SACOBRA with Online Whitening for Solving Optimization Problems with High Conditioning

Technical Report

Samineh Bagheri[1], Wolfgang Konen[1], and Thomas Bäck[2]

[1] TH Köln – Univeristy of Applied Sciences, Gummersbach, Germany
{samineh.bagheri,wolfgang.konen}@th-koeln.de
[2] Leiden University, LIACS, Leiden, The Netherlands
t.h.w.baeck@liacs.leidenuniv.nl

Abstract. Real-world optimization problems often have expensive objective functions in terms of cost and time. It is desirable to find near-optimal solutions with very few function evaluations. Surrogate-assisted optimizers tend to reduce the required number of function evaluations by replacing the real function with an efficient mathematical model built on few evaluated points. Problems with a high condition number are a challenge for many surrogate-assisted optimizers including SACOBRA. To address such problems we propose a new online whitening operating in the black-box optimization paradigm. We show on a set of high-conditioning functions that online whitening tackles SACOBRA's early stagnation issue and reduces the optimization error by a factor between 10 to 10^{12} as compared to the plain SACOBRA, though it imposes many extra function evaluations. Covariance matrix adaptation evolution strategy (CMA-ES) has for very high numbers of function evaluations even lower errors, whereas SACOBRA performs better in the expensive setting ($\leq 10^3$ function evaluations). If we count all parallelizable function evaluations (population evaluation in CMA-ES, online whitening in our approach) as one iteration, then both algorithms have comparable strength even on the long run. This holds for problems with dimension $D \leq 20$.

Keywords: Surrogate models · high condition number · online whitening

1 Introduction

Optimization problems can often be defined as minimization of a black-box objective function $f(\boldsymbol{x})$. An optimization problem is called black-box if no analytical information about itself or its derivatives are given. Evolutionary algorithms including covariance matrix adaptation evolution strategy (CMA-ES) [9], genetic algorithm (GA) [22], differential evolution (DE) [16], and particle swarm optimization (PSO) [23] are among strong derivative-free algorithms suitable for handling black-box optimization problems. All the mentioned optimization algorithms are inspired from the evolution theory of Darwin and tend to evolve

a randomly generated initial population by means of different optimization operators (crossover, mutation, selection, estimating distribution etc.) iteratively. Despite all the significant contributions of differential evolution, solving problems with high-conditioning remains a challenge, as it is mentioned in [26]. In [21] a genetic algorithm is evaluated on a set of black-box problems and it is observed that the algorithm is weak in optimizing high conditioning problems. Despite many evolutionary-based algorithms, CMA-ES is very successful in tackling high-conditioning problems. The advantage of CMA-ES when solving problems with high conditioning stems from the fact that in each iteration the covariance matrix of the new distribution is adapted according to the evolution path which is the direction with highest expected progress. In other words, the covariance matrix adaptation aims to learn the Hessian matrix of the function in an iterative way.

Although the contribution of the mentioned evolutionary based algorithms is significant, they often require too many function evaluations which are not affordable in many real-world applications. That is because determining the value of the objective functions at a specific point $\boldsymbol{x}$ (set of variables) often requires to conduct a time-expensive simulation run. In order to solve expensive optimization problems in an efficient manner, several algorithms were developed which aim at reducing the number of function evaluations through the assistance of surrogate models [4,20,11].

Many of the recently developed surrogate-assisted optimization algorithms go – after an initialization step – through two main phases shown in Fig. 1. Phase I builds a cheap and fast mathematical model (surrogate) from the evaluated points. Phase II runs the optimization procedure *on the surrogate* to suggest a new infill point. The algorithm is sequential: as soon as the new infill point is evaluated on the real function, it will be added to the population of evaluated points and the surrogate will be updated accordingly. The two phases are repeated until a predefined budget of function evaluations is exhausted.

Clearly, the modeling phase has a significant impact on the performance of the optimizer. The surrogate-assisted optimization algorithm can be of no use, if the surrogate models are not accurate enough and do not lead the search to the interesting region. Therefore, it is very important to have an eye on the quality of the surrogates. Radial basis function interpolation (RBF) and Gaussian process (GP) models are commonly used for efficient optimization [2,11,1,3,7,14]. Although the mentioned techniques are suitable for modeling complicated nonlinear functions, both may face challenges in handling other aspects of functions. SACOBRA [3] is an optimization framework which uses RBFs as modeling technique. This algorithm is very successful in handling the commonly used constrained optimization problems, the so-called G-function benchmark [13].

However, it performs poorly when optimizing functions with a large condition number. A function, that has a high ratio of steepest slope in one direction to flattest slope in another direction, has a large condition number. We call this a function with *high conditioning*. The condition number of a function can be

determined as the ratio of the largest to smallest singular value of its Hessian matrix.

Shir et al.[24,25] observe that in high-conditioning problems CMA-ES may converge to the global optimum but fail to learn the Hessian matrix. They propose with FOCAL an efficient approach for determining the Hessian matrix even for functions with high condition number.

The surrogate-assisted CMA-ES algorithms proposed in [14,5] use surrogates in a different way: Whole CMA-ES generations alternate between being generated on the real function or on the surrogate function. Which function is used is determined by the algorithm online during the optimization run, based on a certain accuracy criterion. It turns out that for high-conditioning functions the algorithm effectively uses only the real function. Thus it behaves equivalent to plain CMA-ES and does not use surrogates in the high-conditioning case.

This work focuses on surrogate-assisted optimization of functions with moderate or high condition numbers. In Sec. 2, we provide some illustrative insights *why* such functions are tricky to optimize with surrogate-assisted solvers due to modeling difficulties. Sec. 3 gives a brief description of the SACOBRA algorithm. Then we describe the newly proposed online whitening scheme added to SACOBRA for boosting up the model performance. The experimental setup and the results on the noiseless single-objective BBOB benchmark [8] are described in Sec. 4 and 5, resp. Sec. 6 concludes.

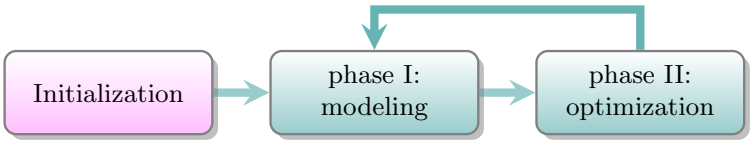

Fig. 1. Conceptualization flowchart of surrogate-assisted optimization

2 Why High Conditioning Is A Problem For Surrogates

In order to investigate the behavior of the RBF interpolation technique for modeling functions with high conditioning, we take a closer look at the second function $F02$ from the BBOB benchmark [8]:

$$F_{02}(\boldsymbol{x}) = \sum_{i=1}^{D} \alpha_i z_i^2 = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} z_i^2 \tag{1}$$

where $\boldsymbol{z} = T_{osz}(\boldsymbol{x} - \boldsymbol{x}^*)$ and $T_{osz}(\boldsymbol{x})$ is a nonlinear transformation [8], used to make the surface of $F_{02}(\boldsymbol{x})$ uneven without adding any extra local optima.

This function can be defined in any D-dimensional space. The large difference between the weights of the lowest variable x_1 to the highest x_D results in the high condition number of 10^6.

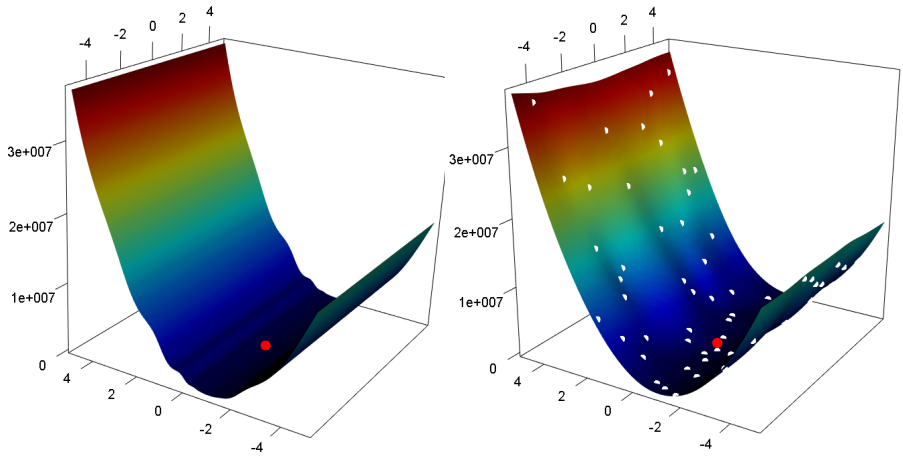

Fig. 2. $F02$ function from the BBOB benchmark (ellipsoidal function). Left: The real function. Right: RBF model for $F02$ built from 60 points (white points). The red point shows the location of the optimal solution.

Fig. 2, left, shows how $F_{02}(\boldsymbol{x})$ looks like for D=2. It is easy to see that $F_{02}(\boldsymbol{x})$ has steep walls in one direction but looks pretty flat in the other direction. Fig. 2, right, is the surrogate determined with a cubic RBF on 60 points (white dots).[3] We can see that the steep walls are reasonably well modeled but the surface is pretty wiggled. At first glance, it is not clear where the weakness of such model is.

In order to have a closer insight and also to be able to visualize higher-dimensional versions of $F_{02}(\boldsymbol{x})$ we plot cuts of the function along each dimension. Fig. 3 shows four cuts of $F_{02}(\boldsymbol{x})$ in the case $D = 4$ where $\boldsymbol{x}$ is a 4-dimensional vector. In this example the optimum is at $x^* = (-1, -1, -1, -1)$.

As one can see, the highest dimension x_4 with the largest coefficient $\alpha_4 = 10^6$ is very well modeled, but the model slices for lower dimensions do not follow the real function and do not contain any useful information about the location of the optimum.

It is important to mention that what makes $F_{02}(\boldsymbol{x})$ a function challenging to model is not the large or small coefficients for each dimension but the large variations of steepness in different directions.

Optimizing the surrogate model shown in Fig. 3 will result in a point $\boldsymbol{x}_{new}$, which has a near-optimal value for the steepest dimension but pretty much random values in all other dimensions.

[3] We note in passing that a GP model for $F02$ would look structurally very similar.

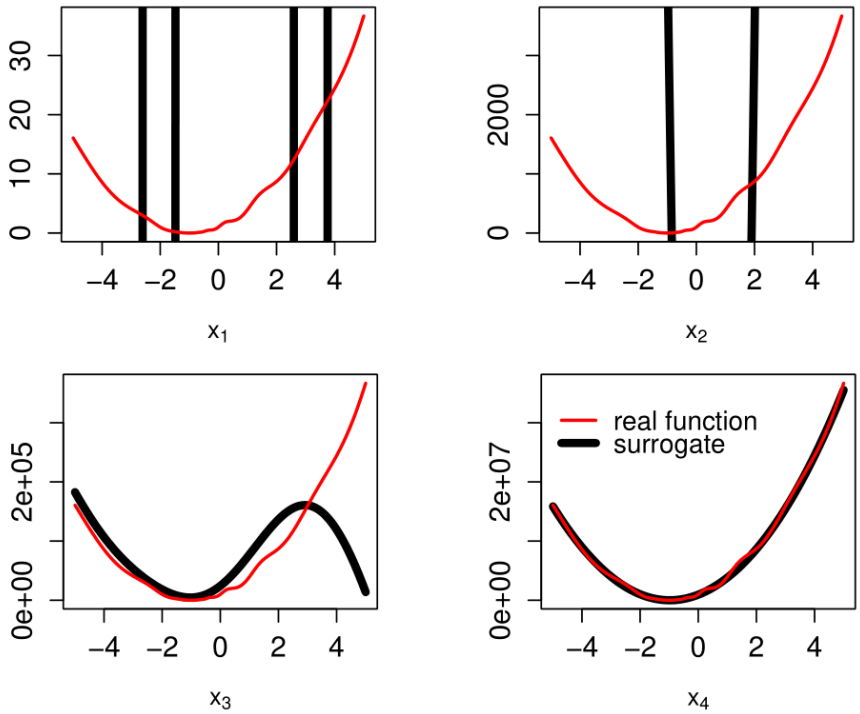

Fig. 3. Four cuts at the optimum $\boldsymbol{x}^*$ of the 4-dimensional function $F02$ (Eq. (1)) along each dimension. The red curve shows the real function and the black curve is the surrogate model. The black curve follows the red curve only in the 'steep' dimension x_4 (and to some extent in dimension x_3). Note the varying y-scales.

3 Methods

This work was motivated by applying the SACOBRA optimizer to the single-objective BBOB set of problems. Although we initially learned that SACOBRA performs poorly on problems with high and moderate conditioning, we investigated the underlying reason and came up with a cure: the so-called online whitening scheme.

3.1 SACOBRA: Self-Adjusting Constrained Optimization By RBF Approximation

SACOBRA, an extension of the COBRA algorithm [19], is a surrogate-assisted optimizer originally designed for high-dimensional constrained black box optimization problems [3], but also applicable to unconstrained problems. SACOBRA uses augmented RBF models (Sec. 3.2) as surrogates and then applies a constraint optimizer to solve the optimization problem on the model(s) (phase II in Fig. 1). The optimization result is evaluated on the real function and is added to the population of points. Then, the model(s) will be updated (phase

I) and the former steps will be repeated as long as the budget is not exhausted. SACOBRA uses self-adjusting techniques to tune sensitive parameters automatically [3]. Although SACOBRA appears to be strong in solving G-problems [13], it is weak on optimizing functions with high conditioning, mainly due to the modeling phase.

3.2 Augmented RBF

Augmented RBFs are linearly weighted combinations of radial basis functions and a polynomial tail as follows:

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_i \varphi(||\boldsymbol{x} - \boldsymbol{x}_{(i)}||) + p(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^D, \tag{2}$$

where $\{\boldsymbol{x}_{(i)} \in \mathbb{R}^D | i = 1, ...n\}$ is the current SACOBRA population and $p(\boldsymbol{x}) = \mu_0 + \boldsymbol{\mu}_1 \boldsymbol{x} + \boldsymbol{\mu}_2 \boldsymbol{x}^2 \cdots + \boldsymbol{\mu}_k \boldsymbol{x}^k$ is a k-th order polynomial in D variables with $kD + 1$ coefficients.

The augmented RBF model requires the solution of the following linear system of equations:

$$\begin{bmatrix} \boldsymbol{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{\mu}' \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \mathbf{0} \end{bmatrix} \tag{3}$$

Here, $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n}$ with $\boldsymbol{\Phi}_{ij} = \varphi(||\boldsymbol{x}_{(j)} - \boldsymbol{x}_{(i)}||)$ and $\mathbf{P} \in \mathbb{R}^{n \times (kD+1)}$ is a matrix with $(1, \boldsymbol{x}_{(i)}, \ldots, \boldsymbol{x}_{(i)}^k)$ in its ith row. $\mathbf{0} \in \mathbb{R}^{(kD+1) \times (kD+1)}$ is a zero matrix, $\boldsymbol{f}$ is a vector with $f(\boldsymbol{x}_{(i)})$ in its ith component and $\boldsymbol{\mu}'$ is the concatenation of the polynomial coefficients in $p(x)$.

In this work we use $\varphi(r) = r^3$ (cubic radial basis functions) with a second order polynomial tail ($k = 2$).

3.3 Online Whitening

As described in Section 2, functions with high conditioning are difficult to model for RBF or GP surrogates. Although the overall modeling error may be small, the models often have spurious local minima along the 'shallow' directions. This obviously hinders optimization. What we show here for RBF surrogate models holds the same way for GP (or Kriging) surrogate models often used in EGO [11]: Problems with a high condition number have a much higher optimization error than those with low conditioning (differing by a factor of 10^7 after 500 function evaluations, as some preliminary experiments have shown that we undertook with EGO using a Matern(3/2)-kernel).

In order to tackle high-conditioning problems with surrogate-assisted optimizers, we propose the online whitening scheme described in Algorithm 1: We seek to transform the objective function $f(\boldsymbol{x})$ with high conditioning to another function $g(\boldsymbol{x})$ which is easier to model by surrogates:

$$g(\boldsymbol{x}) = f(\mathbf{M}(\boldsymbol{x} - \boldsymbol{x}_c)), \tag{4}$$

Algorithm 1 Online whitening algorithm. Input: Function f to minimize, population $\mathbf{X} = \left\{\boldsymbol{x}_{(k)} | k = 1, \ldots, n\right\}$ of evaluated points, $\boldsymbol{x}_{best}$: best-so-far point from SACOBRA.

1: $\mathbf{H} \leftarrow$ Hessian matrix of function $f(\boldsymbol{x})$ at $\boldsymbol{x}_{best}$
2: $\mathbf{M} \leftarrow \mathbf{H}^{-0.5}$ {see Eq. (6) and Appendix B}
3: Update $\boldsymbol{x}_{best}$ with the function evaluations from Hessian calculation
Transformation :
4: $g(\boldsymbol{x}) \leftarrow f(\mathbf{M}(\boldsymbol{x} - \boldsymbol{x}_{best}))$
5: $\mathbf{G} \leftarrow \left\{\left(\boldsymbol{x}_{(k)}, g(\boldsymbol{x}_{(k)})\right) | k = 1, \ldots, n\right\}$ {evaluate all the points in $\mathbf{X}$ on the new function $g(\boldsymbol{x})$}
6: $s(\boldsymbol{x}) \leftarrow$ build surrogate model from $\mathbf{G}$
7: **return** $s(\boldsymbol{x})$ {surrogate model for next SACOBRA step}

where $\mathbf{M}$ is a linear transformation matrix and $\boldsymbol{x}_c$ is the transformation center. The ideal transformation center is the optimum point which is clearly not available. As a substitute, we use in selected iterations the best so-far solution $\boldsymbol{x}_{best}$ as the transformation center. The transformation matrix $\mathbf{M}$ is chosen in such a way that the Hessian matrix of the new function becomes the identity matrix:

$$\frac{\partial^2 g(\boldsymbol{x})}{\partial \boldsymbol{x}^2} = \mathbf{I} \tag{5}$$

It is derived in Appendix A that a solution for Eqs. (4) and (5) is given by:

$$\mathbf{M} = \mathbf{H}^{-0.5} \tag{6}$$

where $\mathbf{H}$ denotes the Hessian matrix of the objective function f.

Appendix B shows how to calculate $\mathbf{M}$ in a numerically stable way. The transformation matrix $\mathbf{M}$ used in our proposed algorithm is similar to the so-called Mahalanobis whitening or sphering transformation, which is commonly used in statistical analysis [12]. A *whitening or sphering transformation* aims at transforming a function in such a way that it has the same steepness in every direction, e. g. the height map of an ellipsoidal function will become spherical.

After determining the transformation matrix, we evaluate all points in the population $\mathbf{X}$ on the new function $g(\boldsymbol{x})$ and store the pairs $\left(\boldsymbol{x}_{(k)}, g(\boldsymbol{x}_{(k)})\right)$ in the set $\mathbf{G}$ (steps 4 and 5 in Algorithm 1). Then we re-build the surrogate model for $g(\boldsymbol{x})$ by passing the set $\mathbf{G}$ to the RBF model builder (step 6).

The Hessian matrix is determined numerically by means of Richardson's extrapolation [6] which requires $4D + 4D^2$ function evaluations. Initial tests have shown that an update of the Hessian matrix in each iteration of SACOBRA is not necessary. Thus we reduce the number of function evaluations by calling the online whitening scheme only every 10 iterations.

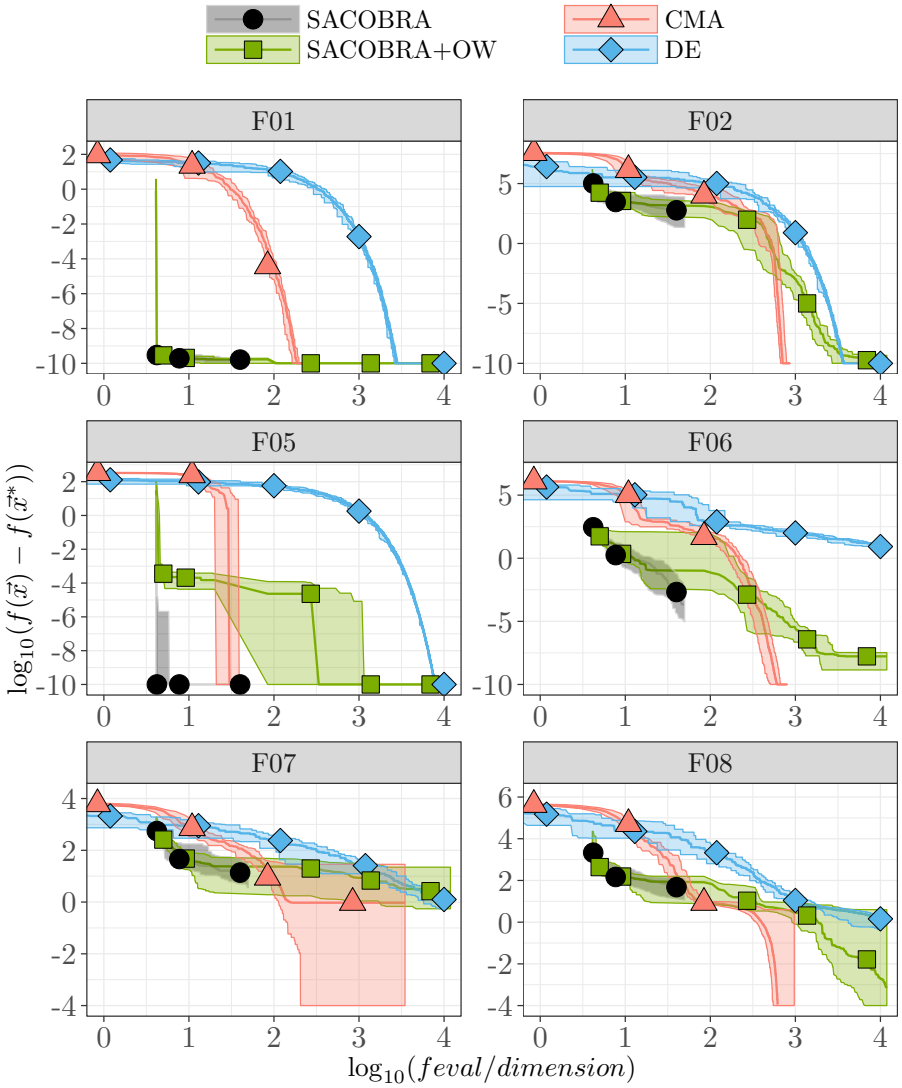

Fig. 4. Comparing the performance of SACOBRA, SACOBRA+OW, DE and CMA-ES algorithms on F01, F02, F05, F06, F07 and F08 optimization problems ($D = 10$).

4 Experimental Setup

We investigate the effectiveness of the online whitening scheme by comparing the standard SACOBRA algorithm (package SACOBRA in `R`) to SACOBRA combined with the online whitening scheme (SACOBRA+OW). To do so, we apply them to 12 of the 14 problems from the three first BBOB benchmarks [8].

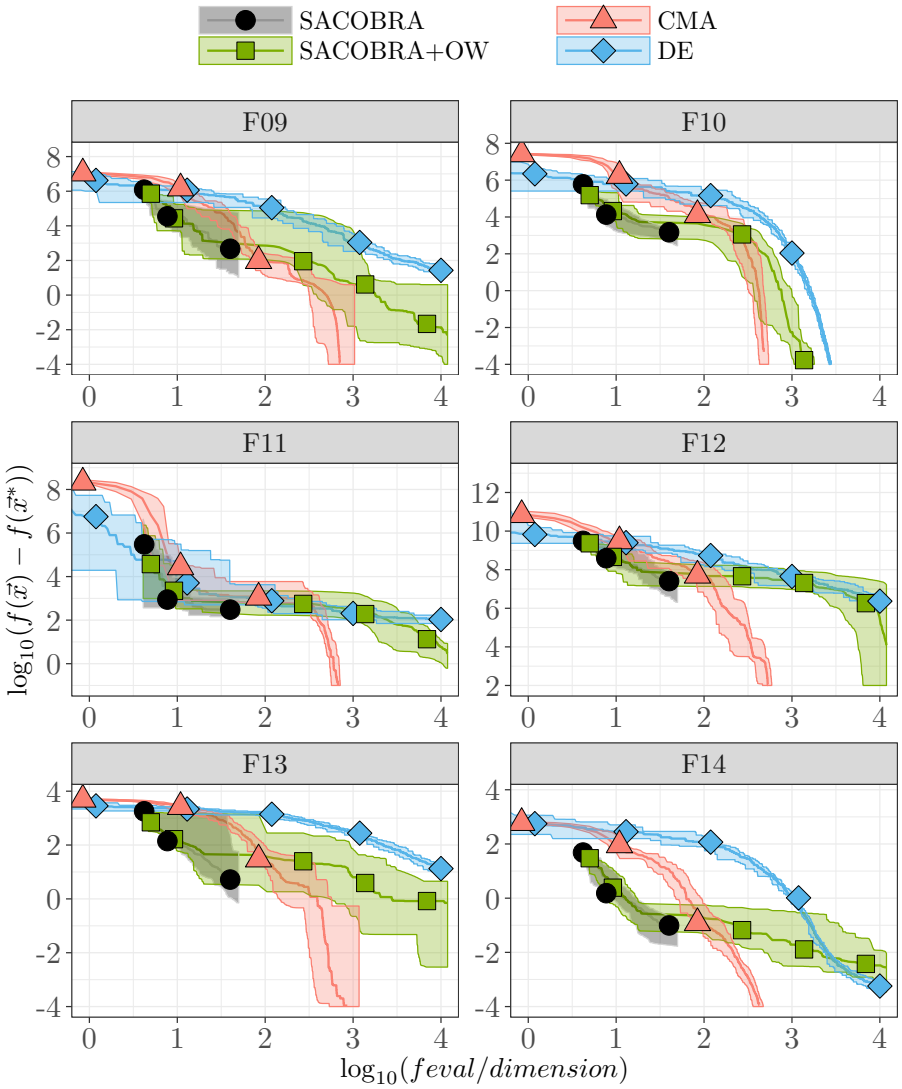

Fig. 5. Comparing the performance of SACOBRA, SACOBRA+OW, DE and CMA-ES algorithms on F09, F10, F11, F12, F13 and F14 optimization problems ($D = 10$).

We exclude two highly multimodal problems (F03 and F04), since they cannot be solved by surrogate modeling. Most of these benchmark functions have moderate to high condition numbers (see Table 1).

Both algorithms, SACOBRA and SACOBRA+OW, are compared as well to the differential evolution (DE) algorithm [18] and to the covariance matrix adaptation evolutionary strategy (CMA-ES) [9], using the DEoptim and rCMA

packages in R, resp. Both optimizers are used with their standard parameters. The default population size is in this case $10D$ and $4+3\lfloor\ln(D)\rfloor$ for the packages DEoptim and rCMA, respectively.

The two surrogate-assisted algorithms (SACOBRA and SACOBRA+OW) have an initial population size of $4D$ individuals. A maximum population size of $50D$ is permitted for both SACOBRA algorithms. It is important to mention that SACOBRA+OW may evaluate more than one point per iteration.

The online whitening scheme in SACOBRA+OW is first called after $20D$ iterations and it will be updated after each 10 iterations. The numerical calculation of the Hessian matrix is performed with the numDeriv package in R. In this work we mainly study and present results for the 10-dimensional problems. In the end, we compare the performance of all algorithms for 5- and 20-dimensional problems as well.

In order to compare the overall performance of different optimization algorithms on a set of problems we use data profiles [15]:

$$d_s(\alpha) = \frac{1}{|\mathbb{P}|}|\{p \in \mathbb{P} : \frac{t_{p,s}}{D_p} \leq \alpha\}|, \tag{7}$$

where $\mathbb{P}$ is a set of problems, $\mathbb{S}$ is a set of solvers and $t_{p,s}$ is the number of iterations that solver $s \in \mathbb{S}$ needs to *solve* problem $p \in \mathbb{P}$. D_p is the dimension of problem p. An optimization problem is said to be *solved* if a solution $\boldsymbol{x}_{best}$ is found whose objective value $f(\boldsymbol{x}_{best})$ deviates from the true solution $f(\boldsymbol{x}^*)$ less than a given tolerance τ:

$$|f(\boldsymbol{x}_{best}) - f(\boldsymbol{x}^*)| < \tau \tag{8}$$

Data profiles plot $d_s(\alpha)$ against α with $\alpha =$ feval/dimension.

5 Results & Discussion

Figs. 4–5 compare the optimization results achieved by SACOBRA, SACOBRA+OW, CMA-ES and DE on the BBOB benchmark problems listed in Table 1. Both SACOBRA and SACOBRA+OW become computationally expensive

Table 1. Condition numbers for all the investigated problems. The condition number is defined as the ratio of slope in the steepest direction to the slope in the flattest direction [10].

Function	Condition number	**Function**	Condition number
F01	1	F09	10^2
F02	10^6	F10	10^6
F05	1	F11	10^6
F06	10^3	F12	10^6
F07	10^2	F13	10^2
F08	10^2	F14	10^4

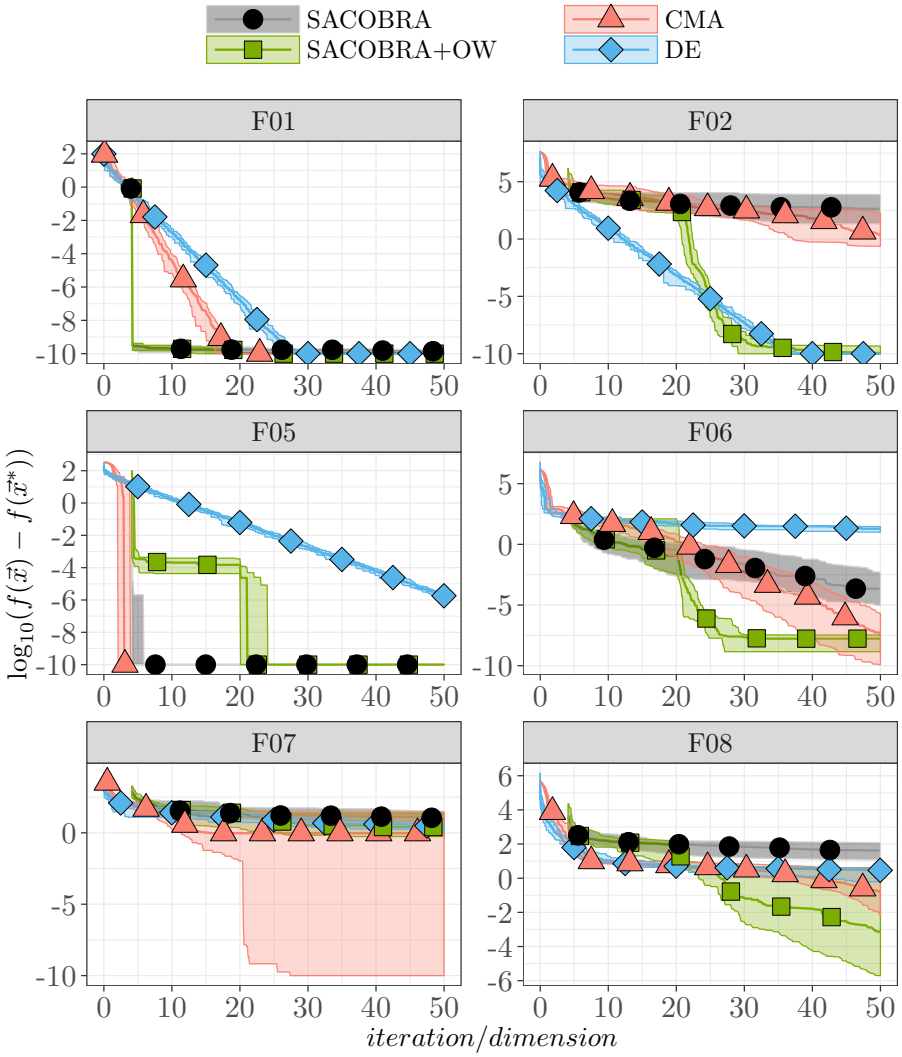

Fig. 6. Comparing the performance of SACOBRA, SACOBRA+OW, DE and CMA-ES algorithms on F01, F02, F05, F06, F07 and F08 optimization problems ($D = 10$). Now the x-axis shows iterations instead of function evaluations.

as the population size grows. Therefore we apply them for at most $50D$ iterations on each problem. This is the reason why all SACOBRA curves in Figs. 4–5 end at $1.7 = log_{10}(500/10)$ corresponding to a population size of 500. But SACOBRA+OW utilizes more real function evaluations when it starts to do the online whitening as described in Algorithm 1.

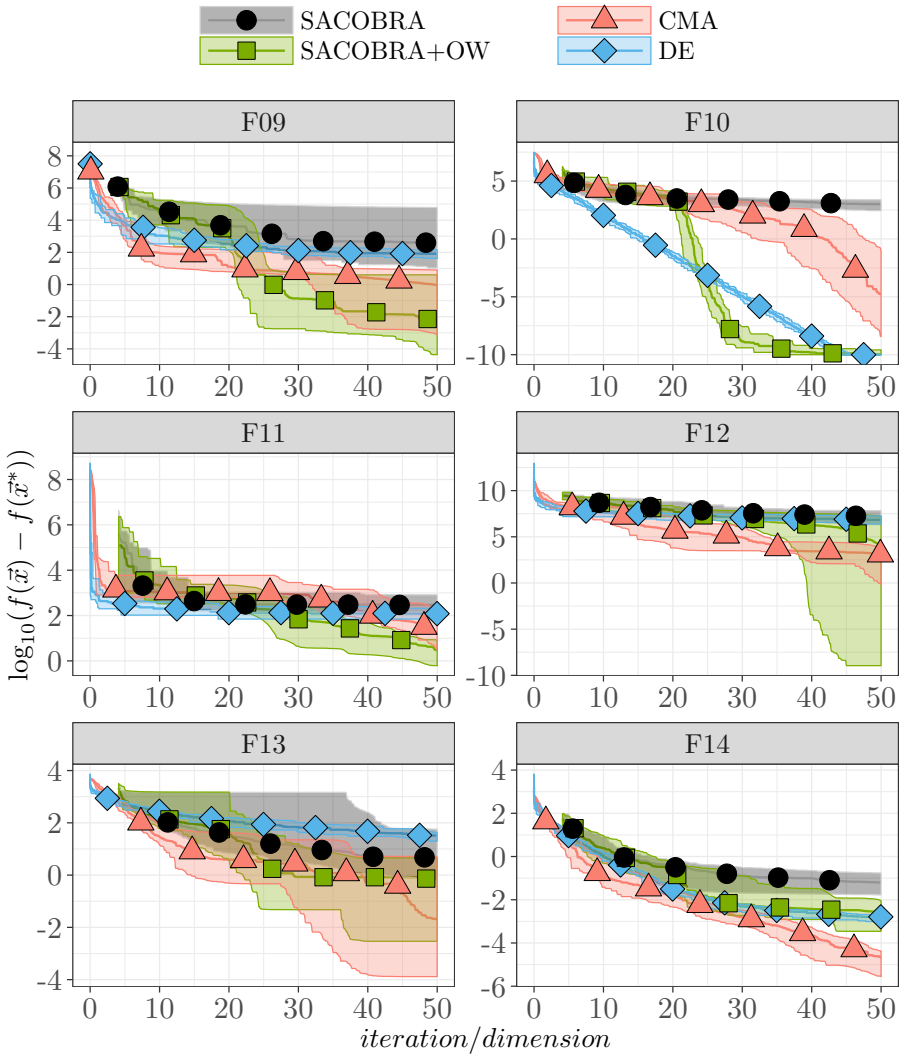

Fig. 7. Comparing the performance of SACOBRA, SACOBRA+OW, DE and CMA-ES algorithms on F09, F10, F11, F12, F13 and F14 optimization problems ($D = 10$). Now the x-axis shows iterations instead of function evaluations.

SACOBRA solves problems with low conditioning like F01 (sphere function) and F05 (linear slope) after very few function evaluations ($< 10D$) with a very high accuracy. CMA-ES and DE require 10 to 1000 times more function evaluations to find solutions as accurate as SACOBRA for these two problems. This strong performance of SACOBRA for F01 and F05 is probably due to the near-

perfect models that can be built with RBFs for such simple functions from just a few points.

However, for more complicated functions with high conditioning, SACOBRA often stagnates at a mediocre solution. Observing SACOBRA's behavior on high-conditioning functions in Figs. 4–5 indicates that, although SACOBRA has a fast progress in the first 100 iterations, it gradually becomes very slow and eventually stagnates. This is because the surrogates model only the steep walls reasonably well. Therefore, after being down in the valley between the steep walls, SACOBRA is effectively blind for the correct direction, and it suggests random points within the valley. This picture makes it clear – and experimental results confirm this – that it is of no use to add more points to the SACOBRA population, because the surrogate model stays wrong in all directions but the steepest ones.

SACOBRA+OW, which uses online whitening as a remedy for the modeling issues, can boost SACOBRA's optimization performance significantly. As it is shown in Figs. 4–5, SACOBRA+OW finds solutions whose optimization errors are between 10 times (in the case of F07) and 10^{12} times (in the case of F02) smaller than in SACOBRA.

Although SACOBRA and SACOBRA+OW have the same population sizes, the latter requires significantly more function evaluations due to the Hessian calculation in the whitening procedure.

This makes SACOBRA+OW no longer suitable for expensive optimization benchmarks, if the real world restrictions does not permit any form of parallelization of the Hessian matrix computation.

But it shows how to utilize surrogate models in cases with medium to high function evaluation budgets, which usually cannot be consumed completely by the surrogate model population.

Although SACOBRA+OW outperforms DE in 10 of 12 problems, it can compete with CMA-ES only when the function evaluation budget is 10^3 or less. Beyond this point, CMA-ES is usually the best algorithm.

Now we turn to the 'optimistic parallelizable' case: The numerical calculation of a Hessian matrix is not a sequential procedure and can be performed in parallel. Therefore, if enough computational resources are available, the Hessian matrix can be determined in the same time that a SACOBRA iteration needs. We call this the 'optimistic parallelizable' case. In this case, the efficiency of the SACOBRA+OW optimizer should be measured by its improvement per iteration (which need to be done one at a time). In the evolutionary strategies DE and CMA-ES, the evaluation of populations in each generation can be parallelized as well. So we count similarly all function evaluations needed to evaluate one DE- or CMA-ES-generation as *one* iteration, in order to establish a fair comparison.

Figs. 6–7 depict the optimization error *per iteration*[4] of SACOBRA, SACOBRA+OW, DE and CMA-ES for the BBOB problems listed in Tab. 1. We

[4] Each OW call is counted as one iteration, as well as each SACOBRA call. OW is first called at iteration $20D$ and then after each 10 SACOBRA iterations, one OW call is performed.

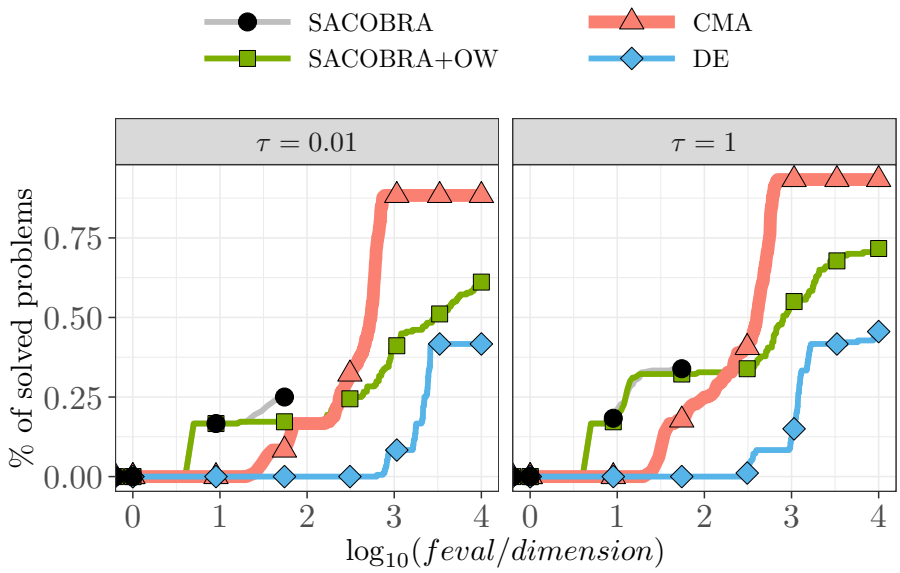

Fig. 8. Data profiles, Eq. (7), for the algorithms SACOBRA, SACOBRA+OW, DE and CMA-ES, showing the overall performance on 12 BBOB problems with dimension $D = 10$. The x-axis has the number of function evaluations, divided by D.

compare the performances of the mentioned algorithms within the first 500 iterations. As illustrated in Fig. 6–7, SACOBRA+OW appears to be the leading algorithm in terms of speed of convergence for 8 of the problems. F07 and F14 are the only problems for which CMA-ES can find significantly better solutions than SACOBRA+OW within the limit of 500 iterations. F05 and F13 can be optimized by CMA-ES and SACOBRA+OW similarly well. In general, SACOBRA+OW outperforms DE, although DE finds better solutions for F02 and F10 in the early iterations $1, \ldots, 250$ before SACOBRA+OW overtakes.

Fig. 8 compares the overall performance of the four investigated algorithms by means of data profiles (Sec. 4). It shows that the surrogate-assisted optimization is superior for low budgets (up to $100D$ function evaluations).

Fig. 8 indicates that SACOBRA can only solve 25% of the problems with accuracy $\tau = 0.01$, while SACOBRA+OW increases this ratio to about 62%. With the same accuracy level, our proposed algorithm can solve 25% more problems than DE but also about 25% less than CMA-ES.

Fig. 9 shows the data profiles for the 'optimistic parallelizable' case. Here SACOBRA+OW is consistently better than all other algorithms if we spent a budget of at most $50D$ iterations.

Fig. 10 compares the overall performances of the studied algorithms for the 5- and 20-dimensional case. While the results for the case $D = 5$ are similar to $D = 10$, the higher-dimensional case $D = 20$ shows that SACOBRA and

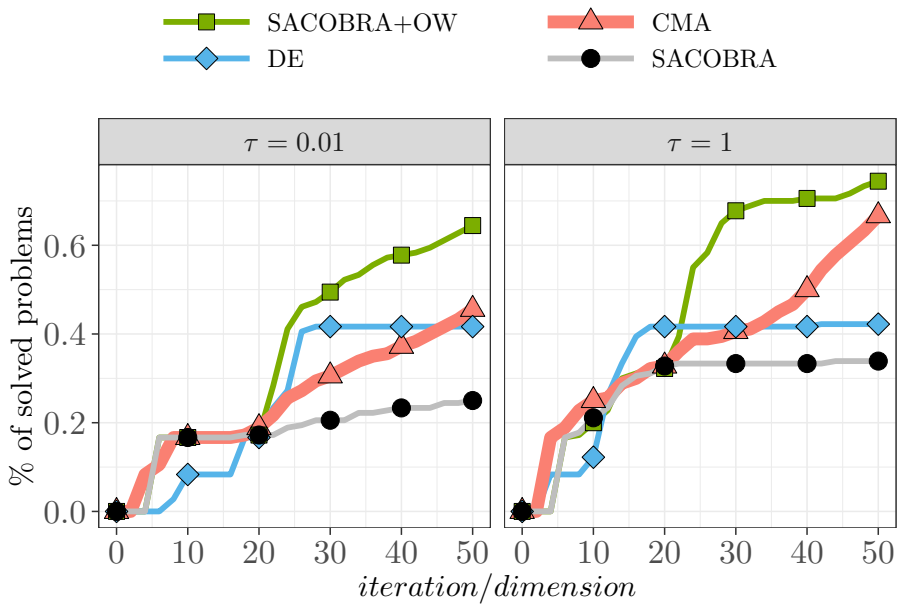

Fig. 9. Same as Fig. 8, but now for the 'optimistic parallelizable' case: We show on the x-axis the number of iterations (or generations), divided by D.

SACOBRA+OW as well as DE deteriorate notably. However, CMA-ES stays robust and performs best regardless of the dimensionality.

6 Conclusion

Surrogate-assisted optimizers are very fast solvers for linear or non-linear functions with low condition number. But they have severe difficulties when the function to optimize has a high condition number. Although we investigated here in detail only RBFs as surrogate models, we have given theoretical arguments that this holds as well for most types of surrogate models, namely for GP models[5].

We have proposed with SACOBRA+OW a new surrogate-assisted optimization algorithm with online whitening (OW) which aims at transforming online a high-conditioning into a low-conditioning problem. The method OW is applicable to all types of surrogates, not only to RBFs.

The results are encouraging in the sense that SACOBRA+OW finds better solutions than SACOBRA with the same population size. The percentage of solved problems on a subset of the BBOB benchmark is more than doubled when enhancing SACOBRA with OW.

[5] and we have experimental evidence for GP from other runs not shown here

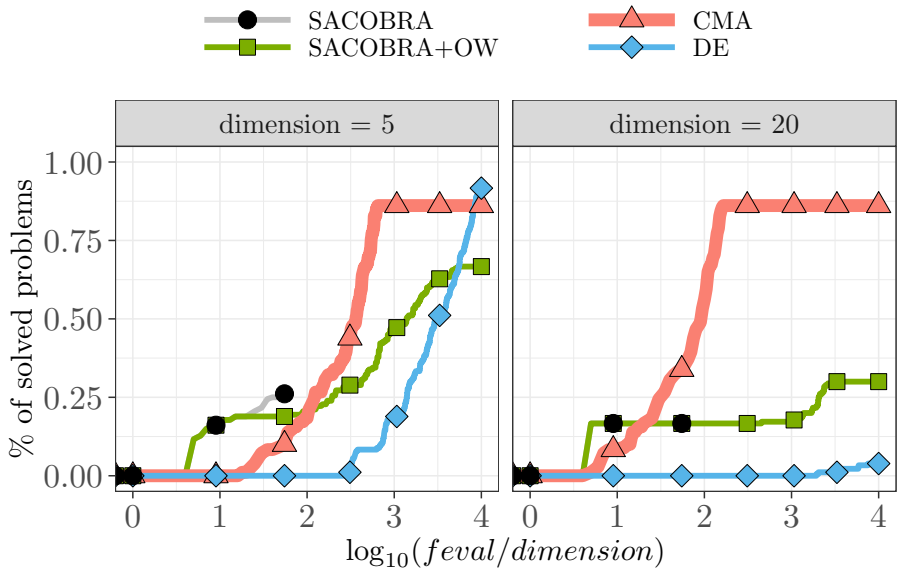

Fig. 10. Same as Fig. 8, but now for dimension $D = 5$ and $D = 20$. The accuracy level is set to $\tau = 0.01$.

Although for large budgets ($1000D$ function evaluations and more) SACOBRA+OW outperforms DE, it can no longer be considered as an optimizer for truly expensive problems because of the large number of function evaluations needed for determining the Hessian matrix. While SACOBRA is better for less than $100D$ function evaluations, CMA-ES finds consistently better solutions beyond this point, if we compare by number of function evaluations. But if we have the possibility for parallel computing of the Hessian matrix, then, if we compare by number of iterations, SACOBRA+OW appears to be the most efficient optimizer among the tested ones. In theory it is always possible to compute a Hessian matrix in parallel but in practice parallelizing this procedure is restricted to the amount of available resources. For example, if the objective function to optimize is evaluated through a time-expensive simulation run, then $4D + 4D^2$ computational cores running in parallel will be required for determining the Hessian matrix in one call. This can be an unrealistic demand when the number of dimensions D is higher.

Another limitation of SACOBRA+OW is that it currently only works well for dimensions $D \leq 20$.

We plan to investigate whether a combination of CMA-ES and surrogate-assisted optimizers could lead to an optimizer which combines 'the best of both worlds'. The efficient Hessian estimation of FOCAL [24,25] might be an interesting starting point for this.

7 Appendix

A Derivation of the Transformation Matrix

Let us assume that the objective function $f(\boldsymbol{x})$ is continuous and at least two times differentiable. Its Hessian (matrix of second derivatives) is $\frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x}^2} = \mathbf{H}$. Here and in the following all partial derivatives are meant to be evaluated at $\boldsymbol{x} = \boldsymbol{x}_c$, but we suppress this for better readability. $\boldsymbol{x}_c$ is the transformation center defined in Eq. (4).

We show that there is a transformation matrix $\mathbf{M}$ in such a way that the new function $g(\boldsymbol{x}) = f(\mathbf{M}(\boldsymbol{x} - \boldsymbol{x}_c))$ becomes spherical, so that its Hessian is $\frac{\partial^2 g(\boldsymbol{x})}{\partial \boldsymbol{x}^2} = \mathbf{I}$. We calculate the derivatives as:

$$\frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial f(\boldsymbol{u})}{\partial \boldsymbol{x}} \tag{9}$$

$$= \frac{\partial f(\boldsymbol{u})}{\partial \boldsymbol{u}} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} \tag{10}$$

$$= \frac{\partial f(\boldsymbol{u})}{\partial \boldsymbol{u}} \cdot \mathbf{M}^T, \tag{11}$$

where $\boldsymbol{u} = M(\boldsymbol{x} - \boldsymbol{x}_c)$ and hence $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} = \frac{\partial (M(\boldsymbol{x} - \boldsymbol{x}_c))}{\partial \boldsymbol{x}} = \mathbf{M}^T$.

$$\frac{\partial^2 g(\boldsymbol{x})}{\partial \boldsymbol{x}^2} = \frac{\partial (\frac{\partial f(\boldsymbol{u})}{\partial \boldsymbol{u}} \cdot \mathbf{M}^T)}{\partial \boldsymbol{x}} \tag{12}$$

$$= \frac{\partial (\frac{\partial f(\boldsymbol{u})}{\partial \boldsymbol{u}} \cdot \mathbf{M}^T)}{\partial \boldsymbol{u}} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} \tag{13}$$

$$= \frac{\partial (\frac{\partial f(\boldsymbol{u})}{\partial \boldsymbol{u}} \cdot \mathbf{M}^T)}{\partial \boldsymbol{u}} \cdot \mathbf{M}^T \tag{14}$$

We abbreviate $\frac{\partial f(\boldsymbol{u})}{\partial \boldsymbol{u}} = \boldsymbol{P}(\boldsymbol{u})$ and can derive

$$\frac{\partial^2 g(\boldsymbol{x})}{\partial \boldsymbol{x}^2} = \frac{\partial \boldsymbol{P} \mathbf{M}^T}{\partial \boldsymbol{P}} \cdot \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{u}} \cdot \mathbf{M}^T \tag{15}$$

$$= \mathbf{M} \cdot \frac{\partial^2 f(\boldsymbol{u})}{\partial \boldsymbol{u}^2} \cdot \mathbf{M}^T \tag{16}$$

$$= \mathbf{M} \cdot \mathbf{H} \cdot \mathbf{M}^T \tag{17}$$

We want to ensure that $\frac{\partial^2 g(\boldsymbol{x})}{\partial \boldsymbol{x}^2} = \mathbf{I}$:[6]

[6] Strictly speaking, this can only be guaranteed if $g(\boldsymbol{x})$ is convex in $\boldsymbol{x}_c$. If $g(\boldsymbol{x})$ is concave in one or all dimensions, we have a saddle point or local maximum at $\boldsymbol{x}_c$. In this case, $\mathbf{I}$ has to be replaced by a diagonal matrix with some elements being -1 instead of 1. But the overall whitening argument remains the same.

$$\mathbf{I} = \mathbf{M} \cdot \mathbf{H} \cdot \mathbf{M}^T \tag{18}$$
$$\mathbf{M}^{-1} = \mathbf{H} \cdot \mathbf{M}^T \tag{19}$$
$$\mathbf{M}^{-1}(\mathbf{M}^T)^{-1} = \mathbf{H} \tag{20}$$
$$\mathbf{M}^T\mathbf{M} = \mathbf{H}^{-1} \tag{21}$$

A possible solution for the last equation is $\mathbf{M} = \mathbf{H}^{-0.5}$.

B Calculation of Inverse Square Root Matrix

We calculate the inverse square root matrix in a numerically stable way with the help of singular value decomposition (SVD) [17]. The symmetric matrix $\mathbf{H}$ has the SVD representation

$$\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{22}$$

with orthogonal matrices $\mathbf{U}$,$\mathbf{V}$ and diagonal matrix $\mathbf{D} = \mathrm{diag}(d_i)$ containing only non-negative singular values d_i. The inverse square root of $\mathbf{D}$ is

$$\mathbf{D}^{-0.5} = \mathrm{diag}(e_i) \quad \text{with} \quad e_i = \begin{cases} \frac{1}{\sqrt{d_i}} & \text{if} \quad d_i > 10^{-25} \\ 0 & \text{else} \end{cases} \tag{23}$$

If we define

$$\mathbf{M} = \mathbf{D}^{-0.5}\mathbf{V}^T \tag{24}$$

and use the fact that a positive-semidefinite $\mathbf{H}$ has $\mathbf{U} = \mathbf{V}$, then it is easy to show that plugging this $\mathbf{M}$ into Eq. (18) fulfills the equation.

References

1. Samineh Bagheri, Wolfgang Konen, Richard Allmendinger, Jürgen Branke, Kalyanmoy Deb, Jonathan Fieldsend, Domenico Quagliarella, and Karthik Sindhya. Constraint handling in efficient global optimization. In *Proc. Genetic and Evolutionary Computation Conference GECCO'17*, pages 673–680, New York, 2017. ACM.
2. Samineh Bagheri, Wolfgang Konen, and Thomas Bäck. Comparing Kriging and radial basis function surrogates. In Frank Hoffmann and Eyke Hüllermeier, editors, *Proc. 27. Workshop Computational Intelligence*, pages 243–259. Universitätsverlag Karlsruhe, November 2017.
3. Samineh Bagheri, Wolfgang Konen, Michael Emmerich, and Thomas Bäck. Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. *Applied Soft Computing*, 61:377 – 393, 2017.
4. Samineh Bagheri, Wolfgang Konen, Christophe Foussette, Peter Krause, Thomas Bäck, and Patrick Koch. SACOBRA: Self-adjusting constrained black-box optimization with RBF. In Frank Hoffmann and Eyke Hüllermeier, editors, *Proc. 25. Workshop Computational Intelligence*, pages 87–96. Universitätsverlag Karlsruhe, 2015.

5. Lukáš Bajer, Zbyněk Pitra, and Martin Holeňa. Benchmarking Gaussian processes and random forests surrogate models on the BBOB noiseless testbed. In *Proc. Genetic and Evolutionary Computation Conference GECCO'15*, pages 1143–1150, New York, 2015. ACM.
6. Douglas M. Bates and Donald G. Watts. *Nonlinear regression analysis and its applications*. Wiley series in probability and mathematical statistics. Wiley, New York [u.a.], 1988.
7. Kalyan Shankar Bhattacharjee, Hemant Kumar Singh, and Tapabrata Ray. Multi-objective optimization with multiple spatially distributed surrogates. *Journal of Mechanical Design*, 138(9):091401, 2016.
8. Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2009.
9. Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proc. of 1996 IEEE International Conference on Evolutionary Computation, Nayoya University, Japan*, pages 312–317, 1996.
10. Nikolaus Hansen, Raymond Ros, Nikolas Mauny, Marc Schoenauer, and Anne Auger. Impacts of Invariance in Search: When CMA-ES and PSO Face Ill-Conditioned and Non-Separable Problems. *Applied Soft Computing*, 11:5755–5769, 2011.
11. Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, December 1998.
12. Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 2017. accepted.
13. JJ Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, PN Suganthan, CA Coello Coello, and Kalyanmoy Deb. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics*, 41:8, 2006.
14. Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. *CoRR*, abs/1204.2356, 2012.
15. Jorge J. Moré and Stefan M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM J. Optimization*, 20(1):172–191, 2009.
16. Petr Pošík and Václav Klemš. Jade, an adaptive differential evolution algorithm, benchmarked on the bbob noiseless testbed. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 197–204, New York, NY, USA, 2012. ACM.
17. William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.
18. Kenneth Price, Rainer Storn, and Jouni A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, 2005.
19. Rommel G. Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243, 2014.
20. Rommel G. Regis. Trust regions in surrogate-assisted evolutionary programming for constrained expensive black-box optimization. In Rituparna Datta and Kalyanmoy Deb, editors, *Evolutionary Constrained Optimization*, pages 51–94. Springer, 2015.

21. Babatunde A Sawyerr, Aderemi O Adewumi, and M Montaz Ali. Benchmarking rcgau on the noiseless bbob testbed. *The Scientific World Journal*, 2015, 2015.
22. Babatunde A. Sawyerr, Aderemi O. Adewumi, and Montaz M. Ali. Benchmarking projection-based real coded genetic algorithm on bbob-2013 noiseless function testbed. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, pages 1193–1200, New York, NY, USA, 2013. ACM.
23. Nizin Saxena, Ashish Tripathi, K. K. Mishra, and A. K. Misra. Dynamic-pso: An improved particle swarm optimizer. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 212–219, May 2015.
24. Ofer M. Shir, Jonathan Roslund, Darrell Whitley, and Herschel Rabitz. Evolutionary Hessian learning: Forced optimal covariance adaptive learning (FOCAL). *CoRR (arXiv)*, abs/1112.4454, 2011.
25. Ofer M Shir, Jonathan Roslund, Darrell Whitley, and Herschel Rabitz. Efficient retrieval of landscape Hessian: Forced optimal covariance adaptive learning. *Physical Review E*, 89(6):063306, 2014.
26. Andrew M Sutton, Monte Lunacek, and L Darrell Whitley. Differential evolution and non-separability: using selective pressure to focus search. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1428–1435. ACM, 2007.