mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions. *arXiv preprint arXiv:1703.03373* (2017).

[2] Malika Charrad, Nadia Ghazzali, Véronique Boiteau, and Azam Niknafs. 2014. NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set. *Journal of Statistical Software, Articles* 61, 6 (2014), 1–36.

[3] Gabor Csardi and Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal* Complex Systems (2006), 1695. http://igraph.org

[4] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.* 7 (2006), 1–30.

[5] J. C. Dunn. 1973. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics* 3, 3 (1973), 32–57.

[6] Alexander Forrester, Andras Sobester, and Andy Keane. 2008. *Engineering Design via Surrogate Modelling.* Wiley.

[7] S. Huband, P. Hingston, L. Barone, and L. While. 2006. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation* 10, 5 (2006), 477–506.

[8] Ying Hung, V. Roshan Joseph, and Shreyes N. Melkote. 2009. Design and Analysis of Computer Experiments With Branching and Nested Factors. *Technometrics* 51, 4 (nov 2009), 354–365. https://doi.org/10.1198/tech.2009.07097

[9] Frank Hutter and Michael A. Osborne. 2013. *A Kernel for Hierarchical Parameter Spaces.* Technical Report arXiv:1310.5738. arXiv.

[10] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13, 4 (1998).

[11] J. Mockus, V. Tiesis, and A. Zilinskas. 1978. *Towards Global Optimization 2.* North-Holland, Chapter The application of Bayesian methods for seeking the extremum.

[12] Katharine Mullen, David Ardia, David Gil, Donald Windover, and James Cline. 2011. DEoptim: An R Package for Global Optimization by Differential Evolution. *Journal of Statistical Software* 40, 6 (2011).

[13] Kevin Swersky, David Duvenaud, Jasper Snoek, Frank Hutter, and Michael Osborne. 2013. Raiders of the Lost Architecture: Kernels for Bayesian Optimization in Conditional Parameter Spaces. In *NIPS workshop on Bayesian Optimization in Theory and Practice.*

[14] Martin Zaefferer. 2017. Combinatorial Efficient Global Optimization in R - CEGO v2.2.0. https://cran.r-project.org/package=CEGO accessed: 2018-01-10.

[15] Martin Zaefferer. 2018. *Surrogate Models for Discrete Optimization Problems.* phdthesis. Technische Universität Dortmund. https://doi.org/10.17877/DE290R-19857

[16] Martin Zaefferer and Thomas Bartz-Beielstein. 2016. Efficient Global Optimization with Indefinite Kernels. In *Parallel Problem Solving from Nature–PPSN XIV.* Springer, 69–79.

[17] Martin Zaefferer and Daniel Horn. 2018. A First Analysis of Kernels for Kriging-Based Optimization in Hierarchical Search Spaces. In *Parallel Problem Solving from Nature – PPSN XV: 15th International Conference (Lecture Notes in Computer Science)*, Vol. 11102. Springer, 399–410.