A. Proof of Claim 5

Proof of Claim 5. First note that $z \in [0,1]$ directly implies $z^{1+\delta} \leq z \leq z^{1-\delta}$ which proves both lower bounds. It remains to prove the upper bounds of δ. We study the mapping $f\colon [0,1] \to \mathbb{R}, z \mapsto z - z^{1+\delta}$. Clearly $f(0) = f(1) = 0 \leq \delta$. Setting the derivative to zero and solving for z we have

$$f'(z) = 1 - (1+\delta)z^{\delta} = 0 \Leftrightarrow z = \left(\frac{1}{1+\delta}\right)^{\frac{1}{\delta}}.$$

Note that $\left(\frac{1}{1+\delta}\right)^{\frac{1}{\delta}} \leq 1$, thus

$$\begin{aligned} f(z) &\leq \left(\frac{1}{1+\delta}\right)^{\frac{1}{\delta}} - \left(\frac{1}{1+\delta}\right)^{\frac{1+\delta}{\delta}} \\ &= \left(\frac{1}{1+\delta}\right)^{\frac{1}{\delta}} \left(1 - \frac{1}{1+\delta}\right) \\ &\leq \frac{1+\delta-1}{1+\delta} = \frac{\delta}{1+\delta} \leq \delta. \end{aligned}$$

Very similar arguments prove the claim for the mapping $f\colon [0,1] \to \mathbb{R}, z \mapsto z^{1-\delta} - z$. Again, $f(0) = f(1) = 0 \leq \delta$. Setting the derivative to zero and solving for z yields

$$f'(z) = (1-\delta)z^{-\delta} - 1 = 0 \Leftrightarrow z = \left(\frac{1}{1-\delta}\right)^{-\frac{1}{\delta}}.$$

Note that $\left(\frac{1}{1-\delta}\right)^{-\frac{1}{\delta}} = (1-\delta)^{\frac{1}{\delta}} \leq e^{-1}$, thus

$$\begin{aligned} f(z) &\leq \left(\frac{1}{1-\delta}\right)^{-\frac{1-\delta}{\delta}} - \left(\frac{1}{1-\delta}\right)^{-\frac{1}{\delta}} \\ &= \left(\frac{1}{1-\delta}\right)^{-\frac{1}{\delta}} \left(\frac{1}{1-\delta} - 1\right) \\ &\leq \frac{1-1+\delta}{1-\delta} e^{-1} \leq 2\delta e^{-1} \leq \delta. \end{aligned}$$

□

B. The Dilation Issue in REMBO

In their seminal work proposing the REMBO variants, Wang et al. (2016b) noted that their random Gaussian embedding A has a dilation that necessitates to scale REMBO's search space to $[-\sqrt{d}, +\sqrt{d}]^d$ to ensure the optimal remains in the search domain. However, REMBO might select a point y such that the random embedding Ay is outside the actual box $\mathcal{X}$. In such a case it projects Ay onto the boundary of $\mathcal{X}$ before evaluating f. This may lead to over-exploration of the boundary regions, which leads to particularly bad performance when optimal point lies in the interior. (Binois et al., 2015) proposed a new warping kernel k_ψ to address this issue by an improved preservation of distances among such points. This method was further improved in (Binois et al., 2019). The analysis in Sect. C below shows that REMBO applies these complex corrections frequently. In contrast, the count-sketch avoids such complex corrections. By construction of the inverse count-sketch S the columns of the matrix are orthogonal, since each row has only a single non-zero entry. Moreover, for any fixed coordinate (dimension) d_i of the low-dimensional space we have that the number of coordinates in the high dimensional space that map to d_i is concentrated at D/d, i.e., this number has mean D/d and a variance that drops (quickly) as D increases. The reason is that h picks the target of each dimension uniformly at random. The dilation of S is thus roughly $\sqrt{D/d}$, and the search domain $\mathcal{Y} = [-1,1]^d$ is mapped to $S\mathcal{Y} = \{Sy \mid y \in \mathcal{Y}\} \subset \mathcal{X} = [-1,1]^D$ with low distortion $(1 \pm \varepsilon)$ as given in Definition 1. Moreover, note that the back-projection is realized by solely copying the coordinates and multiplying them by $\{-1, 1\}$, which assert that no point is projected outside the domain $\mathcal{X}$. Thus, HeSBO avoids complex corrections in contrast to the REMBO variants described above.

C. Analysis of Correction Efforts in REMBO

We examine how often the REMBO variants choose points to sample whose projection to the high-dimensional space is outside of the search domain $\mathcal{X}$. Note that these points are corrected by projecting them back onto the boundary $\partial\mathcal{X}$. Table 1 shows the results on Branin with input $D = 100$ and three different choices for the target dimension d. We see that for larger values of d almost all points chosen by REMBO are projected outside of $\mathcal{X}$. Note that all three algorithms use the same random projection in each iteration, their decisions what points to sample are affected by the respective kernel. In particular, we see that k_X performs better than k_y as intended by Wang et al. (2016b), whereas k_ψ of Binois et al. (2015) performs best among these. Note that the approach proposed in this paper completely avoids the need for corrections.

Table 1. The fraction of steps of the three REMBO variants that project the sample point outside $\mathcal{X}$. The data was collected by running the algorithms on Branin with input dimension $D = 100$ over 30 replications.

	$d = 2$	$d = 3$	$d = 4$
k_y	0.95 ± 0.006	0.993 ± 0.0008	0.9999 ± 0.0001
k_x	0.93 ± 0.009	0.992 ± 0.001	0.9996 ± 0.0002
k_ψ	0.88 ± 0.02	0.992 ± 0.003	0.9997 ± 0.0002

D. Details on Experiments

We provide details needed to replicate the experiments in Table 2. All REMBO variants and HeSBO-EI use identical initial datasets in every replication to initialize the surrogate models, obtained via random Latin hypercube designs. For the other algorithms we used their preferred strategies to obtain the same number of initial data points. We implemented HeSBO-EI and all REMBO variants in Python 2.7 using GPy. They use a GP model with the $\frac{5}{2}$-Matérn kernel and constant mean function. The acquisition function was optimized by L-BFGS-B leveraging its gradient (Frean & Boyle, 2008).

For the other algorithms we used the reference implementations provided by the authors, see the hyperlinks in the references. ADD was provided by Gardner in personal communication. For the neural network parameter search benchmark discussed in Sect. 3, we had to omit ADD because of software incompatibilities with this benchmark and SKL due to its high running time.

Table 2. The fraction of steps of the three REMBO variants that project the sample point outside $\mathcal{X}$. The data was collected by running the algorithms on Branin with input dimension $D = 100$ over 30 replications.

Feature	Choice
number of initial data points	10
sampling approach	Latin hypercube design provided by pyDOE
kernel	Matérn 5/2- ARD
Search space	$[-1, 1]^d$
Method of GP fitting	GPy package (1.9.2)
Number of replications	100
Reported statistic	Median
Error bars	Standard error of median
Processor	Xeon Haswell E5-2695 Dual 2.3 GHz 14-core
Number of CPUs	1

E. Additional Plots for all Conducted Experiments

This section provides the plots for all experiments that we conducted as part of the evaluation. This includes plots already shown in Sect. 3 and additional plots that were omitted due to space constraints. The plots are organized according to the same subsections.

E.1. Performance Evaluation

The performances on Branin as a function of the evaluations are given in Fig. 5.

The performances on Hartmann6 as a function of the evaluations are given in Fig. 6.

The performances on Rosenbrock as a function of the evaluations are given in Fig. 7.

The performances on Styblinski-Tang as a function of the evaluations are given in Fig. 8.

E.2. Robustness

The robustness plots on Branin and Hartmann6 are given in Fig. 9

E.3. Scalability

The performances on Branin, Rosenbrock and Styblinski-Tang as a function of the running time are given in Fig. 10.

E.4. High Dimensional Network Architecture Search

The performances on neural network optimization as a function of the evaluations are given in Fig. 11.

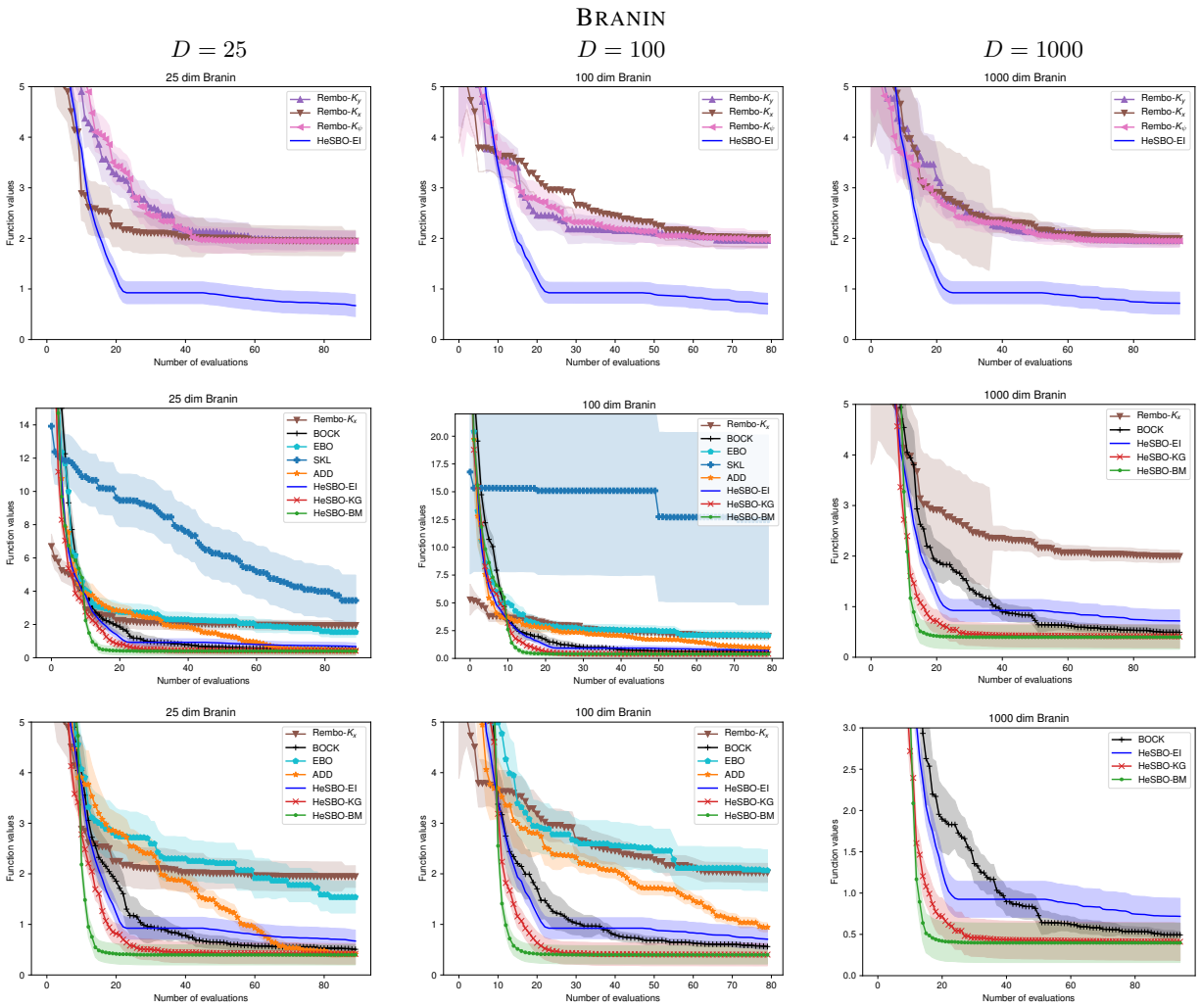

Figure 5. Performance plots for Branin as a function of the evaluations for the projection based algorithms (top), the state-of-the-art algorithms (middle row) and a close-up to the best algorithms (bottom) for different input dimensions $D = 25$ (left), $D = 100$ (middle column), and $D = 1000$ (right).

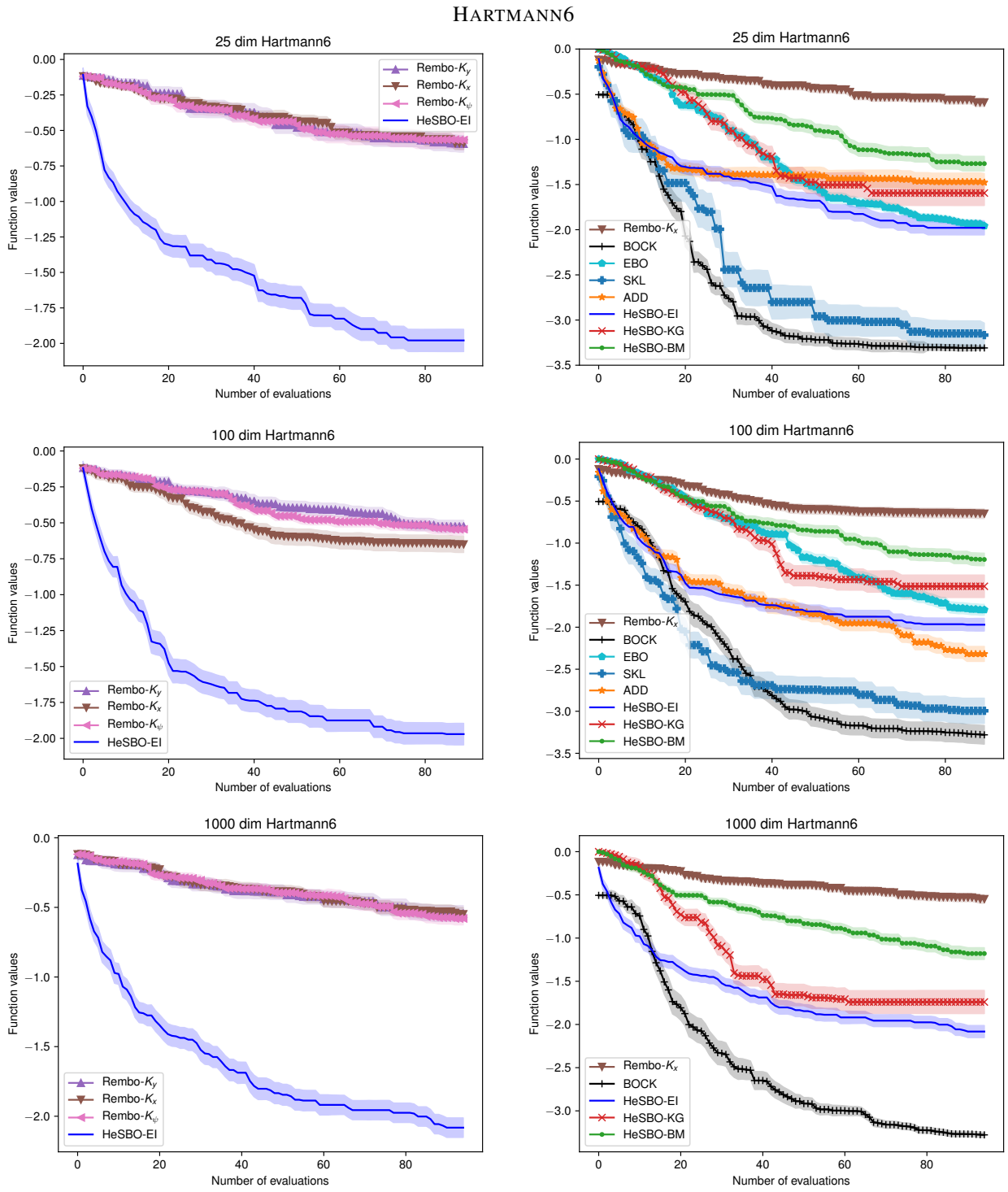

Figure 6. Performance plots for Hartmann-6 as a function of the evaluations for the projection based algorithms (left) and the state-of-the-art algorithms (right) for different input dimensions $D = 25$ (top), $D = 100$ (middle), and $D = 1000$ (bottom).

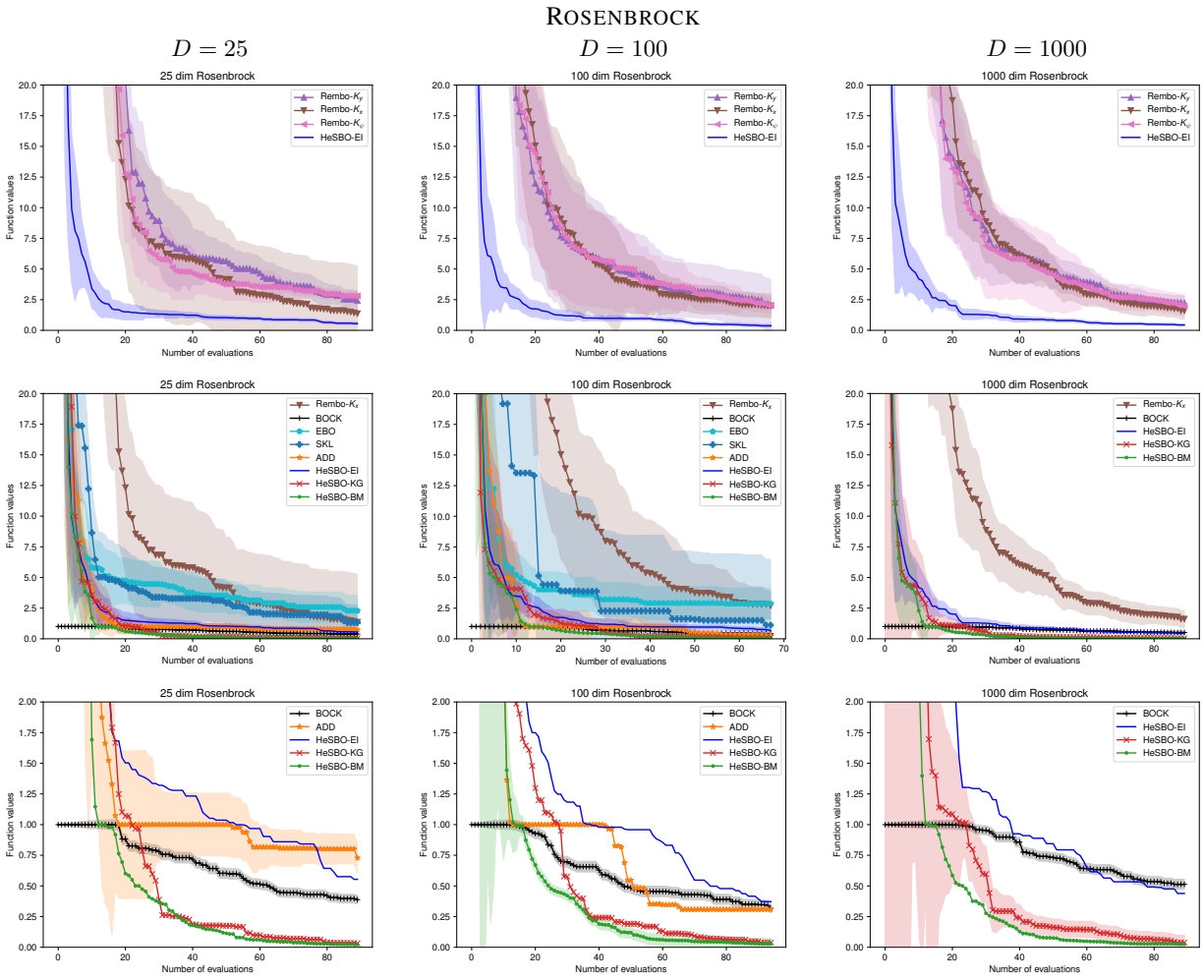

Figure 7. Performance plots for Rosenbrock as a function of the evaluations for the projection based algorithms (top), the state-of-the-art algorithms (middle row) and a close-up to the best algorithms (bottom) for different input dimensions $D = 25$ (left), $D = 100$ (middle column), and $D = 1000$ (right).

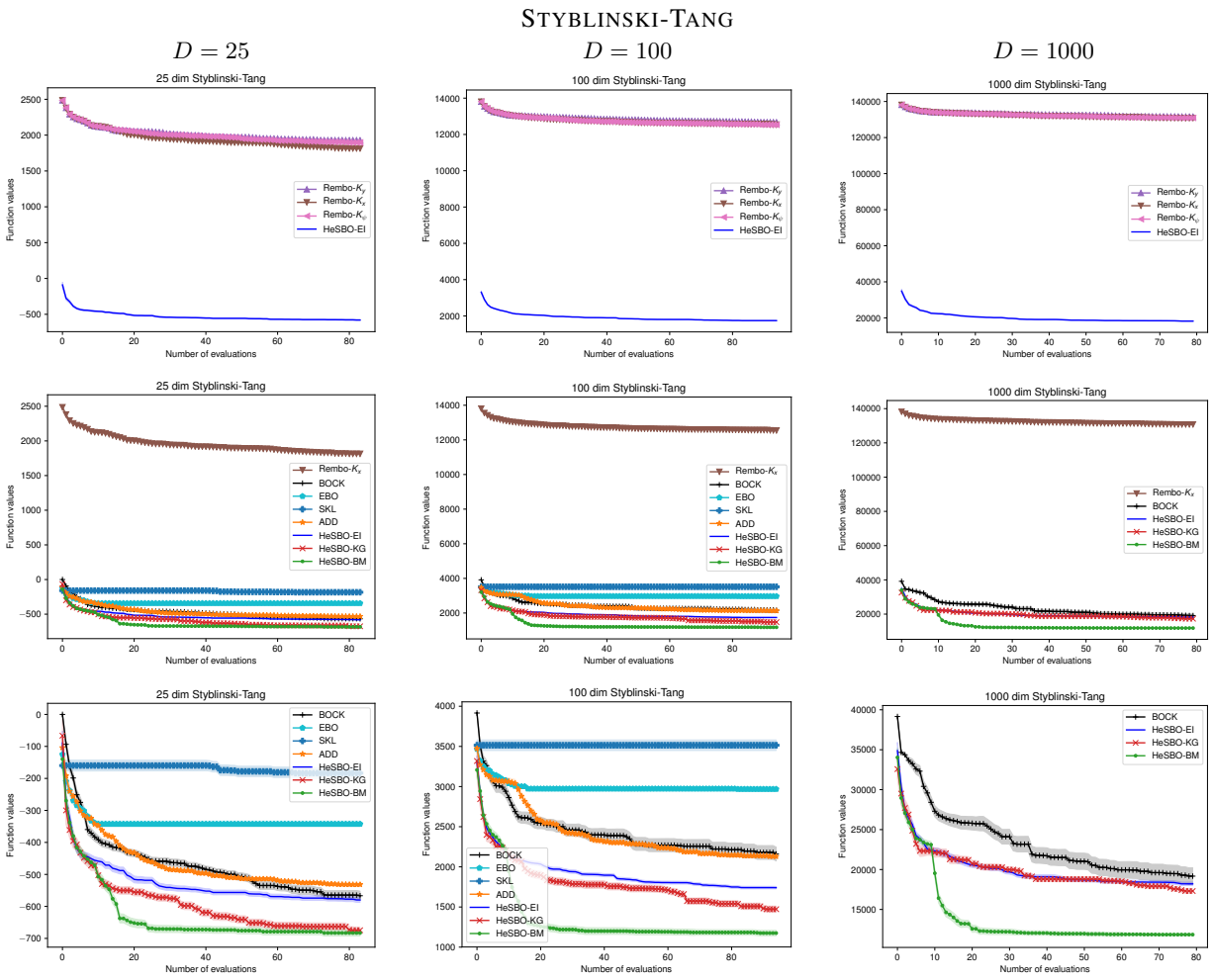

Figure 8. Performance plots for Styblinski-Tang as a function of the evaluations for the projection based algorithms (top), the state-of-the-art algorithms (middle row) and a close-up to the best algorithms (bottom) for different input dimensions $D = 25$ (left), $D = 100$ (middle column), and $D = 1000$ (right).

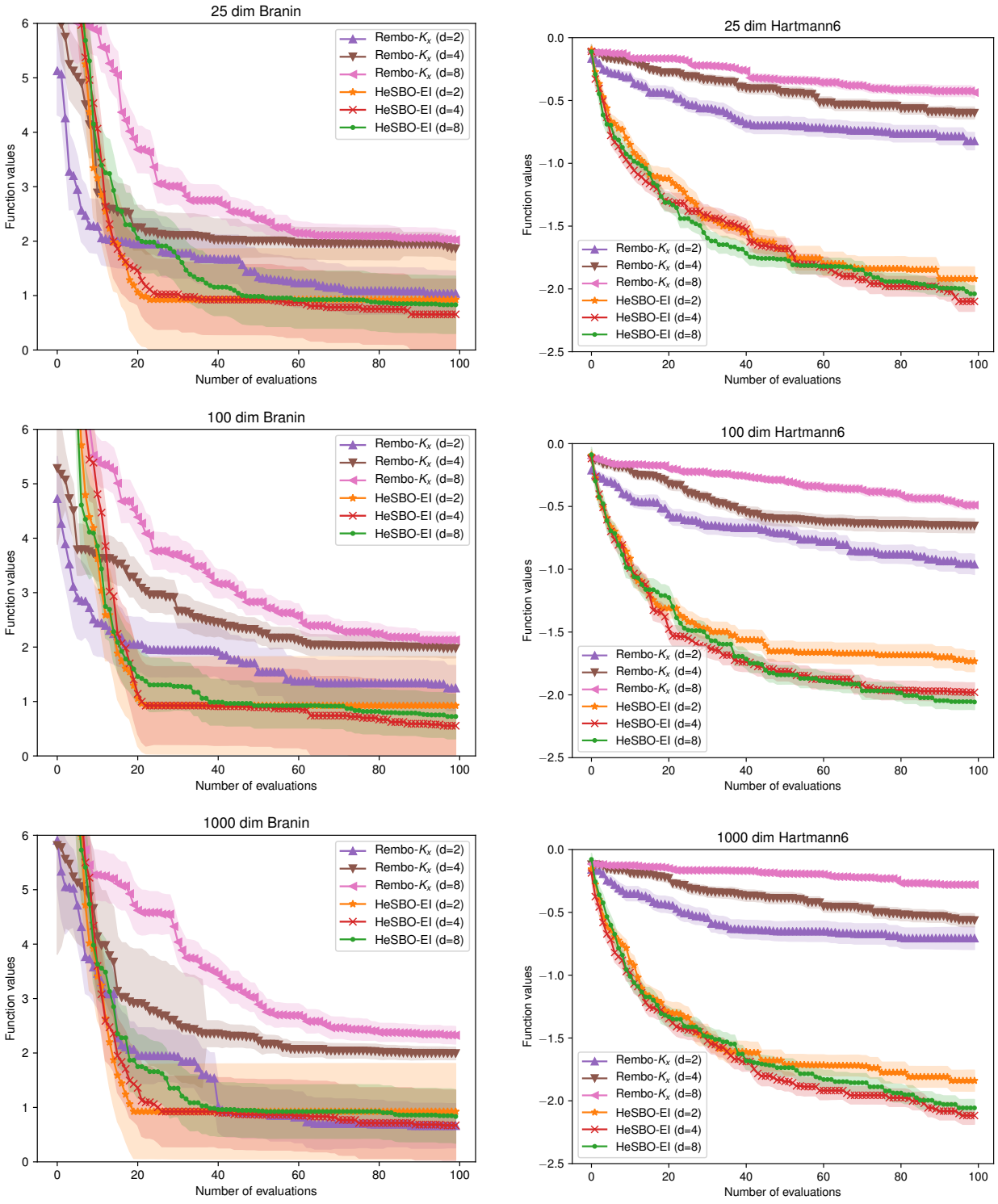

Figure 9. The robustness analysis for Branin (left) and Hartmann6 (right) for $D = 25$ (top), $D = 100$ (middle), $D = 1000$ (bottom) and different values of d.

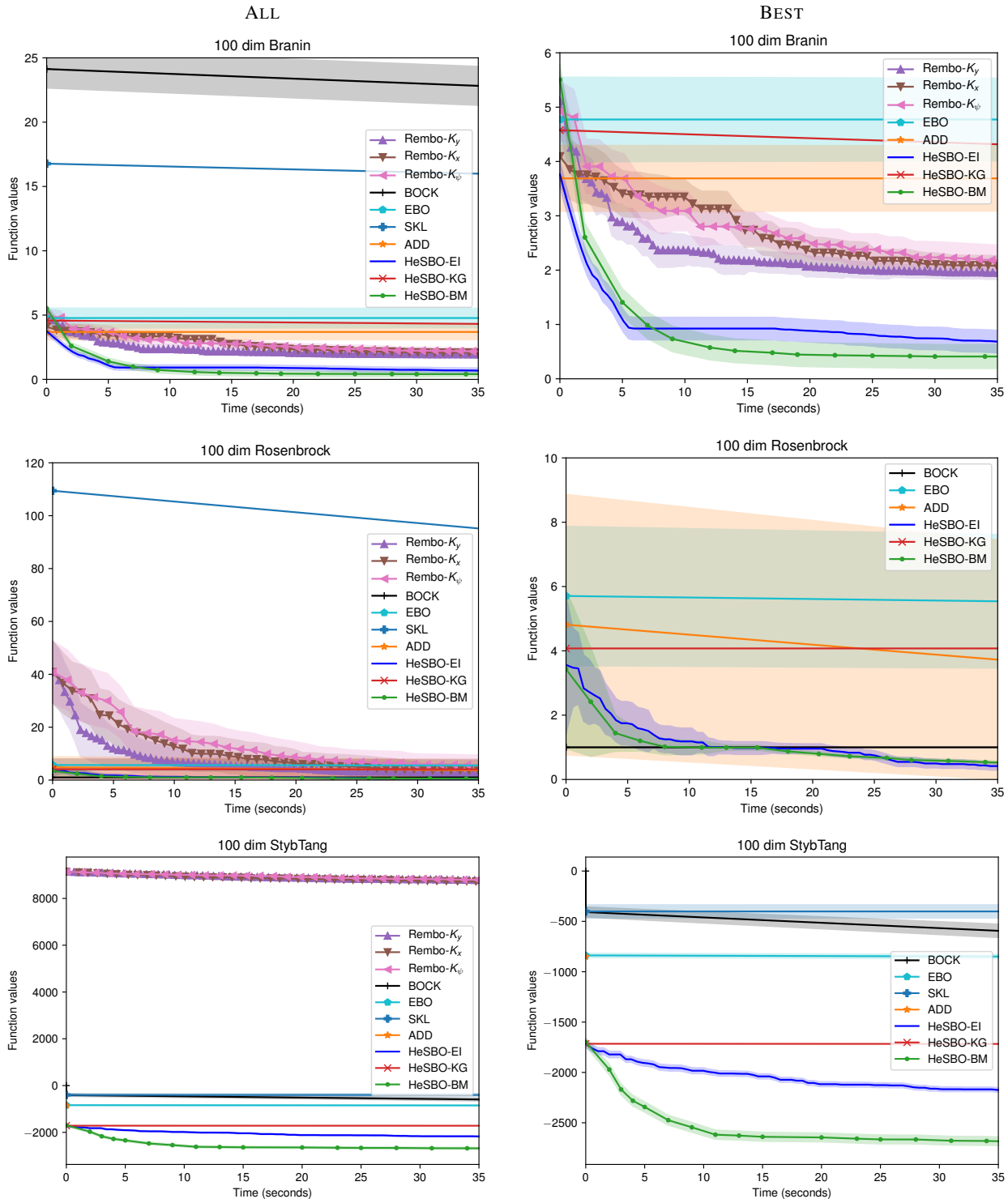

Figure 10. The scalability analysis for all algorithms (left) and a close-up on the best (right) with input dimension $D = 100$ each and different target dimensions for d: Branin $d = 4$ (top), Rosenbrock $d = 4$ (mid) and Styblinski-Tang $d = 12$ (bottom). Note that in the plot on the bottom right, `ADD`'s performance equals `EBO` on the Styblinski-Tang $d = 12$, that is why it is not visible.

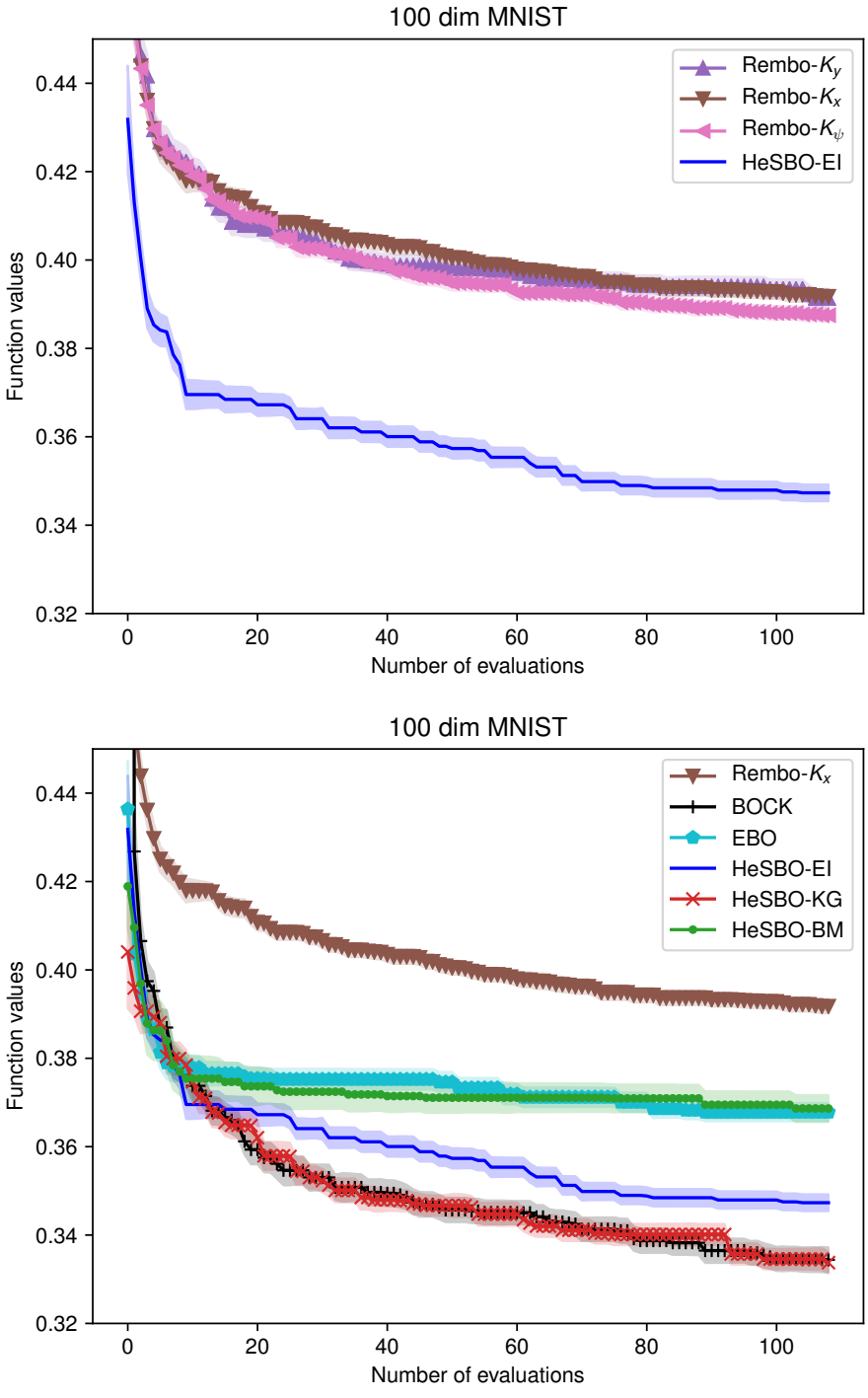

Figure 11. The performance comparison of projection based algorithms (top) and state-of-the-art algorithms (bottom) on the MNIST neural network benchmark with input dimension $D = 100$ and target dimension $d = 12$.