19th International Conference on Modeling & Applied Simulation
17th International Multidisciplinary Modeling & Simulation Multiconference

ISSN 2724-0037 ISBN 978-88-85741-48-5 © 2020 The Authors.
DOI: 10.46354/i3m.2020.mas.008

Hyper-Parameter Handling for Gaussian Processes in Efficient Global Optimization

Bernhard Werth[1,*,†], Johannes Karder[1,2,3,*,†], Andreas Beham[2] and Stefan Wagner[2]

[1]Heuristic and Evolutionary Algorithms Laboratory
University of Applied Sciences Upper Austria, Softwarepark 13, 4232 Hagenberg, Austria
[2]Josef Ressel Center for Adaptive Optimization in Dynamic Environments
University of Applied Sciences Upper Austria, Softwarepark 13, 4232 Hagenberg, Austria
[3]Institute for Formal Models and Verification
Johannes Kepler University, Altenberger Straße 69, 4040 Linz, Austria

[*]Corresponding authors. Email addresses: bernhard.werth@fh-hagenberg.at, johannes.karder@fh-hagenberg.at
[†]Bernhard Werth and Johannes Karder share first author privileges.

Abstract

In simulation-based optimization, a common issue with many meta-heuristic algorithms is the limited computational budget. Performing a simulation is usually considerably more time-consuming than evaluating a closed mathematical function. Surrogate-assisted algorithms alleviate this problem by using representative models of the simulation which can be evaluated much faster. One of the most promising surrogate-assisted optimization approaches is Efficient Global Optimization, which uses Gaussian processes as surrogate-models. In this paper, the importance of carefully chosen hyper-parameters for Gaussian process kernels and a way of self-configuration is shown. Based on properties of the training set, e.g. distances between observed points, observed target values, etc., the hyper-parameters of the used kernels are initialized and bounded accordingly. With these initial values and bounds in mind, hyper-parameters are then optimized, which results in improved Gaussian process models that can be used for regression. The goal is to provide an automated way of hyper-parameter initialization, which can be used when building Kriging models in surrogate-assisted algorithms, e.g. Efficient Global Optimization (EGO). Obtained results show that applying the proposed hyper-parameter initialization and bounding can increase the performance of EGO in terms of either convergence speed or achieved objective function value.

Keywords: Gaussian Process; Hyper-Parameter; Self-Configuration; Efficient Global Optimization

1. Introduction

Modern day computer simulation is a powerful tool that can be used to gain understanding of real-world processes. It enables practitioners to obtain insights into the systems they are modelling, via statistical analysis and visualization. Simulation models are furthermore also used to make valuable predictions and analyze hypothetical ("what-if") scenarios. Being able to evaluate possible actions or decisions without actually interfering with real-world systems leads directly to the idea of simulation-based optimization (Affenzeller et al., 2015). Current simultion models are often so complex and runtime-intensive that researchers and practition-

ers resort to heuristic and meta-heuristic optimization methods, often in combination with surrogate-models. Such surrogate-models help the optimization algorithm by providing fast evaluations, guiding the search into promising regions and can suppress possible uncertainties caused by stochastic elements in the simulation.

In surrogate-assisted optimization, the well-known approach of *Efficient Global Optimization* (EGO) (Jones et al., 1998), i.e. adaptive sampling through *Expected Improvement* (EI) calculation via *Gaussian process* (aka *Kriging*) models, is still, to this day, considered one of the most promising ways to optimize expensive black box functions. Thus, EGO also finds many applications in the field of simulation-based optimization, where an expensive simulation model is used to evaluate solution candidates and therefore only a limited number of simulations can be computed in a reasonable amount of time. Within EGO, Gaussian processes (Rasmussen and Williams, 2005) are used to create a surrogate-model which imitates the actual expensive objective function. Gaussian processes cannot only be used for regression, i.e. for predicting target values for unobserved input values. They also provide an uncertainty value for each prediction made. These predictions and uncertainties can be used to calculate the EI (Jones et al., 1998). The kernels used by Gaussian processes have hyper-parameters, which have to be optimized to achieve models with good prediction accuracy and uncertainty estimation. The better these hyper-parameters are tuned, the better the Gaussian process model will be and therefore the more informative the EI will become. Hyper-parameters can be optimized in various ways, usually using a gradient descent algorithm with multiple restarts. By setting the hyper-parameter bounds, we aim to constrain the gradient descent to areas of the hyper-parameter space where the resulting model is beneficial for the search.

In this paper, we show how the bounds of hyper-parameter values can be calculated based on already observed data, i.e. inputs and outputs. Software frameworks and libraries, such as e.g. *scikit-learn* (https://scikit-learn.org/) for Python, offer implementations of various kernels that can be used for Kriging. However, the configuration of such kernel parameters is a non-trivial task, at least for people that are not as firm in the subject as others. In comparison, configuring a Differential Evolution that yields good results for a particular kind of objective function is in our opinion much easier to accomplish. Therefore, our goal is to provide a sound, practical guide for setting defaults for Kriging hyper-parameters, as well as respective upper and lower limits, which can be used in various EGO or Kriging implementations.

The rest of this paper is structured as follows: In Section 2, literature relevant to the present topic is shown, while more specific information on EGO is given in Section 3 The proposed hyper-parameter handling is explained in Section 4, and Section 5 outlines the experimental setup and the achieved results. Finally, conclusions are drawn, and possible future work is proposed in Section 6.

2. Related Literature

In this section, we provide references to multiple papers on Gaussian processes, machine learning kernels, hyper-parameter optimization, infill criteria, as well as theoretical foundations and practical experiments for EGO. People more interested in EGO itself can have a look at the surveys (Bartz-Beielstein, 2016) and (Regis, 2019), or (Yang et al., 2019).

EGO nowadays, compared to the standard EGO proposed by Jones et al. (1998), can be seen as more of a framework than one single algorithm. During the last decades, the algorithm has been modified and extended in various ways.

It is important for EGO to be able to balance the search between exploration and exploitation. This is usually achieved by using EI as an *Infill Criterion* (IC), which results in promising solutions that are either expected to be of high quality (exploitation) or located in sparsely observed areas of the search space (exploration). Picheny et al. (2013) published a benchmark in which various infill criteria for noisy optimization were evaluated. Zaefferer and Bartz-Beielstein (2016) showed how to use indefinite kernels within EGO to be able to optimize combinatorial problems. Most infill criteria are inherently tied to Gaussian process models and/or require models to yield a value of uncertainty for their respective predictions. However, there exist estimation heuristics that provide uncertainty measures for any type of model and can be used instead. One example for such a heuristic is the knn-uncertainty measure proposed by van Stein et al. (2018).

Furthermore, Palar and Shimoyama (2019) extend EGO with model selection for kernel functions in order to create good Kriging kernels. Various surrogate-assisted optimization algorithms, including EGO, have also been parallelized, as e.g. shown by Haftka et al. (2016).

Selecting the right kernels and hyper-parameters, is a non-trivial task and many different approaches. A critical drawback of Gaussian Processes is their relatively high computational training time. This problem compounds, if the heuristic search for optimal hyper-parameters requires many models to be trained.

One paper which gives insights into a particular way of hyper-parameter tuning has been published only recently by Berkenkamp et al. (2019). They apply online hyper-parameter adaption and show how the length scales of Kriging kernels can be adjusted to guarantee eventual convergence to the optimum. Xiao et al. (2014) propose two methods to optimize Kriging ker-

nel parameters based on farthest nearest neighbor distances within observed samples and what they call "tightness" of decision variables. They also show how hyper-parameter optimization can use objective function values other than maximum likelihood. An empirical analysis (Tran et al., 2020) states that there is no a priori guarantee that hyper-parameter optimization improves performance of prediction models. They consider classification algorithms and propose a method that determines whether hyper-parameters should be optimized at all or kept at their default values. Even more generalizable, the No-Free-Lunch theorem for supervised machine learning (Wolpert, 2002) tells us that on all possible datasets, a model created with and a model created without hyper-parameter optimization will succeed and fail equally often. However, for many practical applications hyper-parameter tuning appears to be the correct choice. Bull (2011) shows that by taking assumptions on the smoothness of an underlying fitness landscape, theoretical upper bounds for the convergence behavior of an ideal EGO algorithm can be calculated. An "ideal" EGO does not suffer from numerical instabilities and the optimization of the infill criterion and the kernel hyper-parameters never yields suboptimal results. In Bull's derivation, the upper bound of e.g. the length scale of Kriging kernels is found to be of great importance in order for an ideal EGO to be able to converge.

3. Efficient Global Optimization

Since its introduction by Jones et al. (1998), the combination of adaptive sampling using infill criteria and building meta-models via Kriging, aka. EGO, has established itself as one of the most prominent algorithmic approaches in surrogate-assisted optimization. EGO is described in pseudocode as follows:

```
points ← SamplePoints()

while termination criterion not met do
    model ← BuildKrigingModel(points)
    point ← OptimizeEI(model, points)
    objVal ← EvaluateExpensively(point)
    points.Add(point, objVal)
end
```

Algorithm 1: Pseudocode describing EGO.

The next point to evaluate expensively is determined by traversing the search space looking for the point with the best infill criterion. Various infill criteria have been proposed in the literature. The most prominent infill criterion, which was also proposed together with the standard EGO, is EI. EI indicates whether a solution is better than the best-found-so-far by considering the predicted objective function value and the uncertainty of the prediction. Solutions with good predicted objective function value are exploited and solutions whose predicted objective function value is affected by high uncertainty are explored too. EI is defined as

$$\mathrm{E}[I(\mathbf{x})] = \underbrace{(f_{\min} - \hat{y})\Phi\left(\frac{f_{\min} - \hat{y}}{s}\right)}_{\text{exploitation}} + \underbrace{s\phi\left(\frac{f_{\min} - \hat{y}}{s}\right)}_{\text{exploration}} \quad (1)$$

where $f_{\min}$ is the current best objective function value observed so far, $\hat{y}$ is the model's predicted objective function value for $\mathbf{x}$, s is the uncertainty of the prediction, Φ is the standard normal density and ϕ the standard normal distribution function.

4. Improved Hyper-Parameter Handling

Stable Gaussian process models are fundamental for the functionality of EGO. Usually kernel hyper-parameters are tuned to improve the model using the Maximum Likelihood Method. To reduce computational complexity, some implementations bound these hyper-parameters. Bounding hyper-parameters is not only useful to reduce the time required for model building, but also to give the Gaussian process hints where optimal parameters should be located. We therefore introduce an algorithm that selects appropriate bounds for used kernel parameters, depending on the data that has already been observed. The goal is to find hyper-parameters that lead to stable Gaussian processes and improve the overall models with regards to prediction accuracy and uncertainty, both of which are required for good estimation/calculation of EI.

The python machine learning library *scikit-learn* (v0.22.1) offers an implementation of Gaussian processes, as well as multiple kernels that can be used in combination. Going forward we will use the following combined kernels for Kriging:

$$k(x, x_i) = C * RBF(x, x_i) + W \quad (2)$$

A Constant kernel C is used to scale an isometric Radial Basis Function kernel RBF (Gaussian kernel) and an added White kernel W allows for the representation of noise. The library offers implementations for all of these, and provides the following default parameters:

Length parameter of RBF kernel:

- default value: $\mathcal{L}_{init} = 1.0$
- bounds: $[\mathcal{L}_{lb}, \mathcal{L}_{ub}] = [10^{-5}, 10^{5}]$

Value of the Constant kernel (scale):

- default value: $\mathcal{S}_{init} = 1.0$
- bounds: $[\mathcal{S}_{lb}, \mathcal{S}_{ub}] = [10^{-5}, 10^{5}]$

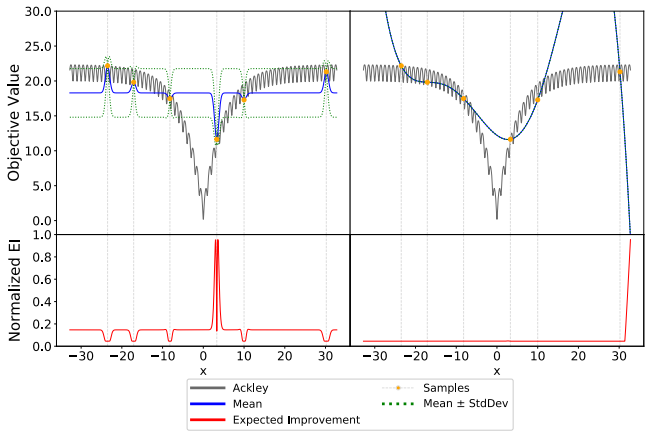

Figure 1. Two Gaussian process models with too small (left) and large (right) length.

Noise level of the White kernel:

- default value: $\mathcal{N}_{init} = 1.0$
- bounds: $[\mathcal{N}_{lb}, \mathcal{N}_{ub}] = [10^{-5}, 10^{5}]$

4.1. Length

The length (usually called length scale in the literature, but here referred to as length in order to not cause any confusion with the scale hyper-parameter introduced below) determines the area of influence observed samples have on the prediction of unobserved points. A small length means that this "influence basin" is reduced and if the length is too small, there is not much influence at all. Figure 1 shows two models with improper length scales. On the left, the model's length is too small. The influence of observed samples drops fast, meaning that the model makes the same predictions for almost all unobserved points. Thus, the highest EI values are very close to the best current observation and the EGO gets effectively trapped in a local optimum.

On the right, the length scale is too high. This results in the model having almost no uncertainty for any prediction, rendering EI ineffective. The only steering of EI can be seen on the far right side of the search space, where the prediction falls rapidly, resulting in higher EI. However, this is an undesired effect, since the extreme mean values are an artefact of the Gaussian Processes smoothing behaviour, rather than an actual approximation of the data in this region.

We propose to lower bound the length parameter of any isometric distance-dependent kernel in the context of EGO as proportional to the distance of the farthest nearest neighbour pair of sample points (cf. Eq. (3)). Depending on the "width" of the kernel function different lower bounds will be appropriate. For the RBF kernel a factor of $\frac{1}{4}$ prevents the EI measure from developing areas of equal infill values along the line between two nearest neighbour sample points (see the left side of Figure 1), ensuring a meaningful search gradient.

$$\mathcal{L}_{lb} = \frac{1}{4} \max_{x^i \in X} \min_{x^j \in X} \sqrt{\sum_{d=1}^{n} (x_d^i - x_d^j)^2} \tag{3}$$

where X are all observed inputs and x^i and x^j are sample points of dimension n. This results in a lower bound for the length so that it is at least as big as the longest distance between nearest neighbours, meaning that along the line connecting two nearest neighbours the gaussian process will never fully revert it's mean state, this ensures a meaningful "direction of search" for the EI optimizer.

The upper bound is calculated by determining the value ranges for every input dimension. From this range vector, the Frobenius norm is calculated, which is the length of the space that all observed points span. By halving this length we ensure that the drop in "point wise influence" is at least somewhat noticeable (cf. Eq. (4)). Note that, similarly to $\mathcal{L}_{lb}$ the multiplicative factor depends on the width and shape of the kernel function.

$$\mathcal{L}_{ub} = \frac{1}{2} \sqrt{\sum_{d=1}^{n} (\max_{x^i \in X} x_d^i - \min_{x^j \in X} x_d^j)^2} \tag{4}$$

The maximum length is therefore bound proportional to the length of the observed space, meaning that the two points with the greatest distance can still heavily influence each other if needed. Finally, in the following experiments, $\mathcal{L}_{init}$ is the center, i.e. mean, of the computed bounds (cf. Eq. (5)).

$$\mathcal{L}_{init} = \frac{\mathcal{L}_{lb} + \mathcal{L}_{ub}}{2} \tag{5}$$

4.2. Scale

The scale closely relates to the length. A low scale results in less uncertainty, which in turn causes EI to favour areas of the search space that are located around the best solution found so far. There is no balance between exploration and exploitation as the latter is heavily favoured, making the algorithm very susceptible to getting stuck in a local optimum. However, scale values that are too big lead to overestimated uncertainties. The EI of unobserved points will mainly depend on the distance to all observed points. This gives unobserved areas greater EI values, which in turn favours exploration, as the predicted value has less impact because of higher uncertainties. This is in a sense a little bit less critical, since the search is not per se "stuck", but it can cause the algorithm to waste evaluations in unpromising regions of the search space. Compared to the length in this configuration, the scale only influences the uncertainties of the model and not

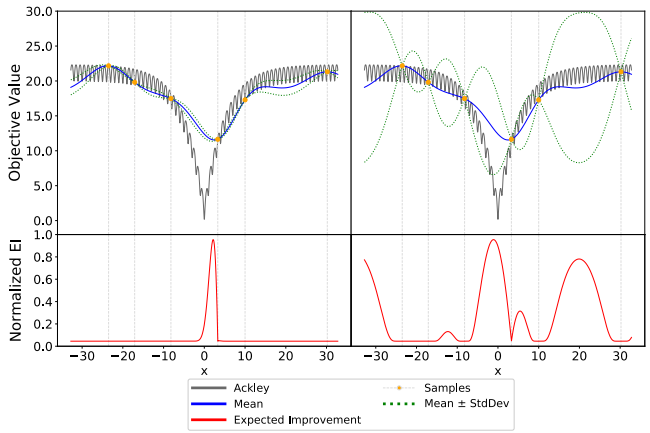

Figure 2. Two Gaussian process models with too small (left) and large (right) scale.

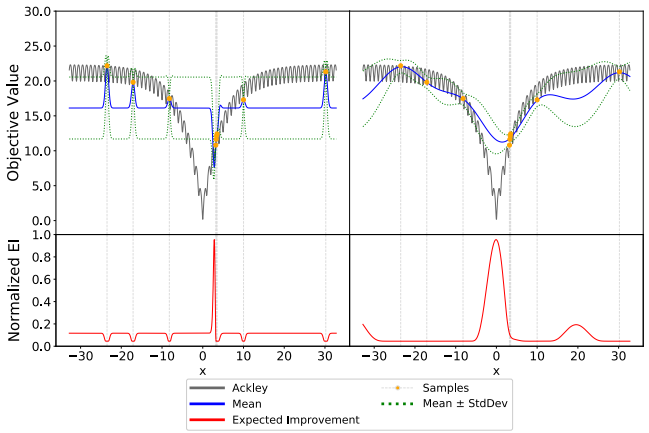

Figure 3. Two Gaussian process models, one trained without (left) and one trained with (right) a noisy kernel.

the predictions. Figure 2 shows two models, left and right, with too small and large scales, respectively.

We propose to set both scale bounds proportional to the range of all observed objective function values $\Delta(y)$ (with y being the set of all sampled objective values). To obtain a lower bound, this range is then divided by twice the number of samples (cf. Eq. (6)). This allows for continuously smaller uncertainties, as the number of sample points in the search space increases.

$$\mathcal{S}_{lb} = \frac{\Delta(y)}{2 * |X|} \tag{6}$$

The scale's upper bound depends on considerably more factors and is defined as follows (cf. Eq. (8)).

$$D = \left\{ \sqrt{\sum_{d=1}^{n} (x_d^i - x_d^j)^2} \; | x^i \in X, x^j \in X, x^i \neq x^j \right\} \tag{7}$$

$$\mathcal{S}_{ub} = \left(1 + \frac{\text{std}(y)}{\Delta(y)}\right) * \frac{\max(D)}{\text{mean}(D)} * \Delta(y) * (1 + n) \tag{8}$$

Preliminary experiments showed that higher dimensional problem instances require larger scale values, hence $\mathcal{S}_{ub}$ is proportional to n. Furthermore, larger scale values are preferred, if the sample points are distributed very unevenly in the search space and/or the objective space. By scaling $\mathcal{S}_{ub}$ with ratio of the maximum and the average distance between sample points, very uneven sample distributions can be accommodated. In a similar vein, we slightly increase $\mathcal{S}_{ub}$ if the standard deviation of the sampled objective values are large compared to their range. Finally, the initial value of the scale parameter is half of the range of target values (cf. Eq. (9))

$$\mathcal{S}_{init} = \frac{\Delta(y)}{2} \tag{9}$$

4.3. Noise

There are situations in which noise kernels can help even when optimizing non-stochastic objective functions. One example is shown in Figure 3. In the shown situation, EGO has sampled a few points rather close to each other in an area where objective values change rapidly. On the left side, no noise kernel was used ($W = 0$). This causes the hyper-parameter optimizer to prefer very small length values and the model is inadequate for EGO as the best EI values lie between the already clustered points. Continuing the algorithm would only compound the problem.

However, if a noisy kernel is used (right side), points do not have to be fit perfectly by the model and it is able to provide an adequate estimation for the overall curve points.

Most EGO-variants use the Maximum Likelihood Criterion to find optimal hyper-parameters within given bounds. It is noteworthy, that the log-likelihood of the left model (-19.043) is decidedly larger than for the right model (-20.305) and would therefore be preferred by most optimizers, although the right model is more appropriate for a continued search.

Introducing noise to the usually noiseless EGO formulation allows the algorithm to fight off model degeneration and even improves numeric stability, but has the distinct drawback of no longer forcing the algorithm to avoid sampling the same data point twice, which could potentially waste evaluations (the search is no longer guaranteed to be eventually global). Furthermore, larger noise values can cause the model to over-generalize and "flatten out". What happens to the model predictions and EI is shown in Figure 4. Low noise (left) causes some uncertainty around already known points, however, the observed target values still influence the model's predictions. When looking at a model with a noise level set too high (right), we can observe that this renders all observations within the

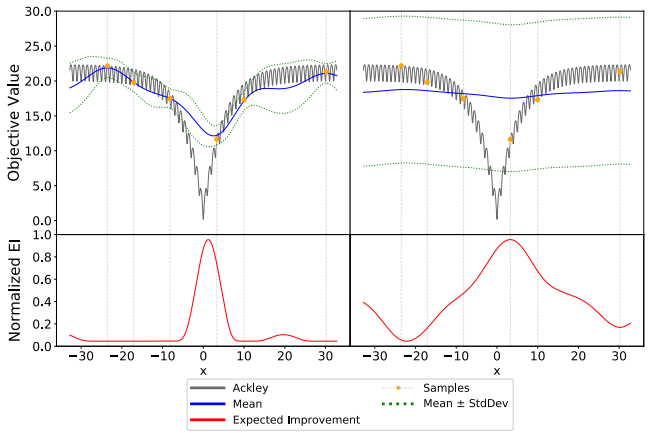

Figure 4. Two Gaussian process models with low (left) and high (right) noise.

noise range more or less equal, and results in poor prediction models.

$\mathcal{N}_{init}$, as well as the upper bound is the standard deviation of all observed target variables, the lower bound is the standard deviation scaled by δ (cf. Eq. (10) and (11)).

$$\mathcal{N}_{ub} = \mathcal{N}_{init} = \text{std}(y) \tag{10}$$

$$\mathcal{N}_{lb} = \text{std}(y) * \delta \tag{11}$$

Scaling with δ, which in our case was chosen to be $\frac{1}{m}$ where m was the total number of iterations an EGO run was allowed to perform, should reduce the lower bound to give the Gaussian process some room and, if necessary, make it possible to use almost noiseless kernel functions. Lastly, it needs to be mentioned that we opted to normalize the objective values of all sample points before training the Gaussian process model (i.e., our models assume a constant non-zero mean). This allows the model to trend towards mean(y) rather than a fixed 0 when extrapolating far from any sample points and makes the algorithm invariant to translations in the objective space. In subsequent iterations of EGO, all bounds are continuously updated and the initial values for the kernel parameters are set to the ones obtained in the previous iteration, while respecting the updated bounds. After the hyper-parameter bounds have been fixed and the initial values have been set, a gradient descent optimizes the parameters within their respective bounds. The optimization budget of the hyper-parameter optimizer is limited to 100 function evaluations. The `minimize` function in SciPy (https://www.scipy.org/) takes a parameter for the maximum iterations that should be run. However, the gradient descent, in our case a L-BFGS-B (Byrd et al., 1995), stops as soon as a local optimum has been reached. If such a local optimum is reached after 40 iterations, there are still 60 iterations left that can be executed. In this case, the gradient descent is restarted with a maximum of 60 iterations. Therefore, multiple restarts can occur until the evaluation budget has been used.

Table 1. The test functions used for the benchmarks.

Ackley	$f(\mathbf{x}) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi\mathbf{x}_i)}$
	Domain: $-32.768 \leq \mathbf{x}_i \leq 32.768$
	Best obj. val.: 0 at $\mathbf{x} = [0, 0, \ldots, 0]$
	Worst obj. val.: $20 + e$
Griewank	$f(\mathbf{x}) = 1 + \sum_{i=1}^{n}\frac{\mathbf{x}_i^2}{4000} - \prod_{i=1}^{n}\cos(\frac{\mathbf{x}_i}{\sqrt{i}})$
	Domain: $-600 \leq \mathbf{x}_i \leq 600$
	Best obj. val.: 0 at $\mathbf{x} = [0, 0, \ldots, 0]$
	Worst obj. val.: $\frac{n*600*600}{4000+1+1}$
Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}\left[\mathbf{x}_i^2 - 10\cos(2\pi\mathbf{x}_i)\right]$
	Domain: $-5.12 \leq \mathbf{x}_i \leq 5.12$
	Best obj. val.: 0 at $\mathbf{x} = [0, 0, \ldots, 0]$
	Worst obj. val.: $n * (10 + 30.3533)$
Sphere	$f(\mathbf{x}) = \sum_{i=1}^{n}\mathbf{x}_i^2$
	Domain: $-5.12 \leq \mathbf{x}_i \leq 5.12$
	Best obj. val.: 0 at $\mathbf{x} = [0, 0, \ldots, 0]$
	Worst obj. val.: $n * 5.12 * 5.12$

5. Experiments and Results

Eight different variations of EGO are compared. We test our proposed method of hyper-parameter initialization and bounding (*bounded*) against the scikit-learn default settings (*unbounded*) with different hyper-parameter optimization budgets, i.e. how many hyper-parameter evaluations the multi-start gradient descent can perform. Each EGO run starts with 10 initial samples obtained via latin-hypercube sampling. 10 repetitions are performed for each variation. To increase comparability between variations, the initial sample set for bounded EGO with budget 100 in repetition r is also used for repetition r of all other variations. The four single-objective functions in Table 1 will serve as a benchmark.

In further experiments, all four benchmark functions were additionally distorted with additive noise. The noisy variants of all test functions $f_a(\mathbf{x})$ change the original objective function value returned by $f(\mathbf{x})$ as defined in Eq. (12). (Δf denotes range of possible objective function values depending on the function and the dimensionality).

$$f_a(\mathbf{x}) = f(\mathbf{x}) + \alpha \qquad \alpha \sim \mathcal{N}(0, 0.01 * \Delta f) \tag{12}$$

Figure 5 depicts objective value histories for all EGO variants, test functions and dimensions, averaged over 10 repetitions, for all noiseless experiments. The same data is plotted in Figure 6 for all experiments where objective function values were modified with additive

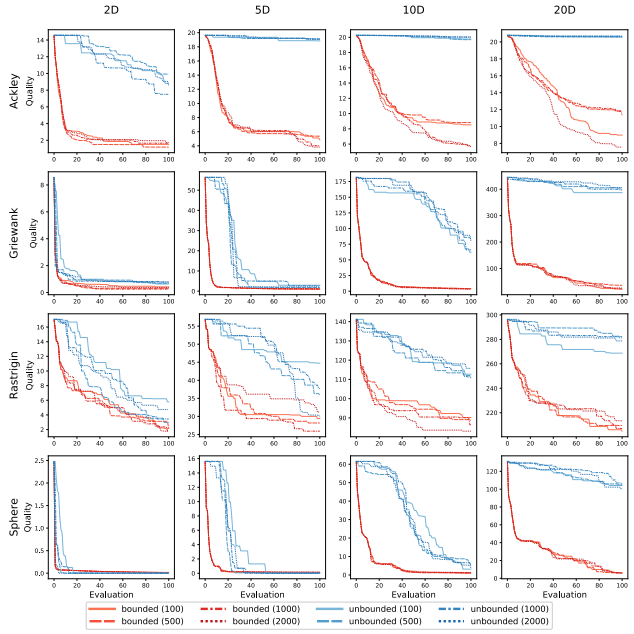

Figure 5. Mean best objective function value histories of all noiseless experiments.

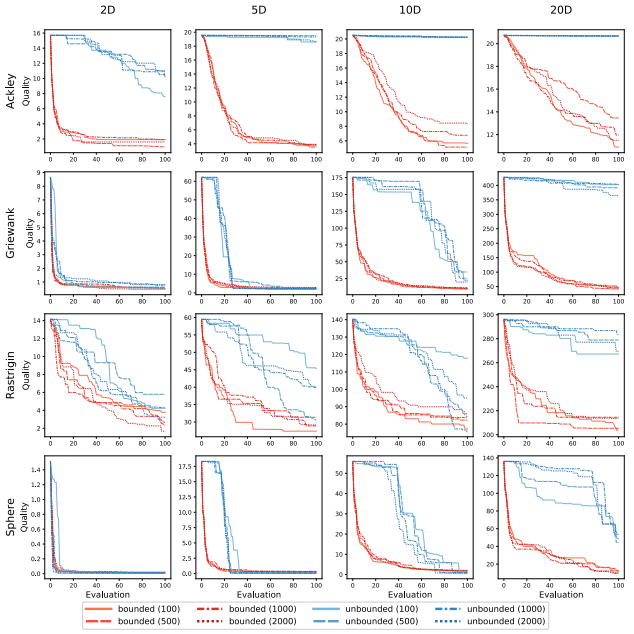

Figure 6. Mean best objective function value histories of all noisy experiments with additive noise.

noise. In the noisy case, all solutions expensively evaluated by EGO have been re-evaluated with the noiseless objective function to obtain "true" objective values.

As depicted in Figures 5 and 6, the algorithms using the proposed hyper-parameter handling usually perform better in terms of best achieved objective value and convergence speed than their unbounded counterparts. The unimodal Sphere function is comparatively easy to solve in lower dimensions, here the unbounded EGO variants slightly outperform the bounded runs. This happens because bounded EGO explores the search space more, while the bounded variants sample closely to their current best solution.

For all variants, the budget in terms of hyper-parameter evaluations seems to have no or only very little effect on algorithm performance. One should note that computational costs of hyper-parameter optimizations scales with $O(n^3)$, where n is the number of training samples. Depending on the simulation that is being optimized, hyper-parameter optimization therefore can make up a larger portion of the overall optimization runtime. It is therefore beneficial to use a method for optimizing the hyper-parameters that requires as few hyper-parameter evaluations as possible.

6. Conclusion and Outlook

From a practical point of view, algorithms based on Gaussian processes such as EGO can easily be misparameterized and present several pitfalls. Effective use of EGO requires a certain amount of understanding of its inner workings. Automated hyper-parameter bounding can help avoid these pitfalls, making EGO easier to apply without running into degenerated algorithm states.

In this paper, an automated way of initializing and bounding hyper-parameter values for Efficient Global Optimization was proposed. The tested EGO implementations have been configured to use an RBF kernel for the length scale, together with a Constant kernel for scaling and a White kernel to account for possibly noisy optimization problems. Depending on the distances between observed samples and respective objective function values, such initial parameter values and bounds can quickly be computed using basic arithmetic operations. An EGO variant using the proposed hyper-parameter handling, as opposed to a default EGO without effective hyper-parameter initialization or bounding, showed quite promising results. Classic EI and an additional infill criterion were explored, and the proposed method appeared beneficial for both criteria, although in a different order of magnitude. We observe a performance increase in both, convergence rate and achieved objective function value, for several problem instances.

One possible way to further improve the hyper-parameter estimation within EGO is to implement relaxation/tightening for hyper-parameter bounds, utilizing the information from previous iterations. So far, the proposals for bounds and initial values shown in this paper were created during semi-empirical testing and thinking about how various fitness landscapes ef-

fect the requirements of different lengths, scales and noise levels of Kriging kernels. Of course, this means that future research must be done in order to provide a solid, mathematical understanding of the effectiveness of our method. As already shown in Section 2, there are several other approaches to hyper-parameter optimization for Gaussian processes, e.g. marginalization. In the future, comparisons should be made between these approaches and the one presented here. We are confident that our hyper-parameter bounding and initialization provides significant benefits while being easy to implement and requiring little overhead in terms of runtime.

7. Acknowledgements

(1) Work described in this paper was done within the Produktion der Zukunft Project Integrated Methods for Robust Production Planning and Control (SIMGENOPT2, #858642), funded by the Austrian Research Promotion Agency (FFG).
(2) The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

References

Affenzeller, M., Beham, A., Vonolfen, S., Pitzer, E., Winkler, S. M., Hutterer, S., Kommenda, M., Kofler, M., Kronberger, G., and Wagner, S. (2015). *Simulation-Based Optimization with HeuristicLab: Practical Guidelines and Real-World Applications*, pages 3–38. Springer International Publishing, Cham.

Bartz-Beielstein, T. (2016). A survey of model-based methods for global optimization. *Bioinspired Optimization Methods and Their Applications*, pages 1–18.

Berkenkamp, F., Schoellig, A. P., and Krause, A. (2019). No-regret bayesian optimization with unknown hyperparameters. *Journal of Machine Learning Research*, 20(50):1–24.

Bull, A. D. (2011). Convergence rates of efficient global optimization algorithms.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*,

Palar, P. S. and Shimoyama, K. (2019). Efficient global optimization with ensemble and selection of kernel functions for engineering design. *Structural and Multidisciplinary Optimization*, 59(1):93–116.

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

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press.

Regis, R. G. (2019). A survey of surrogate approaches for expensive constrained black-box optimization. In *World Congress on Global Optimization*, pages 37–47. Springer.

Tran, N., Schneider, J.-G., Weber, I., and Qin, A. (2020). Hyper-parameter optimization in classification: To-do or not-to-do. *Pattern Recognition*, 103:107245.

van Stein, B., Wang, H., Kowalczyk, W., and Bäck, T. (2018). A novel uncertainty quantification method for efficient global optimization. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 480–491. Springer.

Wolpert, D. H. (2002). *The Supervised Learning No-Free-Lunch Theorems*, pages 25–42. Springer London, London.

Xiao, Y., Wang, H., Zhang, L., and Xu, W. (2014). Two methods of selecting gaussian kernel parameters for one-class svm and their application to fault detection. *Knowledge-Based Systems*, 59:75–84.

Yang, K., van der Blom, K., Bäck, T., and Emmerich, M. (2019). Towards single-and multiobjective bayesian global optimization for mixed integer problems. In *AIP Conference Proceedings*, volume 2070, page 020044. AIP Publishing LLC.

Zaefferer, M. and Bartz-Beielstein, T. (2016). Efficient global optimization with indefinite kernels. In Handl, J., Hart, E., Lewis, P. R., López-Ibáñez, M., Ochoa, G., and Paechter, B., editors, *Parallel Problem Solving from Nature – PPSN XIV*, pages 69–79, Cham. Springer International Publishing.