Model Guided Sampling Optimization for Low-dimensional Problems

Lukáš Bajer[1,2] and Martin Holeňa[1]
[1]*Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, Prague, Czech Republic*
[2]*Faculty of Mathematics and Physics, Charles University in Prague, Malostranské nám. 25, Prague, Czech Republic*

Keywords: black-box Optimization, Gaussian Process, Surrogate Modelling, EGO.

Abstract: Optimization of very expensive black-box functions requires utilization of maximum information gathered by the process of optimization. Model Guided Sampling Optimization (MGSO) forms a more robust alternative to Jones' Gaussian-process-based EGO algorithm. Instead of EGO's maximizing expected improvement, the MGSO uses sampling the probability of improvement which is shown to be helpful against trapping in local minima. Further, the MGSO can reach close-to-optimum solutions faster than standard optimization algorithms on low dimensional or smooth problems.

1 INTRODUCTION

Optimization of expensive empirical functions forms an important topic in many engineering or natural-sciences areas. For such functions, it is often impossible to obtain any derivatives or information about smoothness; moreover, there is no mathematical expression nor an algorithm to evaluate. Instead, some simulation or experiment has to be performed, and the value obtained through a simulation or experiment is the value of the objective function being considered. Such functions are also called black-box functions. They are usually very expensive to evaluate; one evaluation may cost a lot of time and money to process.

Because of the absence of derivatives, standard continuous first- or second-order derivative optimization methods cannot be used. In addition, the functions of this kind are usually characterized by a high number of local optima where simple algorithms can be trapped easily. Therefore, different derivative-free optimization methods (often called meta-heuristics) have been proposed. Even though these methods are rather slow and sometimes also computationally intensive, the cost of the empirical function evaluations is always much higher and the cost of these evaluations dominates the computational cost of the optimization algorithm. For this reason, it is crucial to decrease the number of function evaluations as much as possible.

Evolutionary algorithms constitute a broad family of meta-heuristics that are frequently used for black-box optimization. Furthermore, some additional algorithms and techniques have been designed to minimize the number of objective function evaluations. All of the three following approaches use a model (of a different type in each case), which is built and updated within the optimization process.

Estimation of distribution algorithms (EDAs) (Larrañaga and Lozano, 2002) represent one such approach: EDAs iteratively estimate the distribution of selected solutions (usually solutions with fitness above some threshold) and sample this distribution forming a new set of solutions for the next iteration.

Surrogate modelling is the technique of learning and usage of a regression model of the objective function (Jin, 2005). The model (called surrogate model in this context) is then used to evaluate some of the candidate solutions instead of the original costly objective function.

Our method, *Model Guided Sampling Optimization* (MGSO), takes inspiration from both these approaches. It uses a regression model of the objective function, which also provides an error estimate. However, instead of replacing the objective function with this model, it combines its prediction and the error estimate to get a probability of reaching a better solution in a given point. Similarly to EDAs, the MGSO then samples this pseudo-distribution[1], in order to obtain

[1]a function proportional to a probability distribution, it's value is given by the *probability of improvement*

Bajer L. and Holeňa M..
Model Guided Sampling Optimization for Low-dimensional Problems.
DOI: 10.5220/0005222404510456
In *Proceedings of the International Conference on Agents and Artificial Intelligence* (ICAART-2015), pages 451-456
ISBN: 978-989-758-074-1

the next set of solution candidates.

The MGSO is also similar to Jones' Efficient Global Optimization (EGO) (Jones et al., 1998). Like EGO, the MGSO uses a Gaussian process (GP, see (Rasmussen and Williams, 2006) for reference), which provides a guide where to sample new candidate solutions in order to explore new areas and exploit promising parts of the objective-function landscape. Where *EGO selects* a single or very few solutions maximizing a chosen criterion – Expected Improvement (EI) or Probability of Improvement (PoI) – the *MGSO samples* the latter criterion. At the same time, the GP serves for the MGSO as a surrogate model of the objective function for a small proportion of the solutions.

This paper further develops the MGSO algorithm introduced in (Bajer et al., 2013). It brings several improvements (new optimization procedures and more general covariance function) and performance comparison to EGO. The following section introduces methods used in the MGSO, Section 3 describes the MGSO algorithm, and Section 4 comprises some experimental results on several functions from the BBOB testing set (Hansen et al., 2009).

2 GAUSSIAN PROCESSES

Gaussian process is a probabilistic model based on Gaussian distributions. This type of model was chosen because it predicts the function value in a new point in the form of univariate Gaussian distribution: the mean and the standard deviation of the function value are provided. Through the predicted mean, the GP can serve as a surrogate model, and the standard deviation is an estimate of the prediction uncertainty in a new point.

The GP is specified by mean and covariance functions and a relatively small number of covariance function's hyper-parameters. The hyper-parameters are usually fitted by the maximum-likelihood method.

Let $\mathbf{X}_N = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^D\}_{i=1}^N$ be a set of N training D-dimensional data points with known dependent-variable values $\mathbf{y}_N = \{y_i\}_{i=1}^N$ and $f(\mathbf{x})$ be an unknown function being modelled for which $f(\mathbf{x}_i) = y_i$ for all $i \in \{1,\ldots,N\}$. The GP model imposes a probabilistic model on the data: the vector of known function values $\mathbf{y}_N$ is considered to be a sample of a N-dimensional multivariate Gaussian distribution with the value of the probability density $p(\mathbf{y}_N \mid \mathbf{X}_N)$. If we take into consideration a new data point $(\mathbf{x}_{N+1}, y_{N+1})$, the value of the probability density in the new point is

$$p(\mathbf{y}_{N+1} \mid \mathbf{X}_{N+1}) = \frac{\exp(-\frac{1}{2}\mathbf{y}_{N+1}^\top \mathbf{C}_{N+1}^{-1}\mathbf{y}_{N+1})}{\sqrt{(2\pi)^{N+1}\det(\mathbf{C}_{N+1})}} \quad (1)$$

where $\mathbf{C}_{N+1}$ is the covariance matrix of the Gaussian distribution (for which mean is usually set to constant zero) and $\mathbf{y}_{N+1} = (y_1,\ldots,y_N,y_{N+1})$ (see (Buche et al., 2005) for details). This covariance can be written as

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^\top & \kappa \end{pmatrix} \quad (2)$$

where $\mathbf{C}_N$ is the covariance of the Gaussian distribution given the N training data points, $\mathbf{k}$ is the vector of covariances between the new point and training data, and κ is the variance of the new point itself.

Predictions. Predictions in Gaussian processes are made using Bayesian inference. Since the inverse $\mathbf{C}_{N+1}^{-1}$ of the extended covariance matrix can be expressed using the inverse of the training covariance $\mathbf{C}_N^{-1}$, and $\mathbf{y}_N$ is known, the density of the distribution in a new point simplifies to a univariate Gaussian with the density

$$p(y_{N+1} \mid \mathbf{X}_{N+1}, \mathbf{y}_N) \propto \exp\left(-\frac{1}{2}\frac{(y_{N+1}-\hat{y}_{N+1})^2}{s_{y_{N+1}}^2}\right) \quad (3)$$

with the mean and variance given by

$$\hat{y}_{N+1} = \mathbf{k}^\top \mathbf{C}_N^{-1}\mathbf{y}_N, \quad (4)$$

$$s_{y_{N+1}}^2 = \kappa - \mathbf{k}^\top \mathbf{C}_N^{-1}\mathbf{k}. \quad (5)$$

Further details can be found in (Buche et al., 2005).

Covariance Functions. The covariance $\mathbf{C}_N$ plays a crucial role in these equations. It is defined by the covariance-function matrix $\mathbf{K}$ and signal noise σ as

$$\mathbf{C}_N = \mathbf{K}_N + \sigma\mathbf{I}_N \quad (6)$$

where $\mathbf{I}_N$ is the identity matrix of order N. Gaussian processes use parametrized covariance functions K describing prior assumptions on the shape of the modelled function. The covariance between the function values at two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is given by $K(\mathbf{x}_i,\mathbf{x}_j)$, which forms the (i,j)-th element of the matrix $\mathbf{K}_N$. We used the most common squared-exponential function

$$K(\mathbf{x}_i,\mathbf{x}_j) = \theta\exp\left(-\frac{1}{2\ell^2}(\mathbf{x}_i-\mathbf{x}_j)^\top(\mathbf{x}_i-\mathbf{x}_j)\right), \quad (7)$$

which is suitable when the modelled function is rather smooth. The closer the points $\mathbf{x}_i$ and $\mathbf{x}_j$ are, the closer the covariance function value is to 1 and the stronger correlation between the function values $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ is. The signal variance θ scales this correlation, and the parameter ℓ is the characteristic length-scale with which the distance of two considered data points is compared. Our choice of the covariance function was motivated by its simplicity and the possibility of finding the hyper-parameter values by the maximum-likelihood method.

3 MODEL GUIDED SAMPLING OPTIMIZATION (MGSO)

The MGSO algorithm is based on a similar idea as EGO. It heavily relies on Gaussian process modelling, particularly on its regression capabilities and ability to assess model uncertainty in any point of the input space.

While most variants of EGO calculate new points from the *expected improvement* (EI), The MGSO utilizes the *probability of improvement* which is closer to the basic concept of the MGSO: sampling a distribution of promising solutions[2].

This probability is taken as a function proportional to a probability density and is sampled producing a whole population of candidate solutions – individuals. This is the main difference to EGO which takes only very few individuals each iteration, usually the point maximizing EI.

3.1 Sampling

The core step of the MGSO algorithm is the sampling of the probability of improvement. This probability is, for a chosen threshold T of the function value, directly given by the predicted mean $\hat{f}(\mathbf{x}) = \hat{y}$ and the standard deviation $\hat{s}(\mathbf{x}) = s_y$ of the GP model $\hat{f}$ in any point $\mathbf{x}$ of the input space

$$\mathrm{PoI}_T(\mathbf{x}) = \mathrm{P}(\hat{f}(\mathbf{x}) \leqq T) = \Phi\left(\frac{T - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right), \quad (8)$$

which corresponds to the value of cumulative distribution function (CDF) of the Gaussian with density (3) for the value of threshold T. As a threshold T, values near the so-far optimum (or the global optimum if known) are usually taken.

Even though all the variables come from Gaussian distribution, $\mathrm{PoI}(\mathbf{x})$ is definitely not Gaussian-shaped since it depends on the threshold T and the black-box function being modelled f. A typical example of the landscape of $\mathrm{PoI}(\mathbf{x})$ in two dimensions for the Rastrigin function is depicted in Fig. 1. The dependency on the black-box function also causes the lack of analytical marginals or derivatives.

3.2 MGSO Procedure

The MGSO algorithm starts with an initial random sample from the input space forming an initial population, which is evaluated with the black-box objective function (step (2) in Alg. 1). All the evaluated solutions are saved to a database from where they are used as a training set for the GP model.

[2]some EGO variants use PoI, too (Jones, 2001)

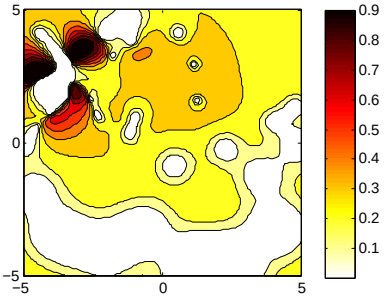

Figure 1: Probability of improvement. Rastrigin function in 2D, the GP model is built using 40 data points.

Algorithm 1: MGSO (Model Guided Sampling Optimization).

1: **Input**: f – black-box objective function
   N – standard population size
   r – the number of solutions for dataset restriction
2: $\mathsf{S}_0 = \{(\mathbf{x}_j, y_j)\}_{j=1}^{N} \leftarrow$ generate N initial samples and evaluate them by f: $y_j = f(\mathbf{x}_j)$
3: **while** no stopping condition is met, for $i = 1, 2, \ldots$ **do**
4: $\mathsf{M}_i \leftarrow$ build a GP model and fit its hyper-parameters according to the dataset S_{i-1}
5: $\{\mathbf{x}_j\}_{j=1}^{N} \leftarrow$ sample the $\mathrm{PoI}_T^{\mathsf{M}_i}(\mathbf{x})$ forming N new points, optionally with different targets T
6: $\mathbf{x}_{\min} \leftarrow \arg\min_{\mathbf{x}} \hat{f}(\mathbf{x})$ – find the minimum of the GP (by local cont. optimization) and replace the nearest solution from $\{\mathbf{x}_j\}_{j=1}^{N}$ with it
7: $\{y_j\}_{j=1}^{N} \leftarrow f(\{\mathbf{x}_j\}_{j=1}^{N})$ {evaluate the population}
8: $\mathsf{S}_i \leftarrow \mathsf{S}_{i-1} \cup \{(\mathbf{x}_j, y_j)\}_{j=1}^{N}$ {augment the dataset}
9: $(\mathbf{x}_{\min}, y_{\min}) \leftarrow \arg\min_{(\mathbf{x},y)\in\mathsf{S}_i} y$ {find the best solution in S_i}
10: **if** any rescale condition is met **then**
11: restrict the dataset S_i to the bounding-box $[\mathbf{l}_i, \mathbf{u}_i]$ of the r nearest solutions along the best solution $(\mathbf{x}_{\min}, y_{\min})$ and linearly transform S_i to $[-1, 1]^D$
12: **end if**
13: **end while**
14: **Output**: the best found solution $(\mathbf{x}_{\min}, y_{\min})$

The main cycle of the MGSO starts with *fitting* the GP model's (M_i) hyper-parameters based on the data from the current dataset S_i (step (4)). Further, the model's $\mathrm{PoI}_T^{\mathsf{M}_i}$ is *sampled* (step (5)) and supplemented with the GP model's minimum (step (6)), forming up to N new individuals $\{\mathbf{x}_j\}_{j=1}^{N}$ where N is a parameter defining the standard population size. The algorithm follows up with the evaluation of the new individuals with the objective function (step (7)), extending the dataset of already-measured samples (step (8)) and finding the best so-far optimum $(\mathbf{x}_{\min}, y_{\min})$ (step (9)).

Covariance Matrix Constraint. As every covariance matrix, Gaussian process' covariance matrix is required to be *positive semi-definite* (PSD). This constraint is checked during sampling, and candidate so-

lutions leading to close-to-indefinite matrix are rejected. Although it could cause smaller population-size in some iterations, it is an important step: otherwise, Gaussian process training and fitting becomes numerically very unstable. If such rejections arise, other two thresholds T for calculating PoI are tested and population with the greatest cardinality is taken. These rejections occur when the GP model is sufficiently trained and sampled solutions become close to the model's predicted values.

Model Minimum. New population is supplemented with the minimum $\mathbf{x}_{\min}$ of the model's predicted values found by continuous optimization[3] (step (6), $\mathbf{x}_{\min} = \arg\min_{\mathbf{x}} \hat{f}(\mathbf{x})$); more precisely, the nearest sampled solution is replaced with this minimum (unless less than N solutions were sampled).

Input Space Restriction. In current implementation, MGSO requires bounds constraints $\mathbf{x} \in [\mathbf{l}, \mathbf{u}]$, $\mathbf{l} < \mathbf{u} \in \mathbb{R}^D$ to be defined on the input space, which is used by the algorithm to internal linear scaling of the input space to hypercube $[-1,1]^D$. As the algorithm proceeds, the input space can be restricted along the so-far optimum to a smaller bounding box (steps (10)–(12)) which is again linearly scaled to $[-1,1]^D$. The size of the new box is defined as a bounding box of r nearest samples from the current so-far optimum $\mathbf{x}_{\min}$; enlarged by 10% at the boundaries. For the parameter $r = 15 \cdot D$ was used as a rule of thumb.

This process not only speeds up model fitting and prediction (due to the smaller number of training data), but focuses the optimization along the best found solution and broaden small regions of non-zero PoI.

Several criteria are defined to launch such input space restriction, from which the most important is occurrence of numerous rejections in sampling due to close-to-indefinite covariance matrix. If the resulting new bounding box from the restriction is close to the previous box (the coordinates are not smaller than $[-0.8, 0.8]^D$), the input space restriction is not performed.

3.3 Gaussian Processes Implementation

Our Matlab implementation of the MGSO makes use of Gaussian Process Regression and Classification Toolbox (GPML Matlab code) – a toolbox accompanying Rasmussen's and Williams' monograph (Rasmussen and Williams, 2006). In the current version of the MGSO, Rasmussen's optimization and model fitting procedure `minimize` was replaced with `fmincon` from Matlab Optimization toolbox and with Hansen's Covariance Matrix Adaptation (CMA-ES) (Hansen and Ostermeier, 2001). These functions are used for the minimization of GP's negative log-likelihood in model's hyper-parameters fitting. Here, `fmincon` is generally faster, but CMA-ES is more robust and does not need a valid initial point.

The next improvement lies in the employment of the diagonal-matrix characteristic length-scale parameter in the squared exponential covariance function, sometimes also called covariance function with automatic relevance determination (ARD)

$$K^{\mathrm{ARD}}(\mathbf{x}_i, \mathbf{x}_j) = \theta \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \operatorname{diag}(\vec{\ell})\,(\mathbf{x}_i - \mathbf{x}_j)\right). \quad (9)$$

The length-scale parameter ℓ in (7) changes to $\operatorname{diag}(\vec{\ell})$ – a matrix with diagonal elements formed by the elements of the vector $\vec{\ell} \in \mathbb{R}^D$. The application of ARD was not straightforward, because hyper-parameters training tends to stuck in local optima. These cases were indicated by an extreme difference between the different scale-length parameter $\vec{\ell}$ components which resulted in poor regression capabilities. Therefore, constraints on maximum difference between components of $\vec{\ell}$ were introduced

$$\vec{\ell}_i \;\leq\; 2.5\, \|\mathrm{median}(\vec{\ell}) - \vec{\ell}_i\|.$$

4 EXPERIMENTAL RESULTS

This section comprises quantitative results from several tests of the MGSO as well as brief discussion of the usability of the algorithm. The current Matlab implementation of the MGSO algorithm[4] has been tested on three different benchmark functions from the BBOB testing set (Hansen et al., 2009): sphere, Rosenbrock and Rastrigin function in three different dimensionalities: 2D, 5D and 10D. For these cases, comparison with CMA-ES – current state of the art black-box optimization algorithm – and Tomlab's implementation of EGO[5] is provided.

The computational times are not quantified, but whereas CMA-ES performs in orders of tens of seconds, the running times of the MGSO and EGO reaches up to several hours. We consider this drawback acceptable since the primary use of the MGSO is

[3] Matlab's `fminsearch` was used

[4] the source is available at http://github.com/charypar/gpeda

[5] http://tomopt.com/tomlab/products/cgo/solvers/ego.php

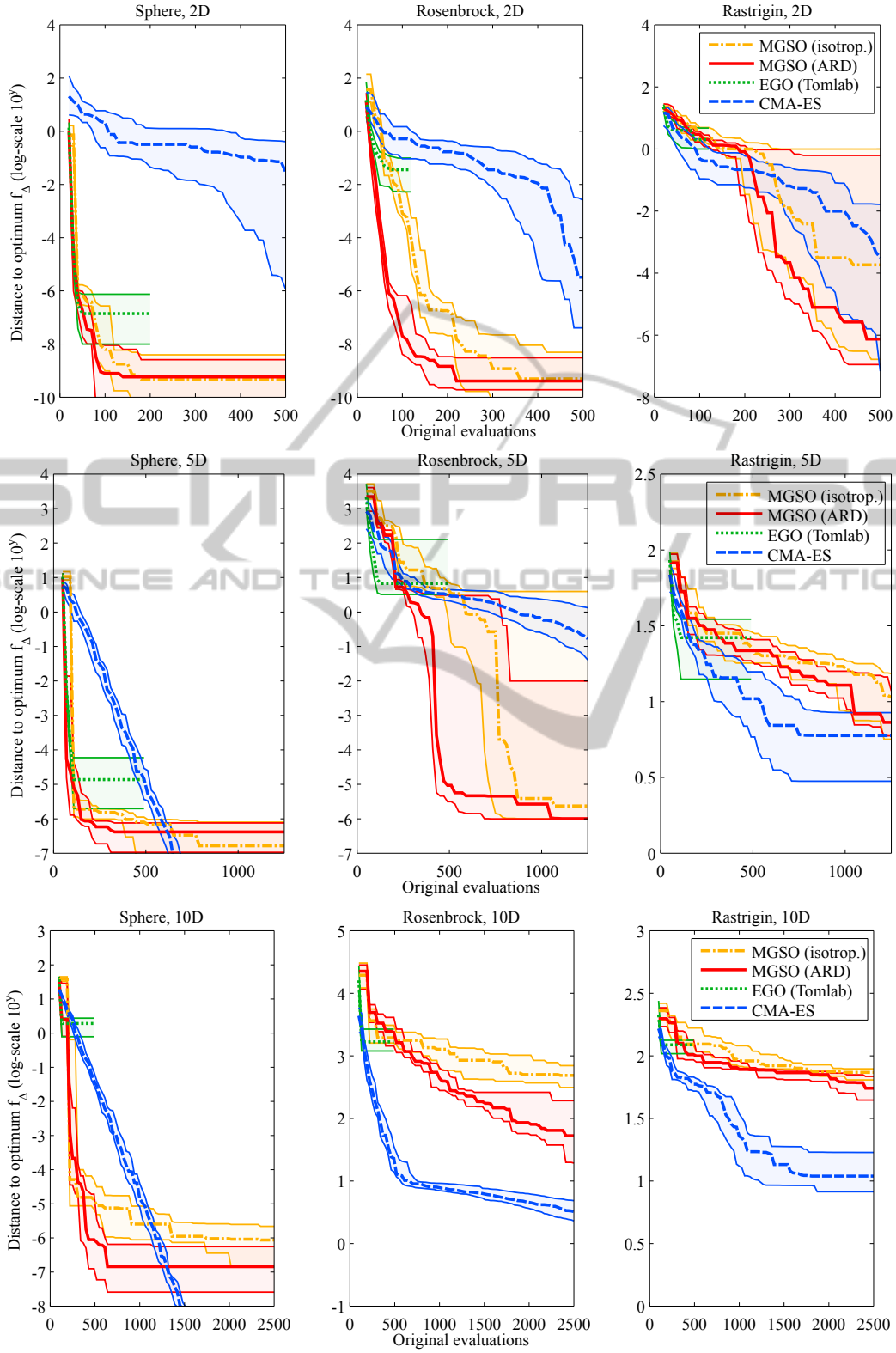

Figure 2: Medians, and the first and third quartiles of the distances from the best individual to optimum ($f_\Delta = f_{\text{best}} - f_{\text{OPT}}$) with respect to the number of objective function evaluations for three benchmark functions. Quartiles are computed out of 15 trials with different initial settings (instances 1–5 and 31–40 in the BBOB framework).

expensive black box optimization where a single evaluation of the objective function can easily take many hours or even days and/or cost a considerable amount of money (Holeňa et al., 2008).

In the case of *two-dimensional problems*, the MGSO performs far better than CMA-ES on quadratic sphere and Rosenbrock function. The results on Rastrigin function are comparable, although with greater variance (see Fig. 2: the descent of the medians is slightly slower within the first 200 function evaluations, but faster thereafter). The Tomlab's implementation of EGO performs almost equally well as the MGSO on sphere function, but on Rosenbrock and Rastrigin, the convergence of EGO is extremely slowed down after few iterations, which can be seen in 5D and 10D, too. The positive effect of ARD covariance function can be seen quite clearly, especially on Rosenbrock function. The difference between ARD and non-ARD results are hardly to see on sphere function, probably because its symmetry means no improvement in ARD covariance employment.

The performance of the MGSO on *five-dimensional problems* is similar to 2D cases. The MGSO descends notably faster on non-rugged sphere and Rosenbrock functions, especially if we concentrate on depicted cases with a very low number of objective function evaluations (up to $250 \cdot D$ evaluations). The drawbacks of the MGSO is shown on 5D Rastrigin function where it is outperformed by CMA-ES, especially between ca. 200 and 1200 function evaluations.

Results of optimization in the case of *ten-dimensional problems* show that the MGSO works better than CMA-ES only on the most smooth sphere function which is very easy to regress by Gaussian process model. On more complicated benchmarks, the MGSO is outperformed by CMA-ES.

The graphs on Fig. 2 show that the MGSO is usually slightly slower than EGO in the very first phases of the optimization, but EGO quickly stops its progress and does not descent further. This is exactly what can be expected from the MGSO in comparison to EGO – sampling the PoI instead of searching for the maximum can easily overcome situations where EGO gets stuck in a local optimum.

5 CONCLUSIONS AND FUTURE WORK

The MGSO, the optimization algorithm based on a Gaussian process model and the sampling of the probability of improvement, is intended to be used in the field of expensive black-box optimization. This paper summarizes its properties and evaluates its performance on several benchmark problems. Comparison with Gaussian-process based EGO algorithm shows that the MGSO is able to easily escape from local optima. Further, it has been shown that the MGSO can outperform state-of-the-art continuous black-box optimization algorithm CMA-ES in low dimensionalities or on very smooth functions. On the other hand, CMA-ES performs better on rugged or high-dimensional benchmarks.

ACKNOWLEDGEMENTS

This work was supported by the Czech Science Foundation (GAČR) grants P202/10/1333 and 13-17187S.

REFERENCES

Bajer, L., Holeňa, M., and Charypar, V. (2013). Improving the model guided sampling optimization by model search and slice sampling. In Vinar, T. e. a., editor, *ITAT 2013 – Workshops, Posters, and Tutorials*, pages 86–91. CreateSpace Indp. Publ. Platform.

Buche, D., Schraudolph, N., and Koumoutsakos, P. (2005). Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(2):183–194.

Hansen, N., Finck, S., Ros, R., and Auger, A. (2009). Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA. Updated February 2010.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Holeňa, M., Cukic, T., Rodemerck, U., and Linke, D. (2008). Optimization of catalysts using specific, description based genetic algorithms. *Journal of Chemical Information and Modeling*, 48:274–282.

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12.

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

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

Larrañaga, P. and Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Kluwer.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. MIT Press.