Self-adjusting surrogate-assisted optimization techniques for expensive constrained black box problems

Bagheri, S.

Chapter 9

SACOBRA for Functions with High-Conditioning

9.1 Outline

Up till now, in this dissertation we introduced two different surrogate-assisted optimizers for handling black box constrained optimization problems in an efficient manner with very few function evaluations. The analyses in the former chapters show that the introduced surrogate-assisted algorithms, especially SACOBRA, can efficiently find near-optimal solutions for a set of various COPs. However, almost all the tested problems have objective and constraint functions with low or moderate condition number. As will be shown in Sec. 9.3, a current roadblock for surrogate-assisted optimizers is optimizing functions with a high condition number.

Although the main concern of this dissertation is tackling expensive constrained optimization problems, in this chapter we study unconstrained optimization problems with a high condition number, in order to reduce the complexity.

We give a brief introduction about this chapter in Sec. 9.2. In Sec. 9.3, we provide some illustrative insights *why* functions with high condition number are tricky to optimize with surrogate-assisted solvers due to modeling difficulties. In Sec. 9.4, we propose a new online whitening algorithm for the SACOBRA framework which tries to improve SACOBRA's performance on ill-conditioned functions. This algorithm operates in the black box optimization paradigm and it adapts itself to new function evaluations. The experimental setup and the results on a subset of the noiseless single-objective BBOB benchmark [58] are described in Sec. 9.5 and 9.6, resp. We show on a set of high-conditioning functions that online whitening reduces the optimization error by a factor between 10 to 10^{12} as compared to the plain surrogate-assisted optimizer. Covariance matrix adaptation evolution strategy (CMA-ES) has for very high numbers of function evaluations even lower errors, whereas our approach performs better in the expensive setting ($\leq 10^3$ function evaluations). If

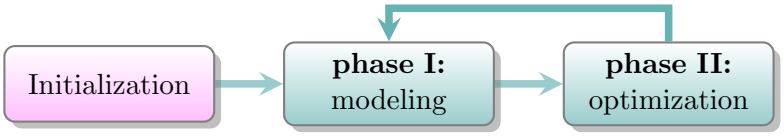

Figure 9.1: Conceptualization flowchart of surrogate-assisted optimization in [141, 15, 14].

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 relatively low dimension $d \leq 20$.

9.2 Introduction and Related Work

Optimization problems can be defined as minimization of a black-box objective function $f(\vec{x})$ as in Eq. (2.1). Evolutionary algorithms including covariance matrix adaptation evolution strategy (CMA-ES) [76], genetic algorithm (GA) [156], differential evolution (DE) [129], and particle swarm optimization (PSO) [94, 157] 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 [168]. In [155] 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. 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 $\vec{x}$ (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 [20, 142, 90].

Many of the recently developed surrogate-assisted optimization algorithms including SACOBRA and SOCU go – after an initialization step – through two main phases shown in Fig. 9.1. 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 [17, 90, 14, 19, 27, 108]. Although the mentioned techniques are suitable for modeling complicated non-linear functions, both may face challenges in handling other aspects of functions.

SACOBRA introduced in Ch. 3 and extended in Ch. 8 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 [107]. 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.[161, 162] 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 [108, 21] 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 mainly the real function. Thus it behaves equivalent to plain CMA-ES and does not use surrogates in the high-conditioning case.

This chapter focuses on optimizing functions with moderate or high condition numbers by means of the surrogate-assisted optimizer SACOBRA, extended with on online whitening mechanism which will be introduced in Sec. 9.4.

9.3 Why Is High Conditioning An Issue 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 function $F02$ from the BBOB benchmark:

$$F_{02}(\vec{x}) = \sum_{i=1}^{d} \alpha_i z_i^2 = \sum_{i=1}^{d} 10^{6\frac{i-1}{d-1}} z_i^2 \tag{9.1}$$

where $\vec{z} = T_{osz}(\vec{x} - \vec{x}^*)$ and $T_{osz}(\vec{x})$ is a nonlinear transformation [58], used to make the surface of $F_{02}(\vec{x})$ 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. 9.2, left, shows how $F_{02}(\vec{x})$ looks like for $d = 2$. It is easy to see that $F_{02}(\vec{x})$ has steep walls in one direction but looks pretty flat in the other direction. Fig. 9.2, right, is the surrogate determined with a cubic RBF on 60 points (white dots). 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 $F_{02}(\vec{x})$ we plot cuts of the function along each dimension. Fig. 9.3 shows four cuts of the 4-dimensional $F_{02}(\vec{x})$. In this example the optimum is at $\vec{x}^* = [-1, -1, -1, -1]$.

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. Optimizing the surrogate model shown in Fig. 9.3 will result in a point $\vec{x}_{new}$, which has a near-optimal value for the steepest dimension but pretty much random values in all other dimensions.

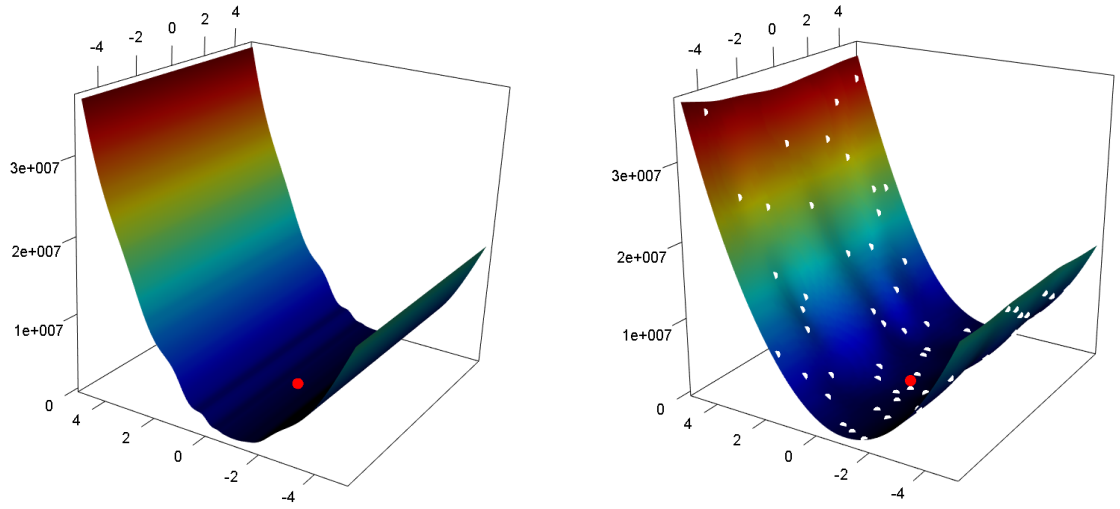

Figure 9.2: $F02$ function from BBOB benchmark set (ellipsoidal function). Left: The real function. Right: RBF model built for $F02$ with 60 points shown as white points. The red point on both plots shows the location of the optimal solution.

Algorithm 8 Online whitening algorithm. Input: Function f to minimize, population $\mathbf{X} = \left\{\vec{x}_{(k)} | k = 1, \ldots, n\right\}$ of evaluated points, $\vec{x}_{best}$: best-so-far point from SACOBRA.

1: $\mathbf{H} \leftarrow$ Hessian matrix of function $f(\vec{x})$ at $\vec{x}_{best}$
2: $\mathbf{M} \leftarrow \mathbf{H}^{-0.5}$ ▷ see Eq. (9.4) and Sec. 9.4.2
3: Update $\vec{x}_{best}$ with the function evaluations from Hessian calculation
4: *Transformation* :
5: $g(\vec{x}) \leftarrow f(\mathbf{M}(\vec{x} - \vec{x}_{best}))$
6: $\mathbf{G} \leftarrow \left\{\left(\vec{x}_{(k)}, g(\vec{x}_{(k)})\right) | k = 1, \ldots, n\right\}$ ▷ evaluate all the points in $\mathbf{X}$ on the new function $g(\vec{x})$
7: $s(\vec{x}) \leftarrow$ build surrogate model from $\mathbf{G}$ **return** $s(\vec{x})$ ▷ surrogate model for next SACOBRA step

9.4 Online Whitening Scheme for SACOBRA

This chapter was originally motivated by applying the SACOBRA optimizer, described in Ch. 3, to the single-objective BBOB set of problems. We investigated the underlying reason for the early stagnation of SACOBRA on ill-conditioned problems and came up with a cure: the so-called online whitening scheme. In this chapter,

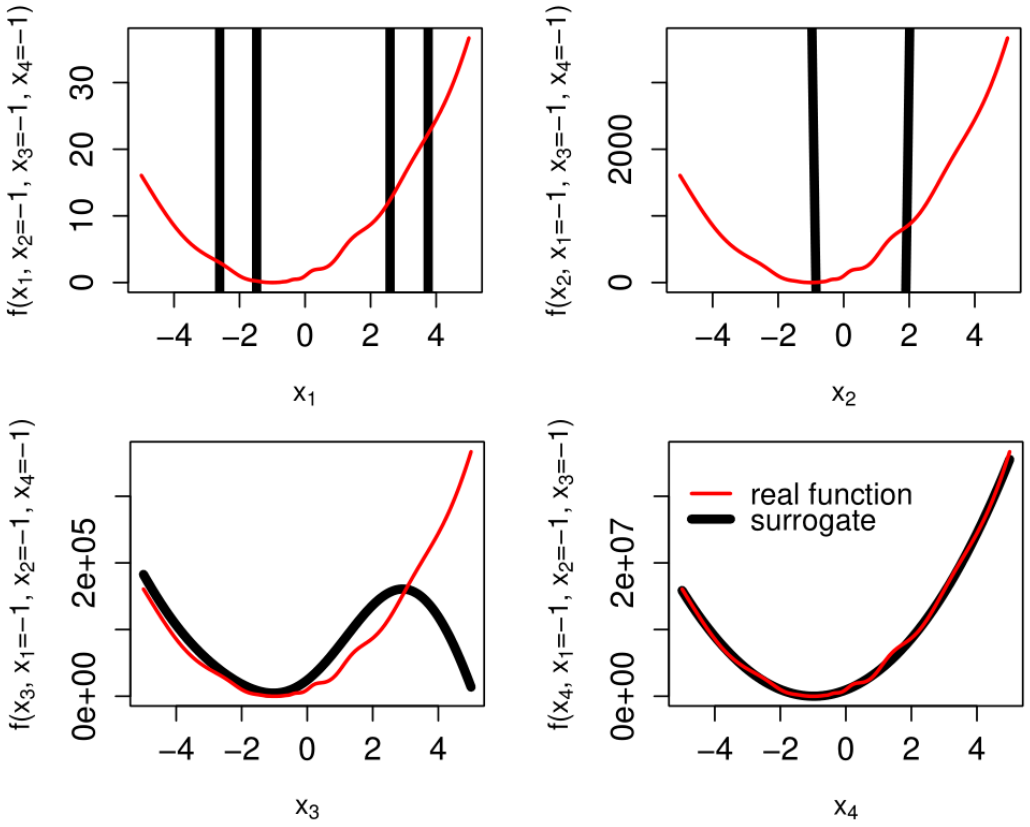

Figure 9.3: Four cuts at the optimum $\vec{x}^*$ of the 4-dimensional function $F02$ (Eq. (9.1)) along each dimension. The red curve shows the real function and the black curve is the surrogate model. The black curve follows the the red curve only in dimension x_4 (and to some extent in dimension x_3) where the function is very steep.

the used surrogate model in the SACOBRA framework is a $\varphi(r) = r^3$ (cubic radial basis functions) with a second order polynomial tail ($k = 2$), described in Ch. 2.

As shown in Sec. 9.3, functions with high conditioning are difficult to be modeled by 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 [90]: 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 8: We seek to transform the fitness function $f(\vec{x})$ with high conditioning to another function $g(\vec{x})$ which is easier to model by surrogates:

$$g(\vec{x}) = f(\mathbf{M}(\vec{x} - \vec{x}_c)), \tag{9.2}$$

where $\mathbf{M}$ is a linear transformation matrix and $\vec{x}_c$ is the transformation center. The ideal transformation center is the optimum point which is clearly not available. As a substitute, we use in each iteration the best so-far solution $\vec{x}_{best}$ 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:

$$\frac{\partial^2 g(\vec{x})}{\partial \vec{x}^2} \quad = \quad \mathbf{I} \tag{9.3}$$

In Sec. 9.4.1 we derive that solving Eqs. (9.2) and (9.3) results in the following equation:

$$\mathbf{M} \quad = \quad \mathbf{H}^{-0.5} \tag{9.4}$$

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

9.4.1 Derivation of the Transformation Matrix

Let us assume that the fitness function $f(\vec{x})$ is continuous and at least two times differentiable. Its Hessian (matrix of second derivatives) at $\vec{x}_c$ is $\frac{\partial^2 f(\vec{x})}{\partial \vec{x}^2} = \mathbf{H}$. Let us assume $\vec{x}_c = 0$ without loss of generality. We show that there is a transformation matrix $\mathbf{M}$ in such a way that the new function $g(\vec{x}) = f(\mathbf{M}\vec{x})$ becomes spherical, so that its Hessian is $\frac{\partial^2 g(\vec{x})}{\partial \vec{x}^2} = \mathbf{I}$. We calculate the derivatives as:

$$\frac{\partial g(\vec{x})}{\partial \vec{x}} = \frac{\partial f(\vec{u})}{\partial \vec{x}} \tag{9.5}$$

$$= \frac{\partial f(\vec{u})}{\partial \vec{u}} \cdot \frac{\partial \vec{u}}{\partial \vec{x}} \tag{9.6}$$

$$= \frac{\partial f(\vec{u})}{\partial \vec{u}} \cdot \mathbf{M}^T, \tag{9.7}$$

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

$$\frac{\partial^2 g(\vec{x})}{\partial \vec{x}^2} = \frac{\partial (\frac{\partial f(\vec{u})}{\partial \vec{u}} \cdot \mathbf{M}^T)}{\partial \vec{x}} \tag{9.8}$$

$$= \frac{\partial (\frac{\partial f(\vec{u})}{\partial \vec{u}} \cdot \mathbf{M}^T)}{\partial \vec{u}} \cdot \frac{\partial \vec{u}}{\partial \vec{x}} \tag{9.9}$$

$$= \frac{\partial (\frac{\partial f(\vec{u})}{\partial \vec{u}} \cdot \mathbf{M}^T)}{\partial \vec{u}} \cdot \mathbf{M}^T \tag{9.10}$$

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

$$\frac{\partial^2 g(\vec{x})}{\partial \vec{x}^2} = \frac{\partial \vec{P}\mathbf{M}^T}{\partial \vec{P}} \cdot \frac{\partial \vec{P}}{\partial \vec{u}} \cdot \mathbf{M}^T \tag{9.11}$$

$$= \mathbf{M} \cdot \frac{\partial^2 f(\vec{u})}{\partial \vec{u}^2} \cdot \mathbf{M}^T \tag{9.12}$$

$$= \mathbf{M} \cdot \mathbf{H} \cdot \mathbf{M}^T \tag{9.13}$$

We want to ensure that $\frac{\partial^2 g(\vec{x})}{\partial \vec{x}^2} = \mathbf{I}$:[1]

[1]Strictly speaking, this can only be guaranteed if $g(\vec{x})$ is convex in $\vec{x}_c$. If $g(\vec{x})$ is concave in one or all dimensions, we have a saddle point or local maximum at $\vec{x}_c$. 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.

$$\begin{aligned} \mathbf{I} &= \mathbf{M} \cdot \mathbf{H} \cdot \mathbf{M}^T && (9.14) \\ \mathbf{M}^{-1} &= \mathbf{H} \cdot \mathbf{M}^T && (9.15) \\ \mathbf{M}^{-1}(\mathbf{M}^T)^{-1} &= \mathbf{H} && (9.16) \\ \mathbf{M}^T\mathbf{M} &= \mathbf{H}^{-1} && (9.17) \end{aligned}$$

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

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

The Hessian matrix is determined numerically by means of Richardson's extrapolation [23] 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, to reduce the number of function evaluations, the online whitening scheme is called usually every 10 iterations.

9.4.2 Calculation of Inverse Square Root Matrix

In this section we show how $\mathbf{M}$ is calculated 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 [95]. 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.

The stable calculation of the inverse square root matrix is done with the help of singular value decomposition (SVD) [132]. The symmetric matrix $\mathbf{H}$ has the SVD representation

$$\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{9.18}$$

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

$$\mathbf{D}^{-0.5} = \mathrm{diag}(e_i) \quad \text{with} \quad e_i = \begin{cases} \frac{1}{\sqrt{d_i}} & \text{if} \quad d_i > 10^{-25} \\ 0 & \text{else} \end{cases} \tag{9.19}$$

If we define

$$\mathbf{M} = \mathbf{D}^{-0.5}\mathbf{V}^T \tag{9.20}$$

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. (9.14) fulfills the equation.

9.5 Experimental Setup

In this chapter, we investigate the effectiveness of the online whitening scheme by comparing the standard SACOBRA algorithm and SACOBRA with the online whitening scheme (SACOBRA+OW). To do so, we apply them to 12 problems from the three first BBOB benchmark categories, where 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 9.1).

Furthermore, our algorithm is compared to a differential evolution (DE) algorithm [133] and a covariance matrix adaptation evolutionary strategy (CMA-ES) [76] using the DEoptim and rCMA packages in `R`. Both optimizers are used with their standard parameters. The two surrogate-assisted algorithms (SACOBRA and SACOBRA+OW) have an initial population size of $4 \cdot d$ individuals. A maximum population size of $50 \cdot d$ 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 $20 \cdot d$ 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 chapter we mainly study and present results for the 10-dimensional problems. In the end,

Table 9.1: Condition numbers for all the investigated problems. The condition number is defined as the ratio of largest to smallest singular value of the Hessian matrix [77].

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

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 [118], also described in Sec. 2.5.

9.6 Results & Discussion

9.6.1 Convergence Curves

Fig. 9.4 compares the optimization results achieved by SACOBRA, SACOBRA+OW, CMA-ES and DE algorithms on the three first categories of the BBOB benchmark problems (excluding the multimodal problems F03 and F04). Both SACOBRA and SACOBRA+OW become computationally expensive 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 Fig. 9.4 end at $1.7 = log_{10}(500/10)$ corresponding to a population size of 500. But SACOBRA+OW makes use of more real function evaluations when it starts to do the online whitening scheme described in Algorithm 8.

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 Fig. 9.4 indicates that, although SACOBRA has 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 Fig. 9.4, 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

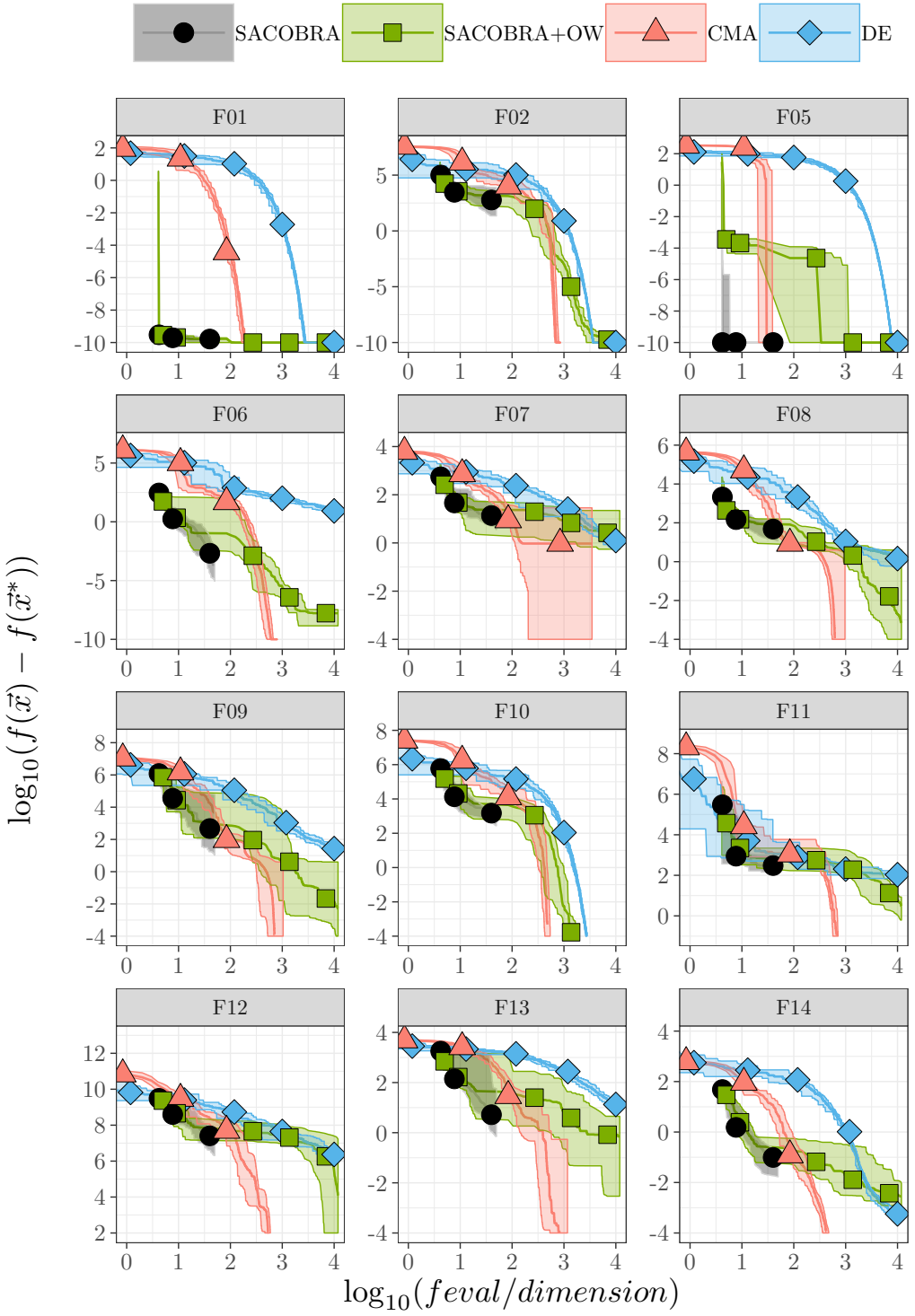

Figure 9.4: Comparing the performance of SACOBRA, SACOBRA+OW, CMA-ES and DE algorithms on 12 of the BBOB optimization problems ($d = 10$).

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 do not permit any form of parallelisation 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.

9.6.2 Parallel Computation

Numerical calculation of the Hessian matrix of a function 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.

Fig. 9.5 depicts the optimization error *per iteration*[2] determined by SACOBRA, SACOBRA+OW, DE and CMA-ES for the BBOB problems, listed in Tab. 9.1. We compare the performances of the mentioned algorithms within the first 500 iterations. As illustrated in Fig. 9.5, 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.

[2]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.

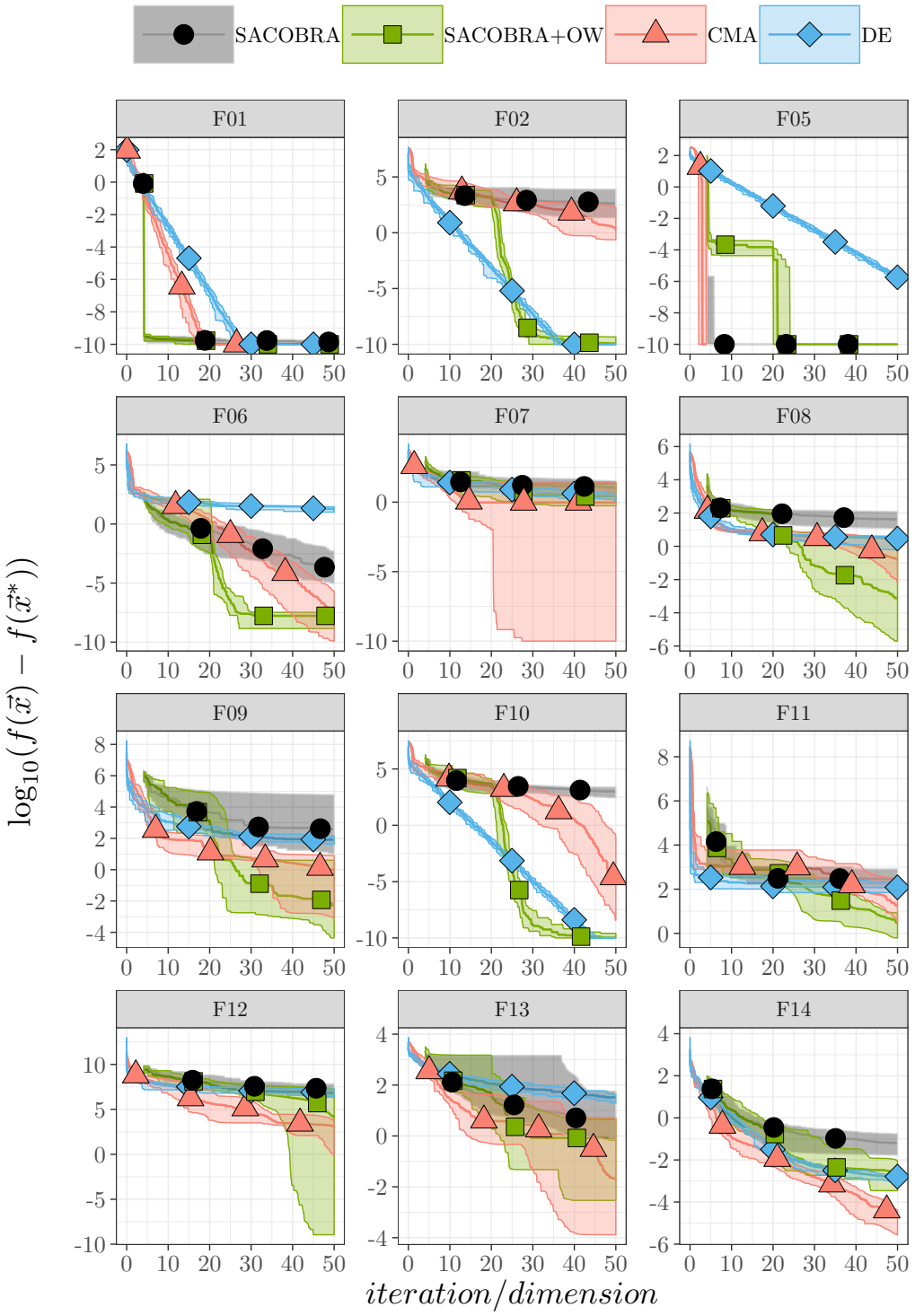

Figure 9.5: Comparing the performance of SACOBRA, SACOBRA+OW, CMA-ES and DE algorithms on 12 of the BBOB optimization problems ($d = 10$).

9.6.3 Data Profile

Figs. 9.6 and 9.7 compare the overall performance of the four investigated algorithms by means of data profiles (Sec. 2.5). Fig. 9.6 shows that surrogate-assisted optimization is superior for low budgets (up to $20d$ function evaluations), and Fig. 9.7 reveals that this advantage continues up to $100d$. Additionally, Fig. 9.7 indicates

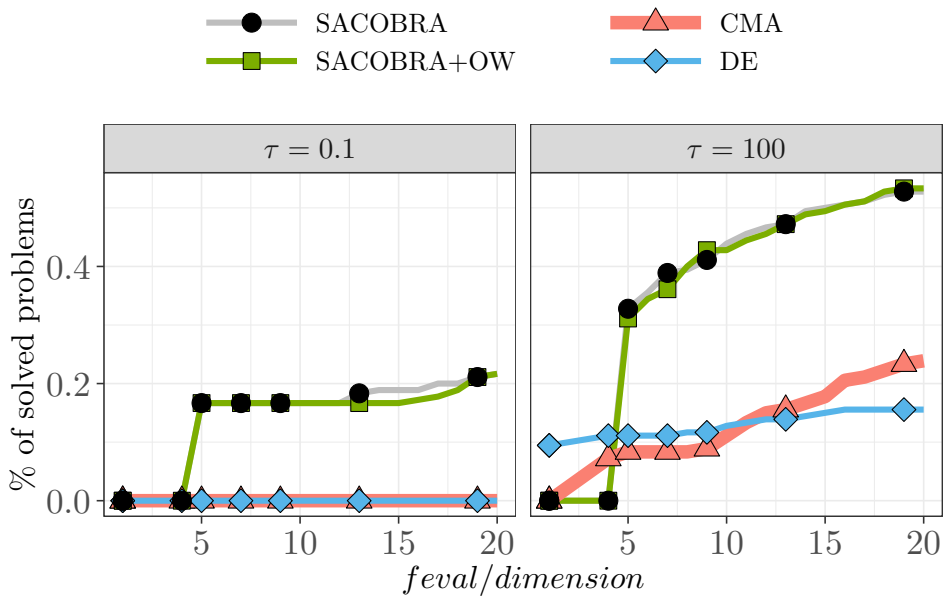

Figure 9.6: Comparing the overall performance of SACOBRA, SACOBRA+OW, DE and CMA-ES algorithms on the 12 studied problems with dimension $d = 10$ and for a very limited number of function evaluations.

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. 9.8 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.

9.6.4 Curse of Dimensionality

Fig. 9.9 compares the overall performances of the studied algorithms for the 5- and 20-dimensional cases. As the dimension grows, SACOBRA and SACOBRA+OW as

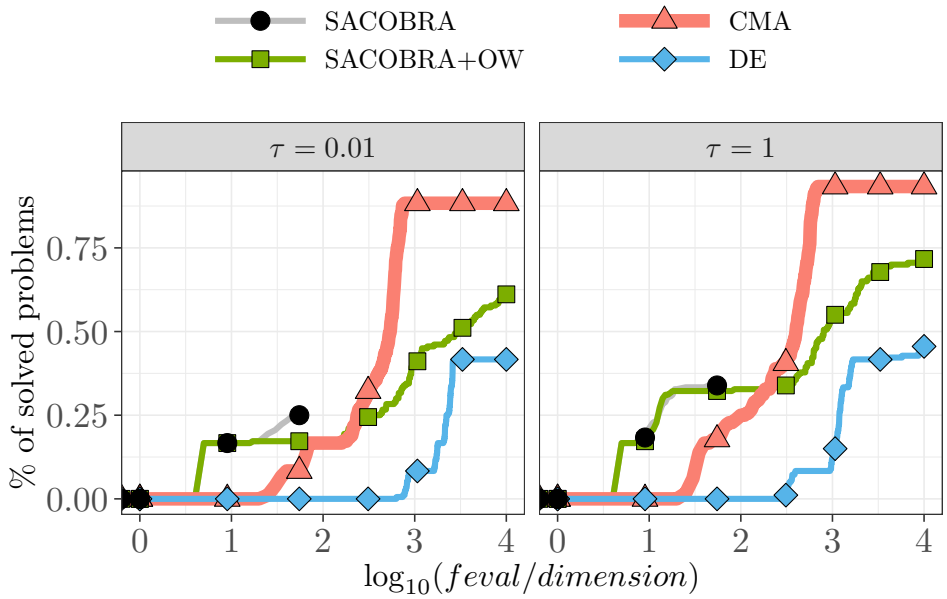

Figure 9.7: Comparing the overall performance of SACOBRA, SACOBRA+OW, DE and CMA-ES algorithms on the 12 studied problems with $d = 10$.

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

9.7 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[3].

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.

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

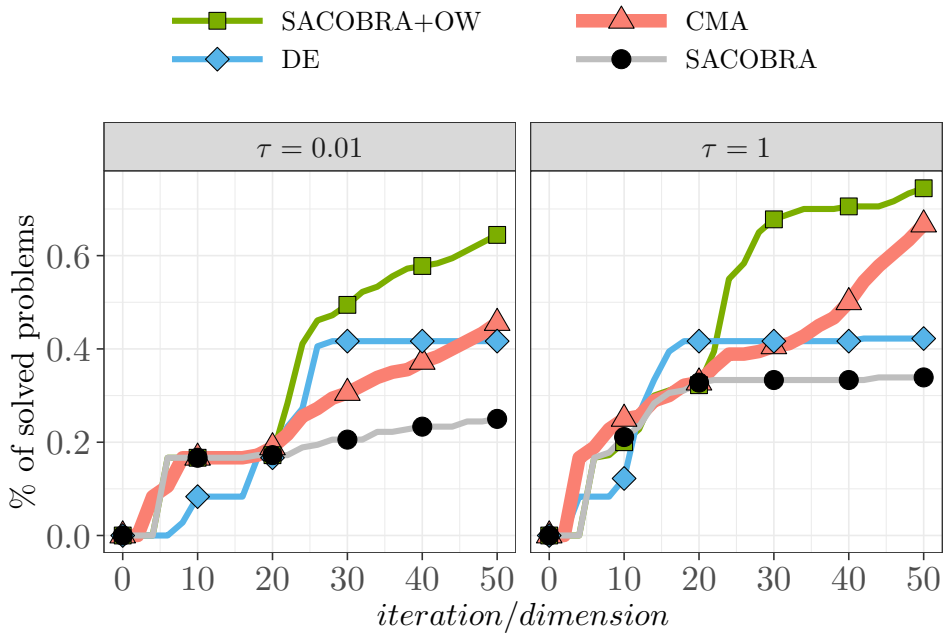

Figure 9.8: Same as Fig. 9.7, but now for the 'optimistic parallelizable' case: We show on the x-axis the number of iterations (or generations), divided by d.

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.

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

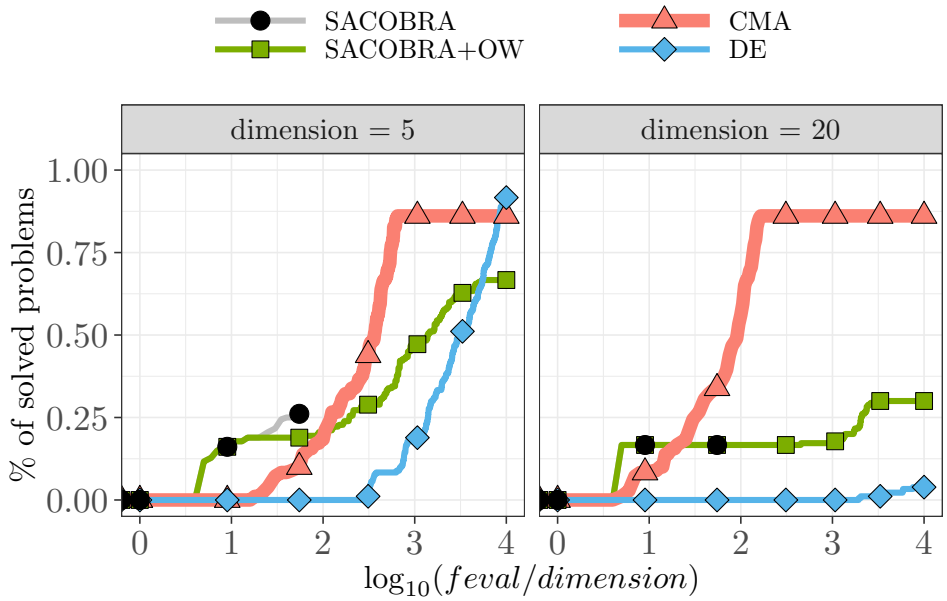

Figure 9.9: Data profiles for all 12 studied problems in the 5-dimensional and 20-dimensional case. The accuracy level is set to $\tau = 0.01$.

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 $d < 20$.

Investigating whether a combination of CMA-ES and surrogate-assisted optimizers could lead to an optimizer which combines 'the best of both worlds' could be a possible future work in this field of research. The efficient Hessian estimation of FOCAL [161, 162] might be an interesting starting point for this.