Supplementary Information

Phoenics: A Bayesian Optimizer for Chemistry

Florian Häse,[†] Loïc M. Roch,[†] Christoph Kreisbeck,[†] and Alán Aspuru-Guzik[*,†,‡]

[†]*Department of Chemistry and Chemical Biology, Harvard University, Cambridge, Massachusetts, 02138 USA*

[‡]*Senior Fellow, Canadian Institute of Advanced Research, Toronto, Ontario M5G 1Z8, Canada*

E-mail: alan@aspuru.com

S.1 Analytic objective functions

We benchmarked the performance of the global optimization algorithm Phoenics on a total of 15 analytic objective functions. Nine of these objective functions have a continuous co-domain and are commonly used to benchmark algorithms developed for unconstrained optimization problems. The remaining six benchmark functions have a discrete co-domain and were specifically designed for this study. We provide Python implementations for all 15 benchmark functions on a GitHub repository.[1]

The nine analytic benchmark functions with continuous co-domain were chosen based on their different features: they differ in the number of local minima, number of global minima and the dimensionality of the parameter space for which they are defined. Details are summarized in Tab. S.1. Fig. S.1 displays contour plots of all objective functions for a two dimensional parameter space.

All of the benchmark functions with discrete co-domain project the parameter space

onto integer values in the $[0, 4]$ interval. Contour plots for these benchmark functions are displayed in Fig. S.1 for a two dimensional parameter space, although all of the six functions generalize to higher dimensions as well. Note, that the global minimum of these functions is not a unique point in parameter space, but rather an entire region.

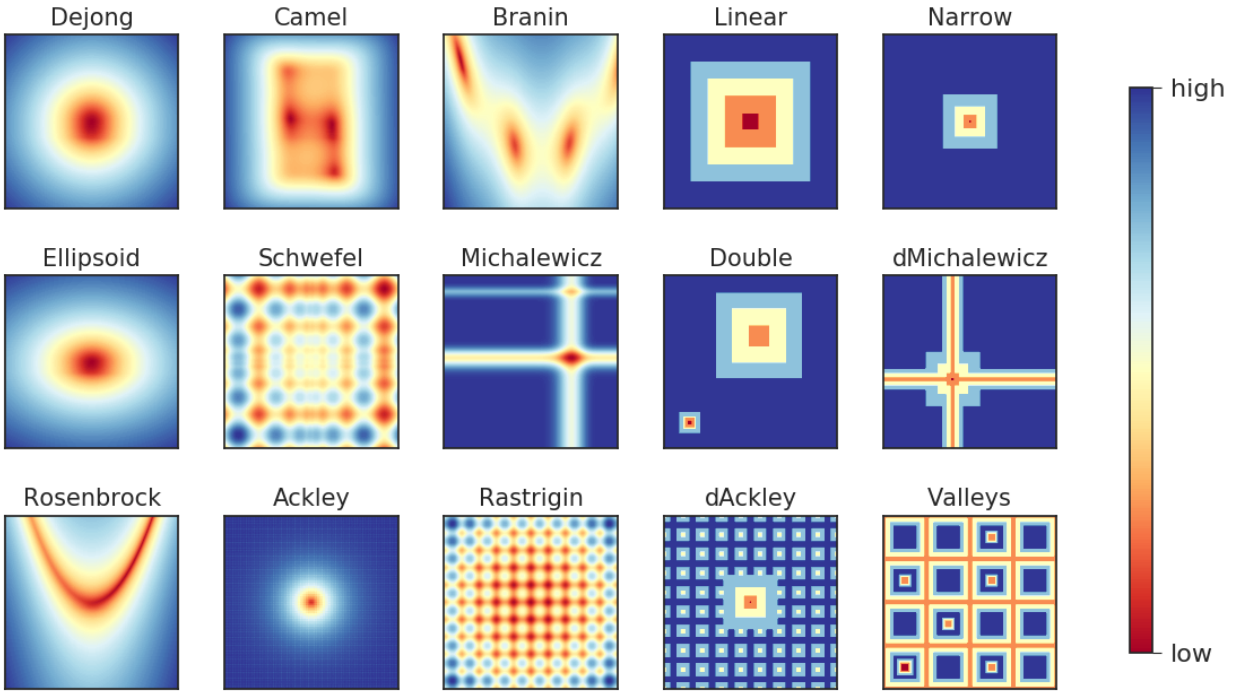

Figure S.1: Contour plot of the two dimensional instances of the objective functions with discrete co-domain used for benchmarking Phoenics, RF and GP optimizers. Python implementations of these objective functions are available on GitHub. [1]

S.2 Random search results

The employed analytic benchmark functions (see Sec. S.1) differ drastically in their shape. While some functions, like the Ackley function or the Schwefel function feature narrow local funnels around their global minima, other functions such as Branin and Dejong have rather broad funnels. The benchmark functions also differ greatly in the range of their co-domain spaces. Instead of reporting the deviation of the current best to the global minimum during an optimization run we therefore compare the optimization algorithms to random searches

Table S.1: Analytic objective functions with continuous co-domain used for benchmarking Phoenics, RF and GP optimizers. Python implementations of these objective functions are available on GitHub.[1]

Objective function	Domain range	Global minimum (2d)	generalizable
Ackley	$x_i \in [-32, 32]$	0.00	yes
Branin	$x_i \in [-5, 15]$	0.397887	no
Camel	$x_i \in [-3, 3]$	-1.0316	no
Dejong	$x_i \in [-5, 5]$	0.00	yes
Ellipsoid	$x_i \in [-5, 5]$	0.00	yes
Michalewicz	$x_i \in [0, 3]$	-1.801	no
Rastrigin	$x_i \in [-5, 5]$	0.00	yes
Rosenbrock	$x_i \in [-2, 2]$	0.00	yes
Schwefel	$x_i \in [-500, 500]$	$-418.9829d$	yes

on the same objective functions.

In one random search run we evaluated each objective function 10^4 times at positions uniformly sampled from the domain space. The lowest achieved function value averaged over 100 random searches with different random seeds serves as the benchmark value for each optimization algorithm. For each algorithm, we record the number of function evaluations needed to discover a point in parameter space which yields a function value lower than the average lowest value encountered in the random searches. The average lowest values for all objective functions in two dimensions are summarized in Tab. S.2.

Table S.2: Lowest achieved function values averaged over 100 independent random searches with 10^4 function evaluations per run. The reported values were used to benchmark Phoenics, RF and GP optimizers.

Loss function	Minimum	Lowest encounter
Ackley	0.00	1.942
Branin	0.397887	0.406
Camel	-1.0316	-1.028
Dejong	0.00	$2.560 \cdot 10^{-3}$
Ellipsoid	0.00	$3.467 \cdot 10^{-3}$
Michalewicz	-1.801	-1.794
Rastrigin	0.00	$4.498 \cdot 10^{-1}$
Rosenbrock	0.00	$4.718 \cdot 10^{-3}$
Schwefel	-837.978	-834.688

Loss function	Minimum	Lowest encounter
Linear funnel	0	0.00
Narrow funnel	0	0.66
Double well	0	0.36
Disc. Ackley	0	0.66
Disc. Michalewicz	0	0.64
Disc. Valleys	0	0.18

S.3 Architecture of the Bayesian neural network

The BNN architecture for all benchmarks in this study consisted in three layers. The dimensionality of the input layer was given by the dimensionality of the parameter space k. We chose to model all hidden layers with 50 units. As described by Eqs. 1 to 3, all layers but the last were connected by hyperbolic tangents; the last layer using a sigmoid activation function

$$\phi_1 = \tanh(x \cdot w_0 + b_0), \tag{1}$$

$$\phi_2 = \tanh(\phi_1 \cdot w_1 + b_1), \tag{2}$$

$$\phi_3 = \text{sigmoid}(\phi_2 \cdot w_2 + b_2), \tag{3}$$

$$\phi_{\text{out}} \sim \mathcal{N}(\phi_3, \tau_n). \tag{4}$$

Priors for weights and biases of the BNN architecture were chosen to be normal distributions with zero mean $\mu_i = 0$ and unit standard deviation $\sigma_i = 1$

$$w_i \sim \mathcal{N}(\mu_i, \sigma_i), \tag{5}$$

$$b_i \sim \mathcal{N}(\mu_i, \sigma_i). \tag{6}$$

The output layer ϕ_3 is used to predict the distributions of means of Gaussian distributions with precisions τ_n, which depend on the number of observations n (see Eq. 4). The precision τ_n of these Gaussian distributions are sampled from a Gamma distribution $\tau_n \sim \Gamma(\alpha, \beta)$ with prior hyperparameters $\alpha = 12n^2$ and $\beta = 1$, where n is the number of observations. With this choice of parameters the precision of the Gaussian distribution increases with the number of observations. Details on the particular choice are provided in Sec. S.4. Note that the parameter space is rescaled to the unit hyper-cube prior to training the model.

The BNN is trained via Bayesian inference (see background). During the training procedure, we update distribution parameters μ_i and σ_i on the Gaussian distributions for weights

and biases as well as parameters α and β on the Gamma distribution from which the precision τ_n is drawn. We collectively refer to all of these model parameters as $\boldsymbol{\theta}$.

S.4 Precision

In this work we propose to approximate a given objective function with a prior constructed from Gaussian distributions. Given a set of n observations $\mathcal{D}_n$ of pairs of parameter points and corresponding objective function values, the approximation to the objective function is constructed from n Gaussians. The locations of the Gaussian distributions are drawn from a BNN trained to predict the locations of observed parameter points. Precisions τ of these Gaussians, however, are drawn from a Gamma distribution parametrized via a probability density function as presented in Eq. 7, where Γ denotes the Gamma function.

$$f(x;\alpha,\beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)} \qquad \text{for } x > 0 \text{ and } \alpha, \beta > 0 \tag{7}$$

The prior for this Gamma distribution is chosen to be $\alpha = 12n^2$ and $\beta = 1$ for a given set $\mathcal{D}_n$ of n observations. With this choice of hyperparameters the expectation value for the prediction is $\langle\tau\rangle = 12n^2$. The prefactor 12 in the expectation value of the precision ensures that in the case of only a single observation the standard deviation of the approximating Gaussian distribution matches the standard deviation of a uniform distribution along a single dimension of the unit hyper-cube. Starting out from an uninformative uniform prior we therefore gradually increase our believe about the parameter space with this particular choice of the prefactor. The dependence of the hyperparameter α on the number of observations guarantees that the precision of the Gaussian distributions increases with more observations, i.e. the standard deviation of the Gaussian decreases. This decrease in the standard deviation with the number of observations is necessary for the convergence of the approximative model to the objective function in the limit of an infinite number of observations.

Several different protocols could be applied to increase the precision of the Gaussian

distributions with the number of observations. In a scenario in which a dataset consisting of n observations is approximated by a single Gaussian distribution, the precision of the Gaussian increases as the square root of the number of observed points, i.e. $\tau \propto \sqrt{n}$. Likewise we can consider a case in which we sample a random variable from the sum of Gaussian distributions. Assuming each of the Gaussian distributions has the same standard deviation σ_0, the standard deviation of the random variable is given by $\sqrt{n}\sigma_0$.

Based on these two considerations we studied the performance of Phoenics with three different protocols for increasing the precision of the Gaussians with the number of observations: increasing the expected value of the precision with the number of observations as (i) $\langle\tau\rangle \propto n$, (ii) $\langle\tau\rangle \propto n^2$ and (iii) $\langle\tau\rangle \propto n^3$ given on the prior considerations. Results of the benchmark runs with Phoenics and the three different schedules on selected objective functions are reported in Tab. S.3. We report the minimum number of required objective function evaluations to reach values lower than the average lowest value encountered after 10^4 random evaluations of the objective function (see Sec. S.2).

Table S.3: Minimum number of required objective function evaluations to reach values lower than the average lowest value encountered after 10^4 random evaluations of the objective function. For each simulation, the precision of the Gaussian distributions was increased with a different schedule based on the number of observations n. Lowest number of required evaluations for each objective function are printed in bold.

Protocol	Ackley	Dejong	Schwefel	dAckley
$\langle\tau\rangle \propto n$	39 ±4	29 ±2	137 ±10	**176** ±8
$\langle\tau\rangle \propto n^2$	**19** ±1	**21** ±2	**108** ±10	179 ±4
$\langle\tau\rangle \propto n^3$	43 ±1	43 ±1	126 ± 6	186 ±4

We find, that the $\langle\tau\rangle \propto n^2$ schedule for shrinking approximating Gaussian distributions performs the best out of the proposed shrinking schedules across all objective functions. Slower increases in the precision of the Gaussian distributions like in the $\langle\tau\rangle \propto n$ schedule seem to create persistent regions in parameter space for which the acquisition function predicts unfavorable objective function values and does not sufficiently enhance exploration. In contrast, the $\langle\tau\rangle \propto n^3$ schedule shrinks Gaussian distributions too quickly such that

acquired knowledge cannot be exploited sufficiently and the algorithm behaves more like random search. Based on these findings, Phoenics adopts the $\langle\tau\rangle \propto n^2$ as shrinking schedule.

S.5 Convergence of the approximation to the objective function

We suggest to approximate an objective function f via a kernel average α of observed function values f_k, where kernels p_k depend in the associated parameter points $\boldsymbol{x}_k$ (see main text for details).

$$\alpha(\boldsymbol{x}) = \frac{\sum_{k=1}^{n} f_k p_k(\boldsymbol{x})}{\sum_{k=1}^{n} p_k(\boldsymbol{x})} \tag{8}$$

More specifically, the kernels p_k are estimated from a BNN. Here, we aim to argue that this approximation of the objective function converges to the objective function in the limit of infinitely many distinct observations. We start our discussion with the construction of the kernel densities p_k. Kernel densities are sampled from the BNN according to Eq. 9

$$p_k(\boldsymbol{x}) = \left\langle \sqrt{\frac{\tau_n}{2\pi}} \exp\left[-\frac{\tau_n}{2}(\boldsymbol{x} - \phi_3(\boldsymbol{\theta}; \boldsymbol{x}_k))^2\right] \right\rangle_{\text{BNN}}. \tag{9}$$

Note, that $\phi_3(\boldsymbol{\theta}; \boldsymbol{x}_k)$ are constructed according to Eqs. 10 to 12, where $\boldsymbol{x}_k$ denotes the parameter point associated to the objective function value f_k.

$$\phi_1 = \tanh(x_k \cdot w_0 + b_0), \tag{10}$$

$$\phi_2 = \tanh(\phi_1 \cdot w_1 + b_1), \tag{11}$$

$$\phi_3 = \text{sigmoid}(\phi_2 \cdot w_2 + b_2), \tag{12}$$

Note, that weights w_i and biases b_i can be chosen, such that $\phi_3 = \boldsymbol{x}_k$. Since $\boldsymbol{x}_k$ can be assumed to be within the d-dimensional unit hyper-cube $\boldsymbol{x}_k \in [0,1]^d$ we can find

$$w_0 = \frac{\arctan(x_k)}{x_k}, \qquad w_1 = \frac{\arctan(x_k)}{x_k}, \qquad w_2 = \frac{1}{x_k}\log\left(\frac{x_k}{1-x_k}\right), \qquad b_0 = b_1 = b_2 = 0, \tag{13}$$

which gives the desired result $\phi_3 = \boldsymbol{x}_k$. In the scenario of a perfectly trained BNN, the kernel densities therefore reduce to

$$p_k(\boldsymbol{x}) = \sqrt{\frac{\tau_n}{2\pi}}\exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}-\boldsymbol{x}_k)^2\right]. \tag{14}$$

The precisions τ_n of these kernels are sampled from a Gamma distribution (see main text), whose expectation value linearly increases with the number of observations n. We now consider the value of the approximation α at the position of an observed parameter point $\boldsymbol{x}_l$ in the limit of $n \to \infty$

$$\lim_{n\to\infty}\alpha(\boldsymbol{x}_l) = \lim_{n\to\infty}\frac{\sum_{k=1}^{n} f_k p_k(\boldsymbol{x}_l)}{\sum_{k=1}^{n} p_k(\boldsymbol{x}_l)}, \tag{15}$$

$$= \lim_{n\to\infty}\frac{\sum_{k=1}^{n} f_k \sqrt{\frac{\tau_n}{2\pi}}\exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}_l-\boldsymbol{x}_k)^2\right]}{\sum_{k=1}^{n} \sqrt{\frac{\tau_n}{2\pi}}\exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}_l-\boldsymbol{x}_k)^2\right]}, \tag{16}$$

$$= \lim_{n\to\infty}\frac{f_l + \sum_{k=1,k\neq l}^{n} f_k \exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}_l-\boldsymbol{x}_k)^2\right]}{1 + \sum_{k=1,k\neq l}^{n} \exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}_l-\boldsymbol{x}_k)^2\right]} = f_l, \tag{17}$$

where we used the fact that the series in the numerator and the denominator are absolute convergent and $||\boldsymbol{x}_l - \boldsymbol{x}_k|| < 1$ for any $\boldsymbol{x_l}$ and $\boldsymbol{x}_k$ on the considered domain. When evaluating the approximation α at an arbitrary point $\boldsymbol{x}$ in the domain, however, we find that the value assumed by the approximation is governed by the observed function value associated with

the closest observed parameter point $\boldsymbol{x}_k$. We carry out the same limit as before

$$\lim_{n\to\infty} \alpha(\boldsymbol{x}) = \lim_{n\to\infty} \frac{\sum_{k=1}^{n} f_k p_k(\boldsymbol{x}_l)}{\sum_{k=1}^{n} p_k(\boldsymbol{x})}, \tag{18}$$

$$= \lim_{n\to\infty} \frac{\sum_{k=1}^{n} f_k \sqrt{\frac{\tau_n}{2\pi}} \exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}-\boldsymbol{x}_k)^2\right]}{\sum_{k=1}^{n} \sqrt{\frac{\tau_n}{2\pi}} \exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}-\boldsymbol{x}_k)^2\right]}. \tag{19}$$

This expression can now be simplified by introducing β_k as

$$\beta_k(\boldsymbol{x}) = \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_k)^2\right). \tag{20}$$

We find

$$\lim_{n\to\infty} \alpha(\boldsymbol{x}) = \lim_{n\to\infty} \frac{\sum_{k=1}^{n} f_k \beta_k^{\tau_n}}{\sum_{k=1}^{n} \beta_k^{\tau_n}}. \tag{21}$$

This expression can be simplified further by dividing both numerator and denominator by the largest β_k, which we denote with $\beta_m = \max_{k=1,\ldots,n}(\beta_k)$. Note, that the largest β_k is obtained for the point $\boldsymbol{x}_m$ in the parameter space, which is the closest to the considered point $\boldsymbol{x}$.

$$\lim_{n\to\infty} \alpha(\boldsymbol{x}) = \lim_{n\to\infty} \frac{\sum_{k=1}^{n} f_k (\beta_k/\beta_m)^{\tau_n}}{\sum_{k=1}^{n} (\beta_k/\beta_m)^{\tau_n}}, \tag{22}$$

$$= \lim_{n\to\infty} \frac{f_m + \sum_{k=1,k\neq m}^{n} f_k (\beta_k/\beta_m)^{\tau_n}}{1 + \sum_{k=1,k\neq m}^{n} (\beta_k/\beta_m)^{\tau_n}} = f_m, \tag{23}$$

as all $\beta_k/\beta_m < 1$ for $k \neq m$. The approximation is therefore dominated by the objective function value f_m observed for the closest point $\boldsymbol{x}_m$ in the parameter space. The same

convergence behavior can be observed for the extended expectation calculation as shown below

$$\lim_{n\to\infty} \alpha(\boldsymbol{x}_l) = \lim_{n\to\infty} \frac{\sum_{k=1}^{n} f_k p_k(\boldsymbol{x}_l) + \lambda p_{\text{uniform}}(\boldsymbol{x}_l)}{\sum_{k=1}^{n} p_k(\boldsymbol{x}_l) + p_{\text{uniform}}(\boldsymbol{x}_l)}, \tag{24}$$

$$= \lim_{n\to\infty} \frac{\sum_{k=1}^{n} f_k \sqrt{\frac{\tau_n}{2\pi}} \exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}_l - \boldsymbol{x}_k)^2\right] + \lambda p_{\text{uniform}}(\boldsymbol{x}_l)}{\sum_{k=1}^{n} \sqrt{\frac{\tau_n}{2\pi}} \exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}_l - \boldsymbol{x}_k)^2\right] + p_{\text{uniform}}(\boldsymbol{x}_l)}, \tag{25}$$

$$= \lim_{n\to\infty} \frac{f_l + \sum_{k=1,k\neq l}^{n} f_k \exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}_l - \boldsymbol{x}_k)^2\right] + \sqrt{\frac{2\pi}{\tau_n}} \lambda p_{\text{uniform}}(\boldsymbol{x}_l)}{1 + \sum_{k=1,k\neq l}^{n} \exp\left[-\frac{\tau_n}{2}(\boldsymbol{x}_l - \boldsymbol{x}_k)^2\right] + \sqrt{\frac{2\pi}{\tau_n}} p_{\text{uniform}}(\boldsymbol{x}_l)} = f_l, \tag{26}$$

where we used the fact that $p_{\text{uniform}}(\boldsymbol{x}_l)$ is constant and finite. Similar to the previous scenario it can be shown that also the extended approximation approaches the objective function value associated with the closest observed parameter point when evaluated at arbitrary parameter points $\boldsymbol{x}$.

We illustrate this convergence behavior in Fig. S.2, where we display the approximations to an objective function constructed from different numbers of observation. We find that the approximation matches the true objective in cases where there are more observations available.

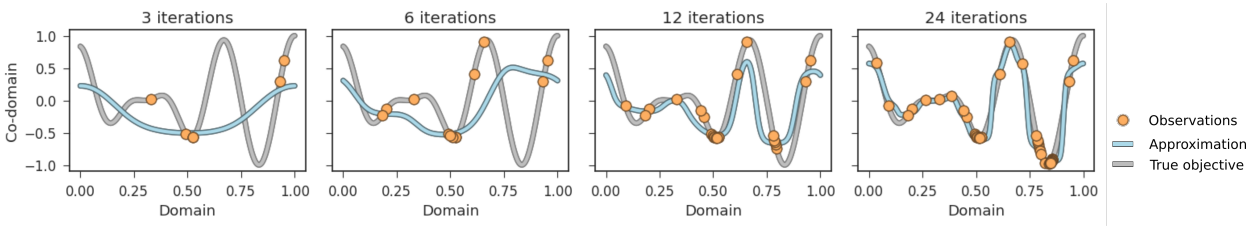

Figure S.2: Illustration of the convergence behavior of the constructed approximation α (blue) to the objective function f (grey) for different numbers of observations (orange).

S.6 Surrogate optimization

New points in parameter space for querying the objective function are proposed based on the global minimum of the acquisition function (see Eq. 3, and Fig. 1). The problem of finding the global minimum of the objective function, which is costly to evaluate, is therefore reduced to searching the global minimum of the surrogate function.

As a compromise of accuracy and computational cost, we search for the global minimum of the approximating function by uniformly sampling points in the parameter space and then running a gradient based optimizer on half of the proposed samples on the objective function approximation. Unless otherwise noted, all reported results were obtained from proposing 2000 uniform samples for each dimension of the parameter space and locally optimizing half of the sampled points with the L-BFGS algorithm for at most 20 optimization steps.

S.7 Benchmark results

We benchmarked Phoenics, RF and GP optimizers on a total of 15 different benchmark functions, which are reported and discussed in detail in Sec. S.1. For each of the benchmark function we ran the optimization algorithm for a total of 200 iterations in 20 independent runs initialized with different random seeds and recorded the lowest discovered objective function values for each iteration. Each objective function was optimized by three independent optimizer instances constructed from different exploration parameter values $\lambda \in \{-1, 0, 1\}$.

Average lowest discovered objective function values and bootstrapped uncertainties are depicted in Fig. S.3. For comparison we provide the average lowest objective function values discovered by Bayesian optimization with Gaussian processes as implemented in spearmint and with random forests as implemented in SMAC.

We find that instances of the optimization algorithm introduced in this study perform better than GP optimization and RF optimization in eleven out of 15 cases. For all studied objective function, instances of the introduced optimization algorithm perform better than

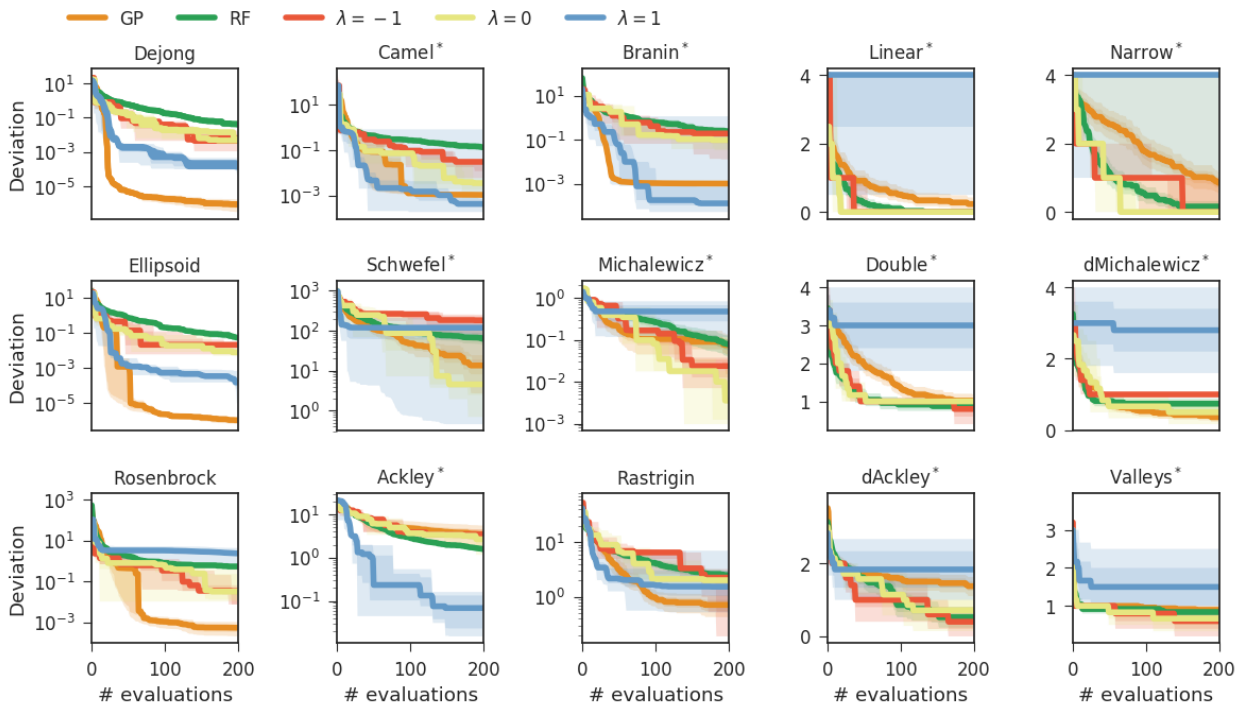

Figure S.3: Lowest discovered objective function values averaged over 20 independent optimizations for the optimization algorithm introduced in this study, which we constructed with three different choices for the exploration parameter $\lambda \in \{-1, 0, 1\}$. We also report average lowest objective function values discovered by Bayesian optimization with Gaussian processes (spearmint package) and random forests (SMAC package). Objective functions for which the optimizer introduced in this study discovered the lowest objective function values are indicated by an asterisk. Uncertainty bands illustrate bootstrapped estimates of the deviation of the means with one and two standard deviations.

RF optimization. GP optimization, however, discovers lower objective function values than instances of Phoenics in four cases.

These four cases, for which GP optimization discovers lower objective function values, are cases of purely convex objective functions (Dejong, Ellipsoid and Rosenbrock) and the Rastrigin function, which is mostly convex with small local minima modulating a general hyperparabolic shape.

We further observe that for some objective functions a more explorative (more negative) value of the exploration parameter improves the performance while in other cases a more exploitative (more positive) value is beneficial. For instance in the case of objective functions with a discrete co-domain (right and second to right columns in Fig. S.3) we observe good

performance with more negative exploration parameter values. In fact, positive values for the exploration parameter result in the algorithm no longer finding the global minimum. This, however, can be explained by the fact that parameter points proposed with favoring exploitation will always be in close proximity to the current best observed parameter point. On a discrete co-domain, however, all points in close proximity to the current best yield the same objective function value.

In other cases, especially for mostly convex objective functions such as the Dejong function or the Camel function, Phoenics performs the best when favoring exploitation over exploration. We therefore conclude that there cannot be a single best balance between exploration and exploitation for objective functions as diverse as the objective functions in this benchmark set.

S.8 Batch optimization results

Instead of marginalizing over the exploration parameter λ in the acquisition function of Phoenics , we suggest to propose parameter points in batches, some of which are determined favoring exploration and others favoring exploitation. Here, we demonstrate the the proposed algorithm benefits from such a procedure even if the proposed parameter points are evaluated sequentially.

Tab. S.4 summarizes the results for a simple batch optimization experiment on a selected set of objective functions. Phoenics was run with sampling parameter values evenly spread out over the $[-1, 1]$ interval. Note, that the evaluation of a batch with p samples was counted as p evaluations of the objective functions. Values for λ were fixed during the entire optimization procedure.

Fig. S.4 displays the lowest achieved deviations between sampled objective function values and their global minima for a selected set of objective functions. Results reported for each objective function are averaged over 20 independent runs. Phoenics was run by proposing

Table S.4: Average lowest achieved errors of the three global optimization algorithms compared in this study. Averages were taken over 20 runs with different random seeds. For each optimizer we report the lowest achieved errors for runs in which a different number of points p were proposed in each training iteration. Lowest numbers of evaluations required by each optimization algorithm are indicated in bold.

Method	p	Ackley	Dejong	Schwefel	dAckley
Phoenics	1	39 ± 5	55 ± 4	66 ± 4	85 ± 8
	2	**26** ± 3	44 ± 2	58 ± 6	72 ± 6
	4	43 ± 1	**36** ± 1	48 ± 8	**22** ± 2
	8	69 ± 2	61 ± 2	**47** ± 2	40 ± 3
RF	1	**185** ± 2	**250** ± 9	**245** ± 3	**143** ± 6
	2	191 ± 5	349 ± 11	277 ± 6	166 ± 5
	4	212 ± 6	713 ± 32	394 ± 10	178 ± 5
	8	246 ± 8	906 ± 47	446 ± 17	225 ± 28
GP	1	**43** ± 3	13 ± 1	**93** ± 2	**100** ± 12
	2	45 ± 3	**12** ± 1	94 ± 3	107 ± 19
	4	46 ± 2	12 ± 1	96 ± 2	126 ± 6
	8	57 ± 3	16 ± 1	95 ± 2	163 ± 20

parameter points in batches of p points, which where then evaluated sequentially. The evaluation of p points in one batch was counted as p objective function evaluations. We report the number of objective function evaluations needed to reach the indicated objective function values. Values for the exploration parameter were drawn evenly spaced from the $[-1, 1]$ interval. We provide the lowest achieved objective functions values in GP optimization and RF optimization for comparison.

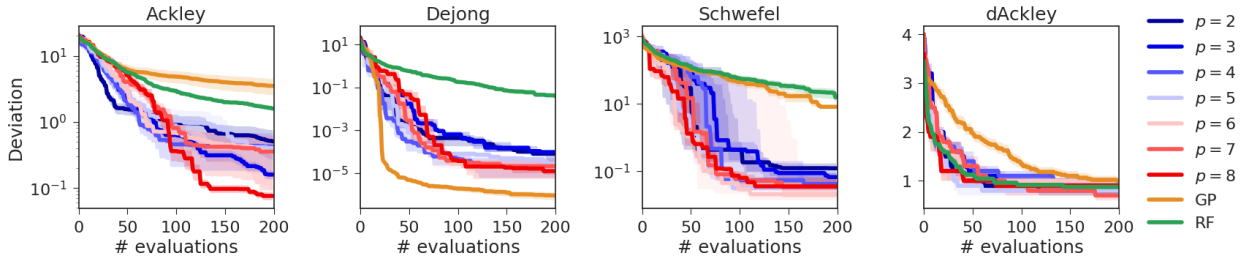

Figure S.4: Lowest achieved average deviation between sampled objective function values and their global minima for a selected set of objective functions. Phoenics proposed a total of p points per batch, which was counted as p function evaluations. Uncertainty bands illustrate bootstrapped estimates of the deviation of the means with one and two standard deviations.

We observe that the synergistic effect of proposing parameter points in batches reported on the Ackley function in the main text (see Fig. 3) also occurs for the other studied objective functions. In fact, even for objective functions with a discrete co-domain (right panel, Fig. S.4) batch optimization seems to enhance the performance of Phoenics, although we demonstrated that Phoenics does not perform well on objective functions with a discrete co-domain when proposing parameter points with a bias towards exploitation.

S.9 Algorithm comparisons

We demonstrated that Phoenics generally performs better when proposing parameter points in batches, where some of the parameter points are proposed with a bias towards exploration and others are proposed with a bias towards exploitation (see Sec.). Here, we report on the performance of Phoenics with batch exploration and compare to RF and GP optimization. Fig. S.5 depicts the traces of average deviations between the lowest achieved objective function values and their global minima for a total of 20 independent runs with all three optimization algorithms. Phoenics proposed 4 points per batch based on exploration parameter values evenly spread across the $[-1, 1]$ interval.

We find that Phoenics finds parameter points yielding objective function values closer to their global optima than RF and GP optimization in 12 out of 15 cases (indicated with $*$ in Fig. S.5). Phoenics is only outperformed by GP optimization if the objective function is convex, i.e. for the Dejong, Ellipsoid and Rosenbrock function. Nevertheless, Phoenics finds reasonable parameter points yielding low objective function values even for these functions.

In addition, we observe a synergistic effect of batch optimization for 12 out of 15 objective functions (indicated with † in Fig. S.5). For these objective functions, batch optimization performs better than any of the exploration parameter choices reported in Fig. S.3). We note, that the same batching protocol was used for all objective functions, indicating the flexibility and broad applicability of this protocol.

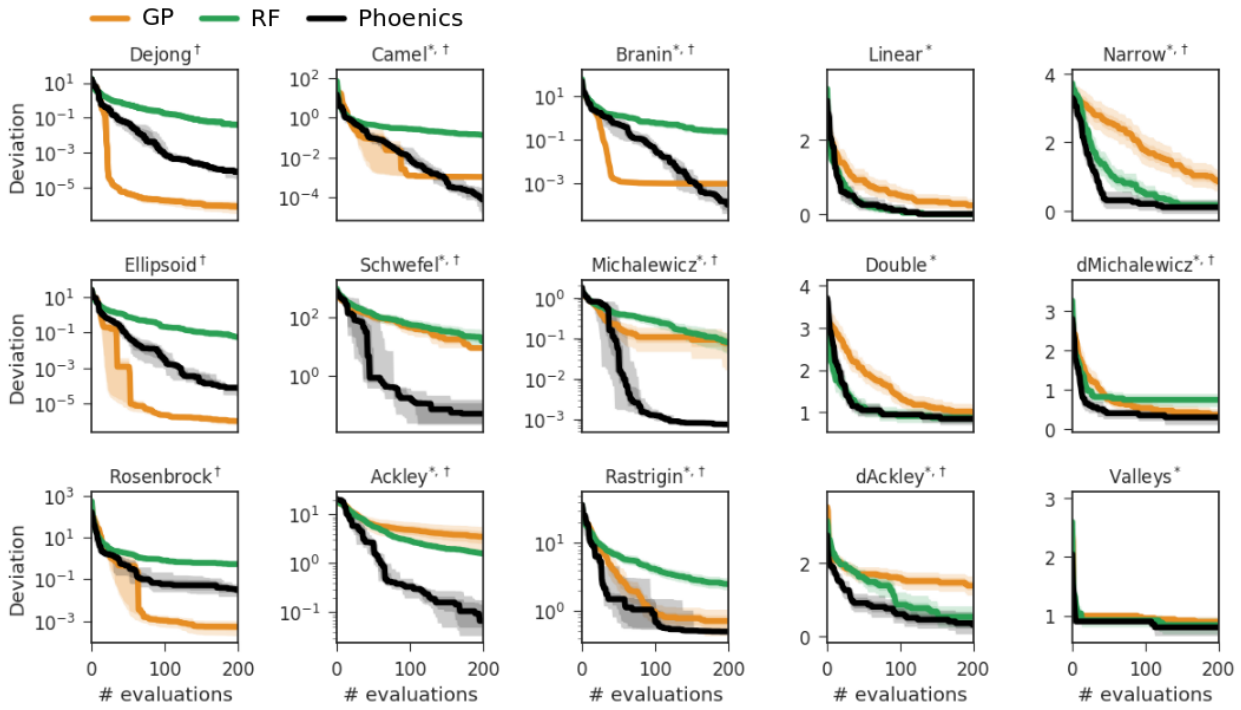

Figure S.5: Deviations between lowest achieved objective function values and their global minima for a total of 20 independent runs with the three studied optimization algorithms. Phoenics was run in a batch exploration proposing 4 points per batch based on exploration parameter values evenly spread across the $[-1, 1]$ interval. Objective functions on which Phoenics performs better than GP and RF optimization are indicated denoted with $*$, function for which a synergistic effect was observed, i.e. improved performance with batch optimization compared to sequential optimization, are denoted with †. Uncertainty bands illustrate bootstrapped estimates of the deviation of the means with one and two standard deviations.

Fig. S.6 reports the performance of particle swarm optimization (PSO) and the covariance matrix adaptation evolution strategy (CMA-ES) on all objective functions. We increased the number of allowed function evaluations for these two algorithms by an order of magnitude, from 200 to 2000. PSO or CMA-ES do not achieve lower deviations than Phoenics for any of the objective functions after 200 evaluations. Even after increasing the number of allowed evaluations by an order of magnitude, CMA-ES fails to achieve lower deviations than Phoenics for 13 out of 15 objective functions, and PSO fails for 12 out of 15 functions as indicated in Fig. S.6.

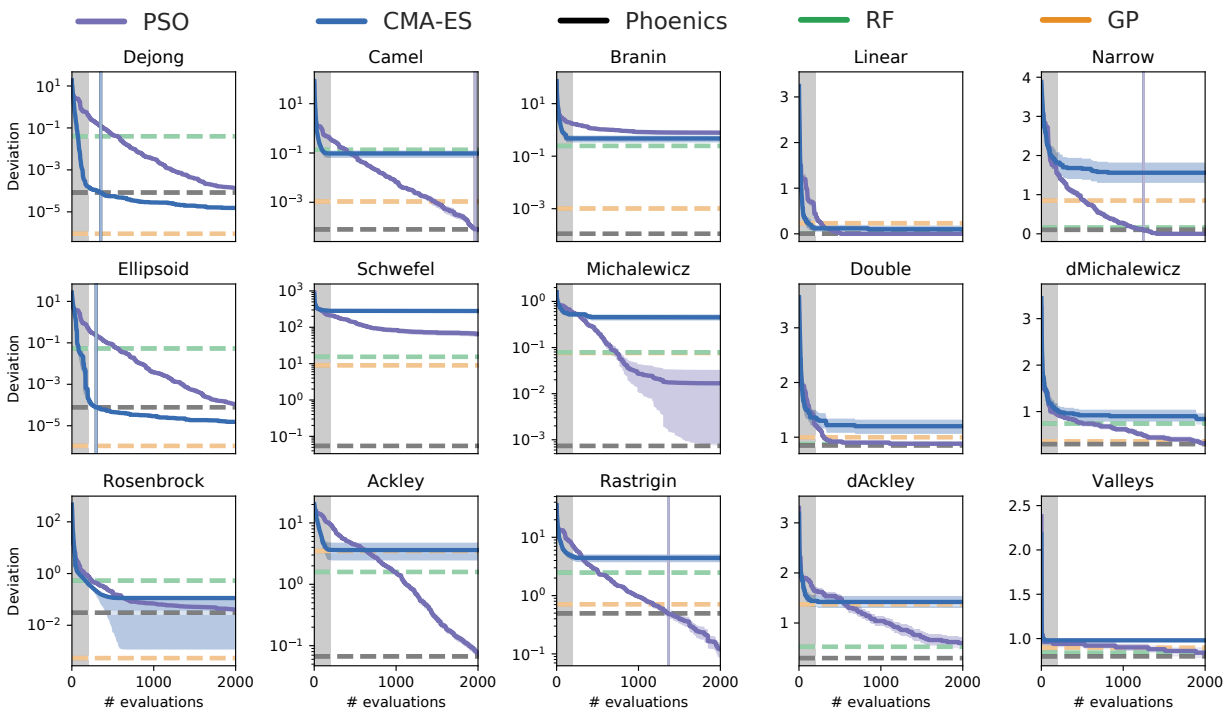

Figure S.6: Deviations between lowest achieved objective function values and their global minima for a total of 20 independent runs with particle swarm optimization (PSO) and the covariance matrix adaptation evolution strategy (CMA-ES). Optimization procedures with both algorithms were carried out for 2000 function evaluations. Grey boxes highlight the region of the first 200 function evaluations, for which optimization procedures were executed with Bayesian optimization algorithms (see Fig. S.5). Dashed lines indicate the deviations achieved by the Bayesian optimization algorithms after 200 evaluations. Vertical lines highlight the number of function evaluations required by PSO or CMA-ES to achieve lower deviations than Phoenics after 200 evaluations.

S.10 Belousov-Zhabotinsky reaction mechanism

The Belousov-Zhabotinsky reaction is a prominent example of a nonlinear chemical oscillator.[2,3] While the detailed reaction mechanism is rather complex and involved a large number of elementary subreactions,[4] the reaction can be summarized in three major subprocesses (see reaction 1).

In reaction (A) a bromite ion is reduced by a bromide ion through a series of two-electron reductions in which malonic acid reacts to bromomalonic acid. Reaction (B) dominates over reaction (A) at low bromide ion concentrations and forms Ce^{IV} from Ce^{III} while consuming

bromite ions. Reaction (C) then removes the Ce^{IV} produced by reaction (B).

$$\frac{d\alpha}{d\tau} = s(\eta - \eta\alpha + \alpha - q\alpha^2), \tag{27}$$

$$\frac{d\eta}{d\tau} = s^{-1}(-\eta - \eta\alpha + f\rho), \tag{28}$$

$$\frac{d\rho}{d\tau} = w(\alpha - \rho). \tag{29}$$

Depending on the particular choice of reaction parameters for the Oregonator model the solutions of the differential equations can differ quantitatively and qualitatively. The set of target parameters features an oscillatory solution with a stable attractive limit cycle. Fig. S.7 illustrates different possible reduced concentration traces for different values of the constructed objective function. All presented concentration traces were sampled in a single optimization run of Phoenics .

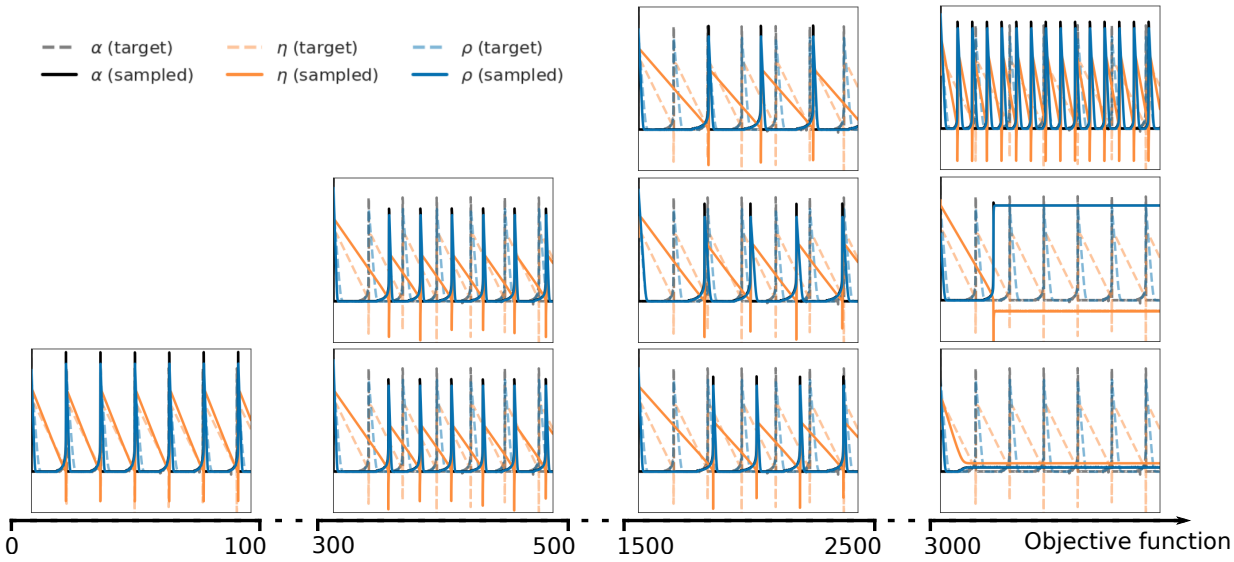

Figure S.7: Examples for concentration traces related to different loss values. Concentration traces were obtained from different parameter sets all sampled by Phoenics within a single optimization run.

We find that loss values below 100 closely resemble the target concentration traces. For such low loss values, we have qualitative and quantitative agreement between the traces.

Loss values between 300 and 500 feature simulations for which the periodicity of the sampled concentration traces matches the periodicity of the target traces, but the traces are shifted by a phase. Slightly different periodicities are developed for losses between 1500 and 2500 while finally at losses above 3000 the system shows rapid oscillations or even steady states.

References

(1) Häse, F.; Roch, L. M.; Kreisbeck, C.; Aspuru-Guzik, A. PHOENICS: A Universal Deep Bayesian Optimizer (https://github.com/aspuru-guzik-group/phoenics). *GitHub* **2018**, DOI: `https://github.com/aspuru-guzik-group/phoenics`.

(2) Zhabotisnky, A. M.; Zaikin, A. N. Oscillatory Processes in Biological and Chemical Systems. *Izdatelstro "Nauka" Publishers, Moscow* **1967**,

(3) Deng, H. Effect of Bromine Derivatives of Malonic Acid on the Oscillating Reaction of Malonic Acid, Cerium Ions and Bromate. *Nature* **1967**, *213*, 589–590.

(4) Gyorgyi, L.; Turányi, R.; Field, R. J. Mechanistic Details of the Oscillatory Belousov-Zhabotinski Reaction. *J. Phys. Chem.* **1990**, *94*, 7162–7170.