Global optimization via inverse distance weighting and radial basis functions

Alberto Bemporad[1] (ORCID)

Abstract

Global optimization problems whose objective function is expensive to evaluate can be solved effectively by recursively fitting a *surrogate function* to function samples and minimizing an *acquisition function* to generate new samples. The acquisition step trades off between seeking for a new optimization vector where the surrogate is minimum (*exploitation* of the surrogate) and looking for regions of the feasible space that have not yet been visited and that may potentially contain better values of the objective function (*exploration* of the feasible space). This paper proposes a new global optimization algorithm that uses inverse distance weighting (IDW) and radial basis functions (RBF) to construct the acquisition function. Rather arbitrary constraints that are simple to evaluate can be easily taken into account. Compared to Bayesian optimization, the proposed algorithm, that we call GLIS (GLobal minimum using Inverse distance weighting and Surrogate radial basis functions), is competitive and computationally lighter, as we show in a set of benchmark global optimization and hyperparameter tuning problems. MATLAB and Python implementations of GLIS are available at http://cse.lab.imtlucca.it/~bemporad/glis.

Keywords Global optimization · Inverse distance weighting · Bayesian optimization · Radial basis functions · Surrogate models · Derivative-free algorithms · Black-box optimization

1 Introduction

Many problems in machine learning and statistics, engineering design, physics, medicine, management science, and in many other fields, require finding a global minimum of a function without derivative information; see, e.g., the excellent survey on derivative-free optimization [32]. Some of the most successful approaches for derivative-free global optimization include deterministic methods based on

✉ Alberto Bemporad
alberto.bemporad@imtlucca.it

[1] IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy

recursively splitting the feasible space in rectangles, such as the DIRECT (DIvide a hyper-RECTangle) [22] and Multilevel Coordinate Search (MCS) [18] algorithms, and stochastic methods such as Particle Swarm Optimization (PSO) [12], genetic algorithms [43], and evolutionary algorithms [16].

The aforementioned methods can be very successful in reaching a global minimum without any assumption on convexity and smoothness of the function, but may result in evaluating the function a large number of times during the execution of the algorithm. In many problems, however, the objective function is a black box that can be very time-consuming to evaluate. For example, in hyperparameter tuning of machine learning algorithms, one needs to run a large set of training tests per hyperparameter choice; in structural engineering design, testing the resulting mechanical property corresponding to a given choice of parameters may involve several hours for computing solutions to partial differential equations; in control systems design, testing a combination of controller parameters involves running a real closed-loop experiment, which is time consuming and costly. For this reason, many researchers have been studying algorithms for black-box global optimization that aim at minimizing the number of function evaluations by replacing the function to minimize with a *surrogate* function [21]. The latter is obtained by sampling the objective function and interpolating the samples with a map that, compared to the original function, is very cheap to evaluate. The surrogate is then used to solve a (much cheaper) global optimization problem that decides the new point where the original function must be evaluated. A better-quality surrogate is then created by also exploiting the new sample and the procedure is iterated. For example, quadratic surrogate functions are used in the well known global optimization method NEWUOA [30].

As underlined by several authors (see, e.g., [21]), purely minimizing the surrogate function may lead to converge to a point that is not the global minimum of the black-box function. To take into account the fact that the surrogate and the true objective function differ from each other in an unknown way, the surrogate is typically augmented by an extra term that takes into account such an uncertainty. The resulting *acquisition function* is therefore minimized instead for generating a new sample of the optimization vector, trading off between seeking for a new vector where the surrogate is small and looking for regions of the feasible space that have not yet been visited.

Bayesian Optimization (BO) is a popular class of global optimization methods based on surrogates that, by modeling the black-box function as a Gaussian process, enables one to quantify in statistical terms the discrepancy between the two functions, an information that is taken into account to drive the search. BO has been studied since the sixties in global optimization [25] and in geostatistics [26] under the name of Kriging methods; it become popular to solve problems of Design and Analysis of Computer Experiments (DACE) [34], see for instance the popular Efficient Global Optimization (EGO) algorithm [23]. It is nowadays very popular in machine learning for tuning hyperparameters of different algorithms [9, 13, 35, 37].

Motivated by learning control systems from data [29] and self-calibration of optimal control parameters [14], in this paper we propose an alternative approach to solve global optimization problems in which the objective function is expensive to evaluate that is based on Inverse Distance Weighting (IDW) interpolation [24, 36]

and Radial Basis Functions (RBFs) [17, 27]. The use of RBFs for solving global optimization problems was already adopted in [11, 15], in which the acquisition function is constructed by introducing a "measure of bumpiness". The author of [15] shows that such a measure has a relation with the probability of hitting a lower value than a given threshold of the underlying function, as used in Bayesian optimization. RBFs were also adopted in [31], with additional constraints imposed to make sure that the feasible set is adequately explored. In this paper we use a different acquisition function based on two exploration components: an estimate of the confidence interval associated with the interpolant function, defined as in [24], and a new measure based on inverse distance weighting that is totally independent of the underlying black-box function and its surrogate. Both terms aim at exploring the domain of the optimization vector. Moreover, arbitrary constraints that are simple to evaluate are also taken into account, as they can be easily imposed during the minimization of the acquisition function.

Compared to Bayesian optimization, our non-probabilistic approach to global optimization is very competitive, as we show in a set of benchmark global optimization problems and on hyperparameter selection problems, and also computationally lighter than off-the-shelf implementations of BO.

A preliminary version of this manuscript was made available in [4] and later extended in [5] to solve preference-based optimization problems. MATLAB and a Python implementations of the proposed approach and of the one of [5] are available for download at http://cse.lab.imtlucca.it/~bemporad/glis. For an application of the GLIS algorithm proposed in this paper to learning optimal calibration parameters in embedded model predictive control applications the reader is referred to [14].

The paper is organized as follows. After stating the global optimization problem we want to solve in Sect. 2, Sects. 3 and 4 deal with the construction of the surrogate and acquisition functions, respectively. The proposed global optimization algorithm is detailed in Sect. 5 and several results are reported in Sect. 6. Finally, some conclusions are drawn in Sect. 7.

2 Problem formulation

Consider the following constrained global optimization problem

$$
\begin{aligned}
&\min_x f(x) \\
&\text{s.t. } \ell \leq x \leq u \\
&\qquad x \in \mathcal{X}
\end{aligned}
\tag{1}
$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is an arbitrary function of the optimization vector $x \in \mathbb{R}^n$, $\ell, u \in \mathbb{R}^n$ are vectors of lower and upper bounds, and $\mathcal{X} \subseteq \mathbb{R}^n$ imposes further arbitrary constraints on x. Typically $\mathcal{X} = \{x \in \mathbb{R}^n : g(x) \leq 0\}$, where the vector function $g : \mathbb{R}^n \to \mathbb{R}^q$ defines inequality constraints, with $q = 0$ meaning that no inequality constraint is enforced; for example, linear inequality constraints are defined by setting $g(x) = Ax - b$, with $A \in \mathbb{R}^{q \times n}$, $b \in \mathbb{R}^q$, $q \geq 0$. We are particularly interested in problems as in (1) such that $f(x)$ is expensive to evaluate and its gradient

is not available, while the condition $x \in \mathcal{X}$ is easy to evaluate. Although not comprehensively addressed in this paper, we will show that our approach also tolerates noisy evaluations of f, that is if we measure $y = f(x) + \varepsilon$ instead of $f(x)$, where ε is an unknown quantity. We will not make any assumption on f, g, and ε. In (1) we do not include possible linear equality constraints $A_e x = b_e$, as they can be first eliminated by reducing the number of optimization variables and therefore perform the exploration more easily in a lower dimensional space.

3 Surrogate function

Assume that we have collected a vector $F = [f_1 \ \ldots \ f_N]'$ of N samples $f_i = f(x_i)$ of f, $F \in \mathbb{R}^N$ at corresponding points $X = [x_1 \ \ldots \ x_N]'$, $X \in \mathbb{R}^{N \times n}$, with $x_i \neq x_j$, $\forall i \neq j$, $i, j = 1, \ldots, N$. We consider next two types of surrogate functions, namely *Inverse Distance Weighting* (IDW) functions [24, 36] and *Radial Basis Functions* (RBFs) [15, 27].

3.1 Inverse distance weighting functions

Given a generic new point $x \in \mathbb{R}^n$ consider the squared Euclidean distance function $d^2 : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$

$$d^2(x, x_i) = (x_i - x)'(x_i - x), \quad i = 1, \ldots, N \tag{2}$$

In standard IDW functions [36] the weight functions $w_i : \mathbb{R}^n \backslash \{x_i\} \to \mathbb{R}$ are defined by the inverse squared distances

$$w_i(x) = \frac{1}{d^2(x, x_i)} \tag{3a}$$

The alternative weighting function

$$w_i(x) = \frac{e^{-d^2(x, x_i)}}{d^2(x, x_i)} \tag{3b}$$

suggested in [24] has the advantage of being similar to the inverse squared distance in (3a) for small values of d^2, but makes the effect of points x_i located far from x fade out quickly due to the exponential term.

By defining for $i = 1, \ldots, N$ the following functions $v_i : \mathbb{R}^n \to \mathbb{R}$ as

$$v_i(x) = \begin{cases} 1 & \text{if } x = x_i \\ 0 & \text{if } x = x_j, \ j \neq i \\ \dfrac{w_i(x)}{\sum_{j=1}^{N} w_j(x)} & \text{otherwise} \end{cases} \tag{4}$$

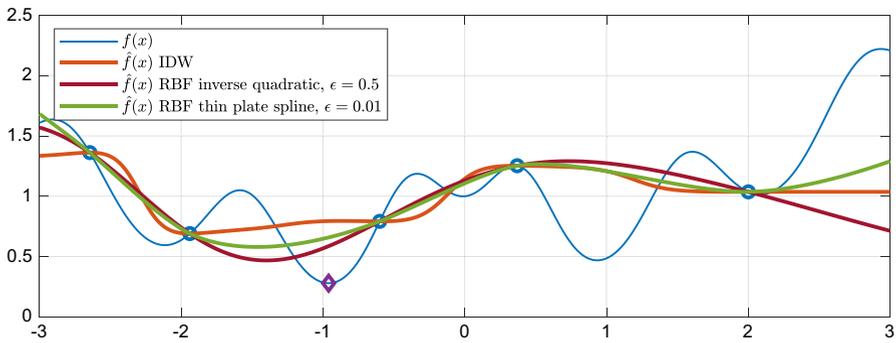the surrogate function $\hat{f} : \mathbb{R}^n \to \mathbb{R}$

Fig. 1 A scalar example of $f(x)$ as in (6) (blue) sampled at $N = 5$ points (blue circles), IDW surrogate $\hat{f}(x)$ (orange) with $w_i(x)$ as in (3b), RBF inverse quadratic with $\epsilon = 0.5$ (red), RBF thin plate spline surrogate with $\epsilon = 0.01$ (green), global minimum (purple diamond) (Color figure online)

$$\hat{f}(x) = \sum_{i=1}^{N} v_i(x)f_i \qquad (5)$$

is an IDW interpolation of (X, F).

Lemma 1 *The IDW interpolation function $\hat{f}$ defined in* (5) *enjoys the following properties*:

> P1. $\hat{f}(x_j) = f_j, \forall j = 1, \ldots, N$;
> P2. $\min_j\{f_j\} \le \hat{f}(x) \le \max_j\{f_j\}, \forall x \in \mathbb{R}^n$;
> P3. $\hat{f}$ *is differentiable everywhere on $\mathbb{R}^n$ and in particular $\nabla f(x_j) = 0$ for all $j = 1, \ldots, N$.*

The proof of Lemma 1 is very simple and is reported in "Appendix A".

Note that in [24] the authors suggest to improve the surrogate function by adding a regression model in (5) to take global trends into account. In our numerical experiments we found, however, that adding such a term does not lead to significant improvements of the proposed global optimization algorithm.

A one-dimensional example of the IDW surrogate $\hat{f}$ sampled at five different points of the scalar function

$$f(x) = \left(1 + \frac{x\sin(2x)\cos(3x)}{1+x^2}\right)^2 + \frac{x^2}{12} + \frac{x}{10} \qquad (6)$$

is depicted in Fig. 1. The global optimizer is $x^* \approx -0.9599$ corresponding to the global minimum $f(x^*) \approx 0.2795$.

3.2 Radial basis functions

A possible drawback of the IDW function $\hat{f}$ defined in (5) is due to property P3: as the number N of samples increases, the surrogate function tends to ripple, having its derivative to always assume zero value at samples. An alternative is to use a radial basis function (RBF) [15, 27] as a surrogate function. These are defined by setting

$$\hat{f}(x) = \sum_{i=1}^{N} \beta_i \phi(\epsilon d(x, x_i)) \tag{7}$$

where $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is the function defining the Euclidean distance as in (2), $d(x, x_i) = \|x - x_i\|_2$, $\epsilon > 0$ is a scalar parameter, β_i are coefficients to be determined as explained below, and $\phi : \mathbb{R} \to \mathbb{R}$ is a RBF. Popular examples of RBFs are

$$
\begin{array}{llll}
\phi(\epsilon d) = \frac{1}{1+(\epsilon d)^2} & \text{(inverse quadratic)} & \phi(\epsilon d) = e^{-(\epsilon d)^2} & \text{(Gaussian)} \\
\phi(\epsilon d) = \sqrt{1 + (\epsilon d)^2} & \text{(multiquadric)} & \phi(\epsilon d) = (\epsilon d)^2 \log(\epsilon d) & \text{(thin plate spline)} \\
\phi(\epsilon d) = \epsilon d & \text{(linear)} & \phi(\epsilon d) = \frac{1}{\sqrt{1+(\epsilon d)^2}} & \text{(inverse multiquadric)}
\end{array}
\tag{8}
$$

The coefficient vector $\beta = [\beta_1 \ \ldots \ \beta_N]'$ is obtained by imposing the interpolation condition

$$\hat{f}(x_i) = f_i, \quad i = 1, \ldots, N \tag{9}$$

Condition (9) leads to solving the linear system

$$M\beta = F \tag{10a}$$

where M is the $N \times N$ symmetric matrix whose (i, j)-entry is

$$M_{ij} = \phi(\epsilon d(x_i, x_j)) \tag{10b}$$

with $M_{ii} = 1$ for all the RBF type listed in (8) but the linear and thin plate spline, for which $M_{ii} = \lim_{d \to 0} \phi(\epsilon d) = 0$. Note that if function f is evaluated at a new sample x_{N+1}, matrix M only requires adding the last row/column obtained by computing $\phi(\epsilon d(x_{N+1}, x_j))$ for all $j = 1, \ldots, N + 1$.

As highlighted in [15, 21], matrix M might be singular, even if $x_i \neq x_j$ for all $i \neq j$. To prevent issues due to a singular M, [15, 21] suggest using a surrogate function given by the sum of a RBF and a polynomial function of a certain degree. To also take into account unavoidable numerical issues when distances between sampled points get close to zero, which will easily happen as new samples are added towards finding a global minimum, in this paper we suggest instead to use a singular value decomposition (SVD) $M = U\Sigma V'$ of M.[1] By neglecting singular values below a certain positive

---

[1] Matrices U and V have the same columns, modulo a change a sign. Indeed, as M is symmetric, we could instead solve the symmetric eigenvalue problem $M = T'\Lambda T$, $T'T = I$, which gives $\Sigma_{ii} = |\Lambda_{ii}|$, and set $U = V = T'$. As N will be typically small, we neglect computational advantages and adopt here SVD decomposition.
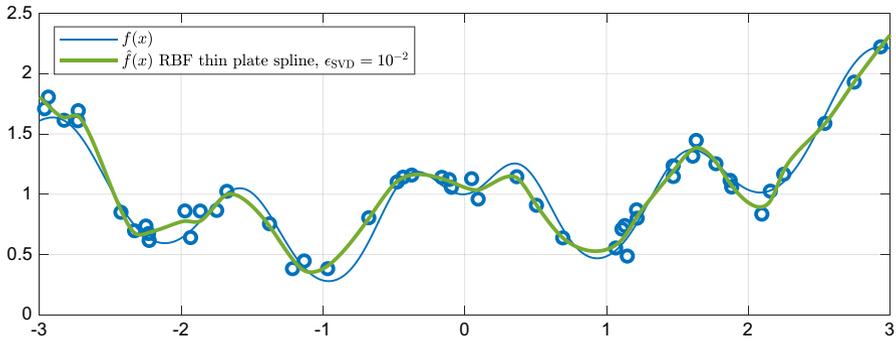
Fig. 2 Function $f(x)$ as in (6) is sampled 50 times, with each sample corrupted by noise $\varepsilon \sim \mathcal{N}(0, 10^{-2})$ (blue). The RBF thin plate spline surrogate with $\varepsilon = 0.01$ (green) is obtained by setting $\epsilon_{\text{SVD}} = 10^{-2}$ (Color figure online)

threshold ϵ_{SVD}, we can approximate $\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix}$, where Σ_1 collects all singular values $\sigma_i \geq \epsilon_{\text{SVD}}$, and accordingly split $V = [V_1 \ V_2], U = [U_1 \ U_2]$ so that

$$\beta = V_1 \Sigma_1^{-1} U_1' F \tag{10c}$$

The threshold ϵ_{SVD} turns out to be useful when dealing with noisy measurements $y = f(x) + \varepsilon$ of f. Figure 2 shows the approximation $\hat{f}$ obtained from 50 samples with ε normally distributed around zero with standard deviation 0.1, when $\epsilon_{\text{SVD}} = 10^{-2}$.

A drawback of RBFs, compared to IDW functions, is that property P2 is no longer satisfied, with the consequence that the surrogate may extrapolate large values $\hat{f}(x)$ where $f(x)$ is actually small, and vice versa. See the examples plotted in Fig. 1. On the other hand, while differentiable everywhere, RBFs do not necessarily have zero gradients at sample points as in P3, which is favorable to better approximate the underlying function with limited samples. For the above reasons, we will mostly focus on RBF surrogates in our numerical experiments.

3.3 Scaling

To take into account that different components x^j of x may have different ranges $u^j - \ell^j$, we simply rescale the variables in optimization problem (1) so that they all range in $[-1, 1]$. To this end, we first possibly tighten the given box constraints $B_{\ell,u} = \{x \in \mathbb{R}^n : \ell \leq x \leq u\}$ by computing the bounding box B_{ℓ_g,u_g} of the set $\{x \in \mathbb{R}^n : x \in \mathcal{X}\}$ and replacing $B_{\ell,u} \leftarrow B_{\ell,u} \cap B_{\ell_g,u_g}$. The bounding box B_{ℓ_g,u_g} is obtained by solving the following $2n$ optimization problems

$$\begin{aligned} \ell_g^i &= \min_{x \in \mathcal{X}} e_i' x \\ u_g^i &= \max_{x \in \mathcal{X}} e_i' x \end{aligned} \tag{11a}$$

where e_i is the ith column of the identity matrix, $i = 1, \ldots, n$. In case of linear inequality constraints $\mathcal{X} = \{x \in \mathbb{R}^n : Ax \leq b\}$, the problems in (11a) can be solved by

linear programming (LP), as shown later in (17). Since now on, we assume that ℓ, u are replaced by

$$
\begin{aligned}
\ell &\leftarrow \max\{\ell, \ell_g\} \\
u &\leftarrow \min\{u, u_g\}
\end{aligned}
\tag{11b}
$$

where "min" and "max" in (11b) operate component-wise. Next, we introduce scaled variables $\bar{x} \in \mathbb{R}^n$ whose relation with x is

$$
x^j(\bar{x}) = \bar{x}^j \left(\frac{u^j - \ell^j}{2} \right) + \frac{u^j + \ell^j}{2}
\tag{12a}
$$

for all $j = 1, \ldots, n$ and finally formulate the following scaled global optimization problem

$$
\begin{aligned}
&\min f_s(\bar{x}) \\
&\text{s.t.} \ -1 \leq \bar{x}^j \leq 1, \quad j = 1, \ldots, n \\
&\quad \bar{x} \in \mathcal{X}_s
\end{aligned}
\tag{12b}
$$

where $f_s : \mathbb{R}^n \to \mathbb{R}, \mathcal{X}_s$ are defined as

$$
\begin{aligned}
f_s(\bar{x}) &= f(x(\bar{x})) \\
\mathcal{X}_s &= \{\bar{x} \in \mathbb{R}^n : x(\bar{x}) \in \mathcal{X}\}
\end{aligned}
$$

In case $\mathcal{X}$ is a polyhedron we have

$$
\mathcal{X}_s = \{\bar{x} : \bar{A}\bar{x} \leq \bar{b}\}
\tag{12c}
$$

where $\bar{A}, \bar{b}$ are a rescaled version of A, b defined as

$$
\begin{aligned}
\bar{A} &= A \, \text{diag}\left(\frac{u-\ell}{2} \right) \\
\bar{b} &= b - A\left(\frac{u+\ell}{2} \right)
\end{aligned}
\tag{12d}
$$

and $\text{diag}(\frac{u-\ell}{2})$ is the diagonal matrix whose diagonal elements are the components of $\frac{u-\ell}{2}$.

Note that, when approximating f_s with $\hat{f}_s$, we use the squared Euclidean distances

$$
d^2(\bar{x}, \bar{x}_i) = \sum_{h=1}^n (\bar{x} - \bar{x}_i)^2 = \sum_{h=1}^n \left(\theta^h (x^h - x_i^h) \right)^{p^h}
$$

where the scaling factors $\theta^h = \frac{u^h - \ell^h}{2}$ and $p^h \equiv 2$ are constant. Therefore, finding a surrogate $\hat{f}_s$ of f_s in $[-1, 1]$ is equivalent to finding a surrogate $\hat{f}$ of f under scaled distances. This is a much simpler scaling approach than computing the scaling factors θ^h and power p as it is common in stochastic process model approaches such as Kriging methods [23, 34]. As highlighted in [21], Kriging methods use radial basis functions $\phi(x_i, x_j) = e^{-\sum_{h=1}^n \theta^h |x_i^h - x_j^h|^{p^h}}$, a generalization of Gaussian RBF functions in which the scaling factors and powers are recomputed as the data set X changes.

Note also that the approach adopted in [5] for scaling automatically the surrogate function via cross-validation could be also used here, as well as other approaches specific for RBFs such as Rippa's method [33]. In our numerical experiments we have found that adjusting the RBF parameter ϵ via cross-validation, while increasing the computational effort, does not provide significant benefit. What is in fact most critical is the tradeoff between exploitation of the surrogate and exploration of the feasible set, that we discuss in the next section.

4 Acquisition function

As mentioned earlier, minimizing the surrogate function to get a new sample $x_{N+1} =$ arg min $\hat{f}(x)$ subject to $\ell \leq x \leq u$ and $x \in \mathcal{X}$, evaluating $f(x_{N+1})$, and iterating over N may easily miss the global minimum of f. This is particularly evident when $\hat{f}$ is the IDW surrogate (5), that by Property P2 of Lemma 1 has a global minimum at one of the existing samples x_i. Besides *exploiting* the surrogate function $\hat{f}$, when looking for a new candidate optimizer x_{N+1} it is therefore necessary to add to $\hat{f}$ a term for *exploring* areas of the feasible space that have not yet been probed.

In Bayesian optimization, such an exploration term is provided by the covariance associated with the Gaussian process. A function measuring "bumpiness" of a surrogate RBF function was used in [15]. Here instead we propose two functions that provide exploration capabilities, that can be used in alternative to each other or in a combined way. First, as suggested in [24] for IDW functions, we consider the confidence interval function $s : \mathbb{R}^n \to \mathbb{R}$ for $\hat{f}$ defined by

$$s(x) = \sqrt{\sum_{i=1}^{N} v_i(x)(f_i - \hat{f}(x))^2} \tag{13}$$

We will refer to function s as the *IDW variance function* associated with (X, F). Clearly, when $\hat{f}(x_i) = f(x_i)$ then $s(x_i) = 0$ for all $i = 1, \ldots, N$ (no uncertainty at points x_i where f is evaluated exactly). See Fig. 3 for a noise-free example and Fig. 4 for the case of noisy measurements of f.

Second, we introduce the new *IDW distance function* $z : \mathbb{R}^n \to \mathbb{R}$ defined by

$$z(x) = \begin{cases} 0 & \text{if } x \in \{x_1, \ldots, x_N\} \\ \frac{2}{\pi} \tan^{-1}\left(\frac{1}{\sum_{i=1}^{N} w_i(x)}\right) & \text{otherwise} \end{cases} \tag{14}$$

where $w_i(x)$ is given by either (3a) or (3b). The rationale behind (14) is that $z(x)$ is zero at sampled points and grows in between. The arctangent function in (14) avoids that $z(x)$ grows excessively when x is located far away from all sampled points. Figure 5 shows a scalar example of functions v_1 and z.

Given parameters $\alpha, \delta \geq 0$ and N samples (X, F), we define the following *acquisition function $a : \mathbb{R}^n \to \mathbb{R}$*
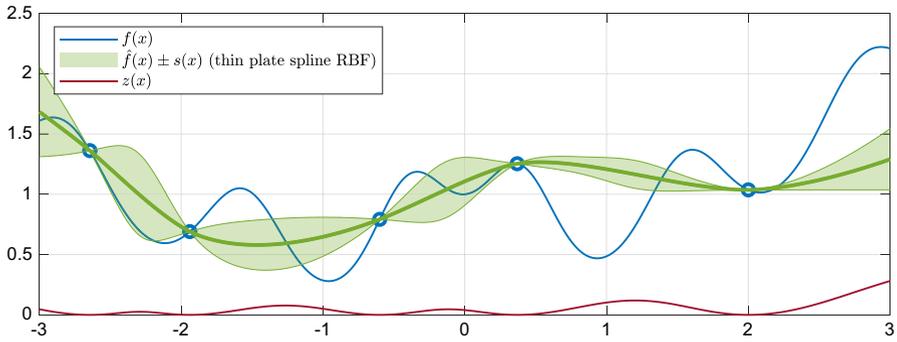
Fig. 3 Plot of $\hat{f}(x) \pm s(x)$ and $z(x)$ for the scalar example as in Fig. 1, with $w_i(x)$ as in (3a) and f as in (6)
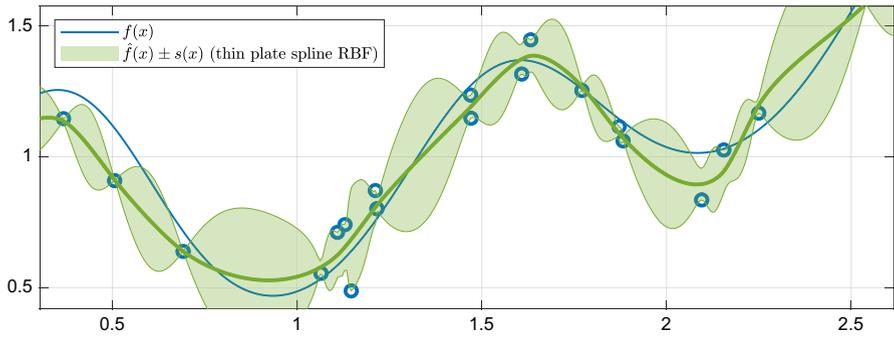

Fig. 4 Zoomed plot of $\hat{f}(x) \pm s(x)$ for the scalar example as in Fig. 3 when 50 samples of $f(x)$ are measured with noise $\varepsilon \sim \mathcal{N}(0, 10^{-2})$ and $\epsilon_{\mathrm{SVD}} = 10^{-2}$
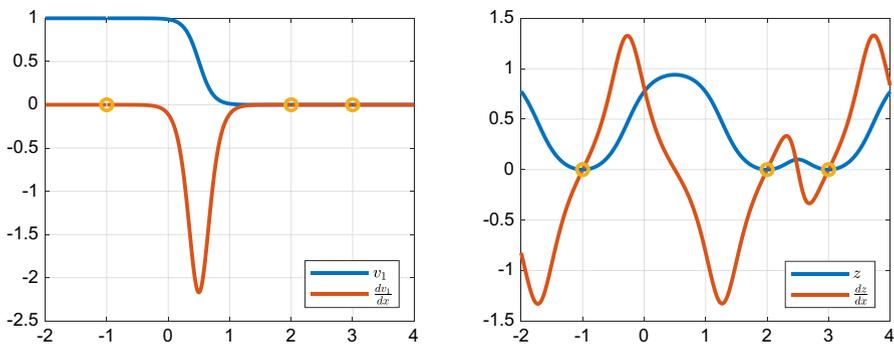

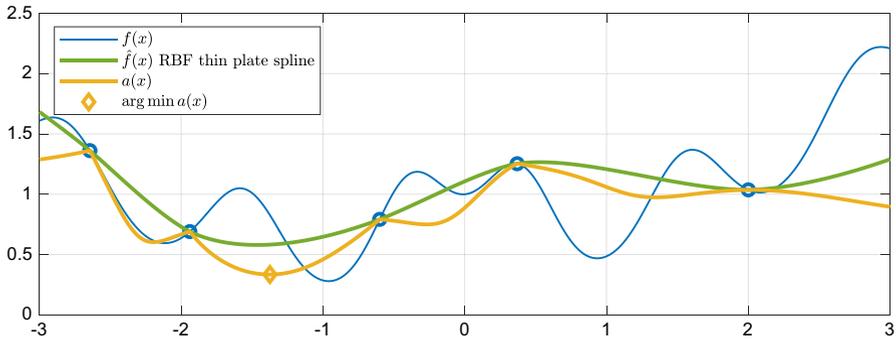**Fig. 5** A scalar example of functions $v_1(x)$ and $z(x)$ for $x_1 = -1, x_2 = 2, x_3 = 3$

Fig. 6 Plot of $\hat{f}(x)$ and acquisition function $a(x)$ with $\alpha = 1$, $\delta = \frac{1}{2}$, thin plate spline RBF with $\epsilon_{SVD} = 10^{-6}$, for the scalar example as in Fig. 1, with $w_i(x)$ as in (3a) and f as in (6)

$$a(x) = \hat{f}(x) - \alpha s(x) - \delta \Delta F z(x) \tag{15}$$

where $\Delta F = \max\{\max_i\{f_i\} - \min_i\{f_i\}, \epsilon_{\Delta F}\}$ is the range of the observed samples F and the threshold $\epsilon_{\Delta F} > 0$ is introduced to prevent the case in which f is not a constant function but, by chance, all sampled values f_i are equal. Scaling z by ΔF eases the selection of the hyperparameter δ, as the amplitude of $\Delta F z$ is comparable to that of $\hat{f}$.

As we will detail next, given N samples (X, F) a global minimum of the acquisition function (15) is used to define the $(N + 1)$th sample x_{N+1} by solving the global optimization problem

$$x_{N+1} = \arg \min_{\ell \leq x \leq u, \, x \in \mathcal{X}} a(x) \tag{16}$$

The rationale behind choosing (15) for acquisition is the following. The term $\hat{f}$ directs the search towards a new sample x_{N+1} where the objective function f is expected to be optimal, assuming that f and its surrogate $\hat{f}$ have a similar shape, and therefore allows a direct *exploitation* of the samples F already collected. The other two terms account instead for the *exploration* of the feasible set with the hope of finding better values of f, with s promoting areas in which $\hat{f}$ is more uncertain and z areas that have not been explored yet. Both s and z provide exploration capabilities, but with an important difference: function z is totally independent on the samples F already collected and promotes a more uniform exploration, s instead depends on F and the surrogate $\hat{f}$. The coefficients α, δ determine the exploitation/exploration tradeoff one desires to adopt.

For the example of scalar function f in (6) sampled at five random points, the acquisition function a obtained by setting $\alpha = 1$, $\delta = \frac{1}{2}$, using a thin plate spline RBF with $\epsilon_{SVD} = 10^{-6}$, and $w_i(x)$ as in (3a), and the corresponding minimum are depicted in Fig. 6.

The following result, whose easy proof is reported in "Appendix A", highlights a nice property of the acquisition function a.

Lemma 2 *Function a is differentiable everywhere on* $\mathbb{R}^n$.

Problem (16) is a global optimization problem whose objective function and constraints are very easy to evaluate. It can be solved very efficiently using various global optimization techniques, either derivative-free [32] or, if $\mathcal{X} = \{x : g(x) \leq 0\}$ and g is also differentiable, derivative-based. In case some components of vector x are restricted to be integer, (16) can be solved by mixed-integer programming.

5 Global optimization algorithm

Algorithm 1, that we will refer to as GLIS (GLobal minimum using Inverse distance weighting and Surrogate radial basis functions), summarizes the proposed approach to solve the global optimization problem (1) using surrogate functions (either IDW or RBF) and the IDW acquisition function (15).

Algorithm 1 GLIS – Global optimization algorithm based on IDW-RBF surrogates

---

Input: Upper and lower bounds (ℓ, u), constraint set $\mathcal{X}$; number N_{init} of initial samples, number $N_{\max} \geq N_{\text{init}}$ of maximum number of function evaluations; $\alpha, \delta \geq 0$, $\epsilon_{\text{SVD}} > 0$, $\epsilon_{\Delta F} > 0$.

---

1. Tighten (ℓ, u) as in (11);
2. Scale problem as in (12);
3. Set $N \leftarrow N_{\text{init}}$;
4. Generate N random initial samples $X = [x_1 \ \ldots \ x_N]'$ using Latin hypercube sampling [28];
5. Compute $\hat{f}$ as in (7), (10) (RBF function) or as in (5) (IDW function);
6. **While** $N < N_{\max}$ **do**
     6.1. Compute acquisition function a as in (15);
     6.2. Solve global optimization problem (16) and get x_{N+1};
     6.3. $N \leftarrow N + 1$;
7. **End**.

Output: Approximation $x^* = x_{N_{\max}}$ of the global optimizer.

As common in global optimization based on surrogate functions, in Step 4 *Latin Hypercube Sampling* (LHS) [28] is used to generate the initial set X of samples in the given range ℓ, u. Note that the generated initial points may not satisfy the inequality constraints $x \in \mathcal{X}$. We distinguish between two cases:

(i)   the objective function f can be evaluated outside the feasible set $\mathcal{F}$;
(ii) f cannot be evaluated outside $\mathcal{F}$.

In the first case, initial samples of f falling outside $\mathcal{F}$ are still useful to define the surrogate function and can be therefore kept. In the second case, since f cannot be evaluated at initial samples outside $\mathcal{F}$, a possible approach is to generate more than N_{init} samples and discard the infeasible ones before evaluating f. For example, the author of [6] suggests the simple method reported in Algorithm 2. This requires the feasible set $\mathcal{F}$

to be full-dimensional. In case of linear inequality constraints $\mathcal{X} = \{x : Ax \leq b\}$, full-dimensionality of the feasible set $\mathcal{F}$ can be easily checked by computing the Chebychev radius $r_{\mathcal{F}}$ of $\mathcal{F}$ via the LP [8]

$$
\begin{aligned}
r_{\mathcal{F}} = \max_{r,x} \ & r \\
\text{s.t. } & A_i x \leq b_i - \|A_i\|_2 r, \quad i = 1, \dots, q \\
& \ell_i + r \leq x_i \leq u_i - r, \quad i = 1, \dots, n
\end{aligned}
\tag{17}
$$

where in (17) the subscript i denotes the ith row (component) of a matrix (vector). The polyhedron $\mathcal{F}$ is full dimensional if and only if $r_{\mathcal{F}} > 0$. Clearly, the smaller the ratio between the volume of $\mathcal{F}$ and the volume of the bounding box B_{ℓ_g, u_g}, the larger on average will be the number of samples generated by Algorithm 2.

Note that, in alternative to LHS, the IDW function (14) could be also used to generate N_{init} feasible points by solving

$$
x_{N+1} = \max_{x \in \mathcal{F}} z(x)
$$

for $N = 1, \dots, N_{\text{init}} - 1$, for any $x_1 \in \mathcal{F}$.

Algorithm 2 Latin hypercube sampling with constraints

Input: Upper and lower bounds (ℓ, u) for x and inequality constraint function $g : \mathbb{R}^n \to \mathbb{R}^q$, defining a full dimensional set $\mathcal{F} = \{x \in \mathbb{R}^n : \ell \leq x \leq u, g(x) \leq 0\}$; number N_{init} of initial samples.

---

1. $N \leftarrow N_{\text{init}}; \ N_k \leftarrow 0;$
2. **While** $N_k < N_{\text{init}}$ **do**
   2.1. Generate N samples using Latin hypercube sampling;
   2.2. $N_k \leftarrow$ number of samples satisfying $x \in \mathcal{X}$;
   2.3. If $N_k < N_{\text{init}}$ then increase N by setting

   $$
   N \leftarrow \begin{cases} \lceil \min\{20, 1.1 \frac{N_{\text{init}}}{N_k}\} N \rceil & \text{if } N_k > 0 \\ 20N & \text{otherwise} \end{cases}
   $$

3. **End**.

Output: N_{init} initial samples $X = [x_1 \ \dots \ x_{N_{\text{init}}}]'$ satisfying $\ell \leq x_i \leq u$, $x \in \mathcal{X}$.

The examples reported in this paper use the Particle Swarm Optimization (PSO) algorithm [41] to solve problem (16) at Step 6.2, although several other global optimization methods such as DIRECT [22] or others [18, 32] could be used in alternative. Inequality constraints $\mathcal{X} = \{x : g(x) \leq 0\}$ can be handled as penalty functions, for example by replacing (16) with

$$
x_{N+1} = \arg \min_{\ell \leq x \leq u} \left\{ a(x) + \rho \Delta F \sum_{i=1}^{q} \max\{g_i(x), 0\}^2 \right\}
\tag{18}
$$

where in (18) $\rho \gg 1$. Note that due to the heuristic involved in constructing function a, it is not crucial to find global solutions of very high optimality accuracy when solving problem (16). Regarding feasibility, in case x_{N+1} violates the constraints and
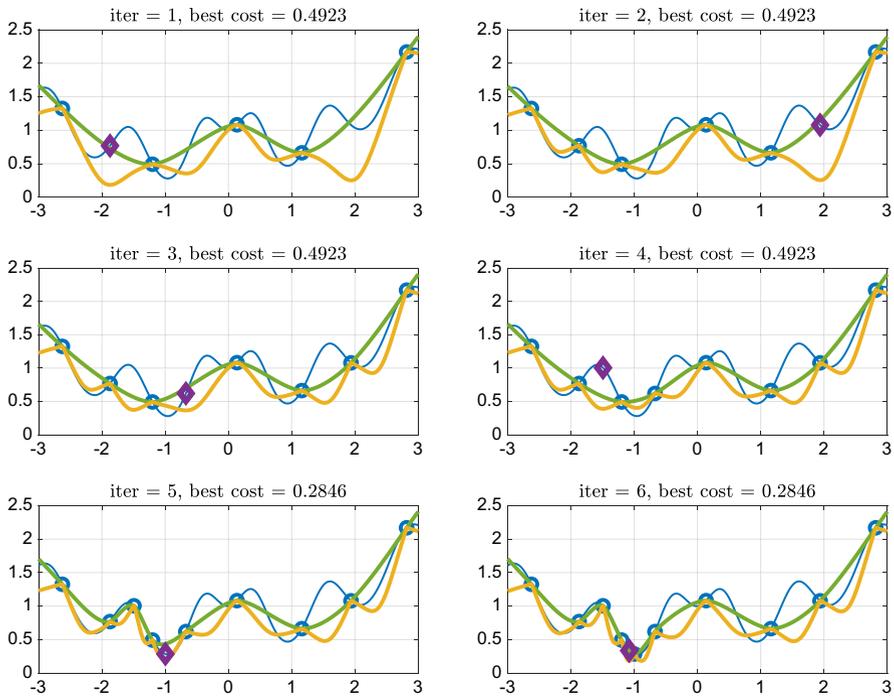
Fig. 7 GLIS steps when applied to minimize the function f given in (6) using the same settings as in Fig. 6 and $\epsilon_{\mathrm{SVD}} = 10^{-6}$. The plots show function f (blue), its samples f_i (blue circles), the thin plate spline interpolation $\hat{f}$ with $\epsilon = 0.01$ (green), the acquisition function a (yellow), and the minimum of the acquisition function reached at x_{N+1} (purple diamond) (Color figure online)

f cannot be evaluated outside $\mathcal{X}$, a remedy would be to increase the penalty parameter ρ and/or to slightly tighten the constraints by penalizing $\max\{g_i(x) + \epsilon_g, 0\}$ in (18) instead of $\max\{g_i(x), 0\}$, for some small positive scalar ϵ_g.

The exploration parameter α promotes visiting points in $[\ell, u]$ where the function surrogate has largest variance, δ promotes instead pure exploration independently on the surrogate function approximation, as it is only based on the sampled points $x_1, \ldots, x_N$ and their mutual distance. For example, if $\alpha = 0$ and $\delta \gg 1$ Algorithm 1 will try to explore the entire feasible region, with consequent slower detection of points x with low cost $f(x)$. On the other hand, setting $\delta = 0$ will make GLIS proceed only based on the function surrogate and its variance, that may lead to miss regions in $[\ell, u]$ where a global optimizer is located. For $\alpha = \delta = 0$, GLIS will proceed based on pure minimization of $\hat{f}$ that, as observed earlier, can easily lead to converge away from a global optimizer.

Figure 7 shows the first six iterations of the GLIS algorithm when applied to minimize the function f given in (6) in $[-3, 3]$ with $\alpha = 1$, $\delta = 0.5$.

5.1 Computational complexity

The complexity of Algorithm 1, as a function of the number N_{max} of iterations and dimension n of the optimization space and not counting the complexity of evaluating f, depends on Steps 6.1 and 6.2. The latter depends on the global optimizer used to solve Problem (16), which typically depends heavily on n. Step 6.1 involves computing $N_{max}(N_{max} - 1)$ RBF values $\phi(\epsilon d(x_i, x_j))$, $i, j = 1, \ldots, N_{max}$, $j \neq i$, compute the SVD decomposition of the $N \times N$ symmetric matrix M in (10a), whose complexity is $O(N^3)$, and solve the linear system in (10a) ($O(N^2)$) at each step $N = N_{init}, \ldots, N_{max}$.

6 Numerical tests

In this section we report different numerical tests performed to assess the performance of the proposed algorithm (GLIS) and how it compares to Bayesian optimization (BO). For the latter, we have used the off-the-shelf algorithm `bayesopt` implemented in the Statistics and Machine Learning Toolbox for MATLAB [39], based on the lower confidence bound as acquisition function. All tests were run on an Intel i7-8550 CPU @1.8GHz machine. Algorithm 1 was run in MATLAB R2019b in interpreted code. The PSO solver [42] was used to solve problem (18). We focus our comparison on BO only as it one of the most efficient methods to deal with the optimization of expensive black-box functions.

6.1 GLIS optimizing its own parameters

We first use GLIS to optimize its own hyperparameters α, δ, ϵ when solving the minimization problem with $f(x)$ as in (6) and $x \in [-3, 3]$. In what follows, we use the subscript $()_H$ to denote the parameters/function used in the execution of the outer instance of Algorithm 1 that is optimizing α, δ, ϵ. To this end, we solve the global optimization problem (1) with $x = [\alpha \; \delta \; \epsilon]'$,

$$f_H(x) = \sum_{i=1}^{N_t} \sum_{h=0}^{N_{max}/2} (h + 1) \min\{f(x_{i,1}), \ldots, f(x_{i,N_{max}/2+h})\} \tag{19}$$

where f is the scalar function in (6) that we want to minimize in $[-3, 3]$, and we set $\ell_H = [0 \; 0 \; 0.1]'$, $u_H = [3 \; 3 \; 3]'$, which is a reasonably large-enough range according to our numerical experience. The min in (19) provides the best objective value found up to iteration $N_{max}/2 + h$, the term $(h + 1)$ aims at penalizing high values of the best objective the more the later they occur during the iterations, $N_t = 20$ is the number of times Algorithm 1 is executed to minimize f_H for the same triplet $(\alpha, \delta, \epsilon)$, $N_{max} = 20$ is the number of times f is evaluated per execution, $x_{i,N}$ is the sample generated by Algorithm 1 during the ith run at step N, $i = 1, \ldots, N_t$, $N = 1, \ldots, N_{max}$. Clearly (19) penalizes failure to convergence close to the global optimum f^* in N_{max} iterations without caring of how the algorithm performs during the first $N_{max}/2 - 1$ iterations.
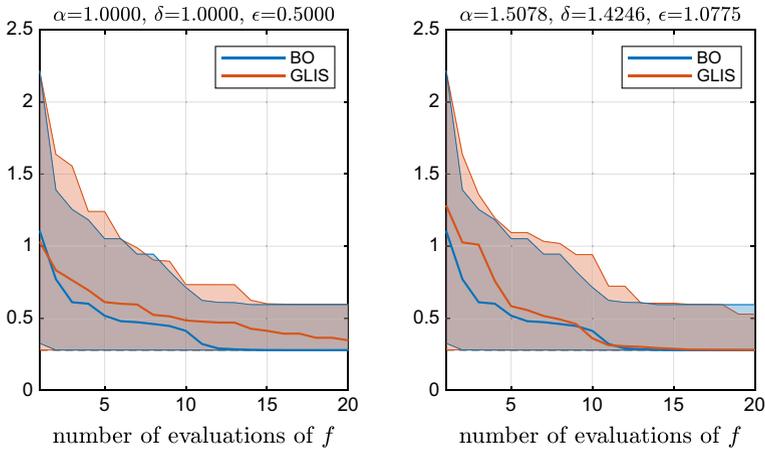
Fig. 8 Minimization of $f(x)$ as in (6) in $[-3, 3]$: tentative hyperparameters (left) and optimal hyperparameters (right). The average performance obtained over $N_{test} = 100$ runs as a function of evaluations of f_H, along with the band defined by the best- and worst-case instances

In optimizing (19), the outer instance of Algorithm 1 is run with $\alpha_H = 2$, $\delta_H = 0.5$, $\epsilon_H = 0.5$, $N_{init,H} = 8$, $N_{max,H} = 100$ [which means that f_H in (19) is evaluated 100 times, each evaluation requiring executing Algorithm 1 $N_t = 20$ times to minimize function f], and PSO as the global optimizer of the acquisition function. The RBF inverse quadratic function is used in both the inner and outer instances of Algorithm 1. The resulting optimal selection is

$$\alpha = 1.5078, \quad \delta = 1.4246, \quad \epsilon = 1.0775 \tag{20}$$

Figure 8 compares the behavior of GLIS (Algorithm 1) when minimizing $f(x)$ as in (6) in $[-3, 3]$ with tentative parameters $\alpha = 1$, $\delta = 1$, $\epsilon = 0.5$ and with the optimal values in (20). The figure also shows the results obtained by using BO on the same problem.

Clearly the results of the hyper-optimization depend on the function f which is minimized in the inner loop. For a more comprehensive and general optimization of GLIS hyperparameters, one could alternatively consider in f_H the average performance with respect to a collection of problems instead of just one problem.

6.2 Benchmark global optimization problems

We test the proposed global optimization algorithm on standard benchmark problems, summarized in Table 1. For each function the table shows the corresponding number of variables, upper and lower bounds, and the name of the example in [19] reporting the definition of the function. For lack of space, we will only consider the GLIS algorithm implemented using inverse quadratic RBFs for the surrogate, leaving IDW only for exploration, because compared to other RBFs it was found a robust choice experimentally.

Table 1 Benchmark problems considered in the comparison

Benchmark problem	n	ℓ	u	Function name [19]	BO[s]	GLIS [s]
ackley	2	$\begin{bmatrix} -5 \\ -5 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 5 \end{bmatrix}$	*Ackley 1, D = 2*	29.39	3.13
adjiman	2	$\begin{bmatrix} -1 \\ -1 \end{bmatrix}$	$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$	*Adjiman*	3.29	0.68
branin	2	$\begin{bmatrix} -5 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 10 \\ 15 \end{bmatrix}$	*Branin RCOS*	9.66	1.17
camelsixhumps	2	$\begin{bmatrix} -5 \\ -5 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 5 \end{bmatrix}$	*Camel - Six Humps*	4.82	0.62
hartman3	3	$[0\ 0\ 0]'$	$[1\ 1\ 1]'$	*Hartman 3*	26.27	3.35
hartman6	6	$x_i \geq 0$	$x_i \leq 1$	*Hartman 6*	54.37	8.80
himmelblau	2	$\begin{bmatrix} -6 \\ -6 \end{bmatrix}$	$\begin{bmatrix} 6 \\ 6 \end{bmatrix}$	*Himmelblau*	7.40	0.90
rosenbrock8	8	$x_i \geq -30$	$x_i \leq 30$	*Rosenbrock 1, D = 8*	63.09	13.73
stepfunction2	4	$x_i \geq -100$	$x_i \leq 100$	*Step 2, D = 5*	11.72	1.81
styblinski-tang5	5	$x_i \geq -5$	$x_i \leq 5$	*Styblinski-Tang, n = 5*	37.02	6.10

Last two columns: average CPU time spent on each benchmark for solving the $N_{\text{test}} = 100$ instances analyzed in Fig. 9 by Bayesian optimization (BO) and GLIS (Algorithm 1)

As a reference target for assessing the quality of the optimization results, for each benchmark problem the optimization algorithm DIRECT [22] was used to compute the global optimum of the function through the NLopt interface [20], except for the ackley and stepfunction2 benchmarks in which PSO is used instead due to the slow convergence of DIRECT on those problems. The corresponding global minima were validated, when possible, against results reported in [19] or, in case of one- or two-dimensional problems, by inspection.

Algorithm 1 is run by using the RBF inverse quadratic function with hyperparameters obtained by dividing the values in (20) by the number n of variables, with the rationale that exploration is more difficult in higher dimensions and it is therefore better to rely more on the surrogate function during acquisition. The threshold $\epsilon_{\text{SVD}} = 10^{-6}$ is adopted to compute the RBF coefficients in (10c). The number of initial samples is $N_{\text{init}} = 2n$.

For each benchmark, the problem is solved $N_{\text{test}} = 100$ times to collect statistically significant enough results. The last two columns of Table 1 report the average CPU time spent for solving the $N_{\text{test}} = 100$ instances of each benchmark using BO and GLIS. As the benchmark functions are very easy to compute, the CPU time spending on evaluating the N_{max} function values F is negligible, so the time values reported in the table are practically those due to the execution of the algorithms. Algorithm 1 (GLIS) is between 4.6 and 9.4 times faster than Bayesian optimization (about 7 times faster on average). The execution time of GLIS in Python 3.7 on the same machine, using the PSO package pyswarm (https://pytho
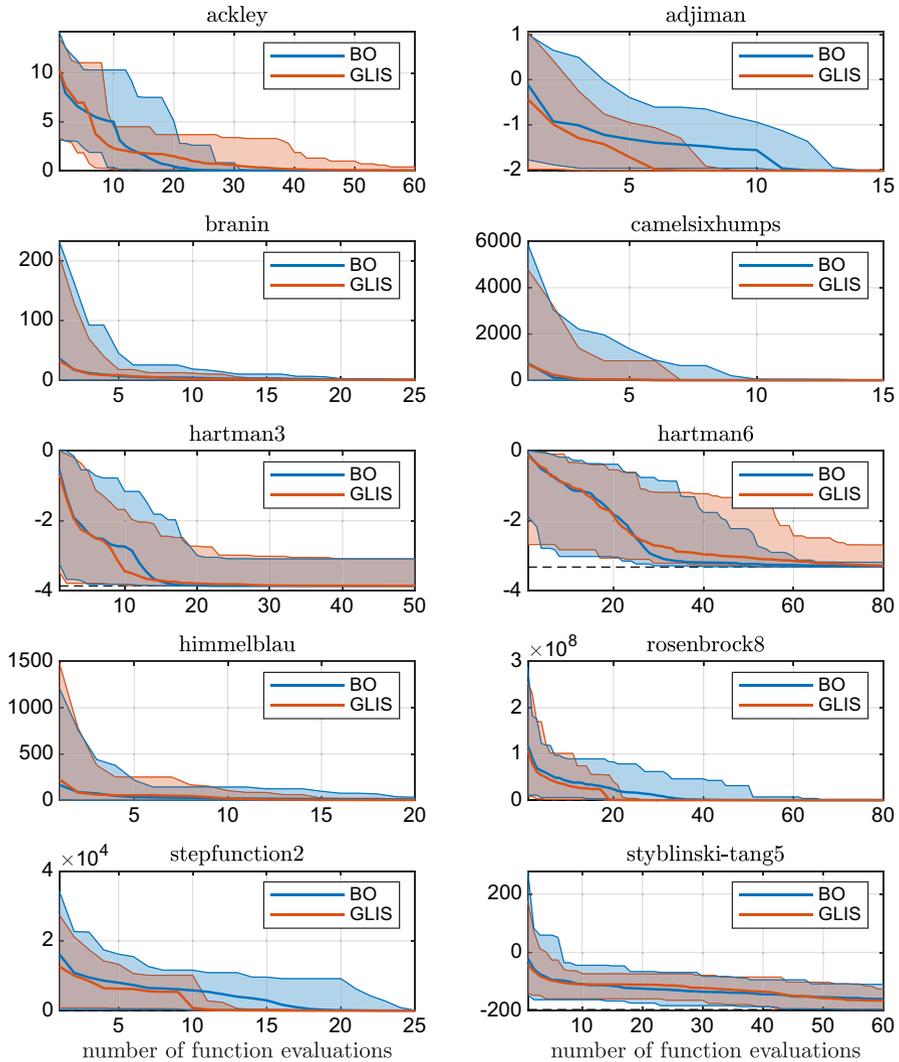
Fig. 9 Comparison between Algorithm 1 (GLIS) and Bayesian optimization (BO) on benchmark problems. Each plot reports the best function value obtained as a function of the number of function evaluations: average over $N_{test} = 100$ runs (thick lines), band defined by the best- and worst-case instances (shadowed areas), and global optimum to be attained (black dashed line)

nhosted.org/pyswarm) to optimize the acquisition function, is similar to that of the BO package `GPyOpt` [38].

The results of the tests are reported in Fig. 9, where in each plot we show the average function value obtained over $N_{test} = 100$ runs as a function of the number of function evaluations, and the band defined by the best-case and worst-case instances, and how the global optimum is approached.
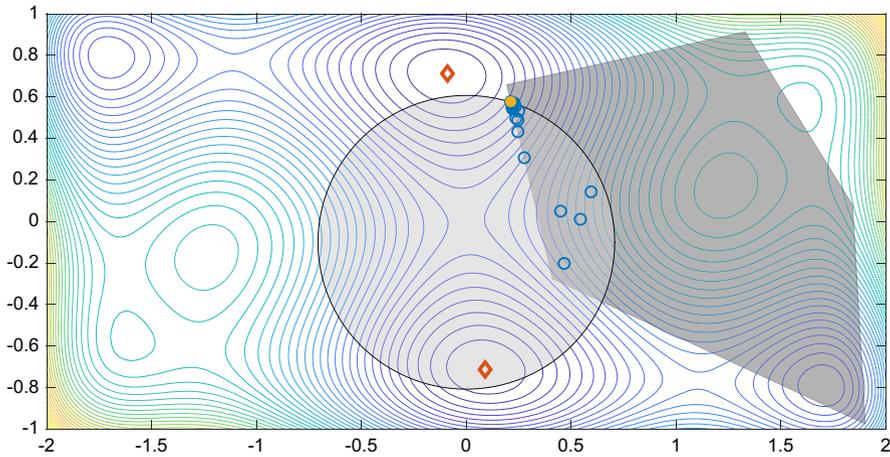
Fig. 10 Constrained camelsixhumps problem: constrained minimum (yellow dot) and unconstrained global minima (red diamonds) (Color figure online)

In order to test the algorithm in the presence of constraints, we consider the camelsixhumps problem and solve it under the following constraints

$$-2 \le x_1 \le 2, \quad -1 \le x_2 \le 1$$

$$\begin{bmatrix} 1.6295 & 1 \\ -1 & 4.4553 \\ -4.3023 & -1 \\ -5.6905 & -12.1374 \\ 17.6198 & 1 \end{bmatrix} x \le \begin{bmatrix} 3.0786 \\ 2.7417 \\ -1.4909 \\ 1 \\ 32.5198 \end{bmatrix}, \quad x_1^2 + (x_2 + 0.1)^2 \le 0.5$$

Algorithm 1 is run with hyperparameters set by dividing by $n = 2$ the values obtained in (20) and with $\epsilon_{\text{SVD}} = 10^{-6}$, $N_{\text{init}} = 2n$ for $N_{\text{max}} = 20$ iterations, with penalty $\rho = 1000$ in (18). The results are plotted in Fig. 10. The unconstrained two global minima of the function are located at $\begin{bmatrix} -0.0898 \\ 0.7126 \end{bmatrix}, \begin{bmatrix} 0.0898 \\ -0.7126 \end{bmatrix}$.

6.3 ADMM hyperparameter tuning for QP

The Alternating Direction Method of Multipliers (ADMM) [7] is a popular method for solving optimization problems such as the following convex Quadratic Program (QP)

$$\phi(\theta) = \min_z \frac{1}{2} z' Q z + (c + F\theta)' z \tag{21}$$
$$\text{s.t. } Az \le b + S\theta$$

where $z \in \mathbb{R}^n$ is the optimization vector, $\theta \in \mathbb{R}^p$ is a vector of parameters affecting the problem, and $A \in \mathbb{R}^{q \times n}$, $b \in \mathbb{R}^q$, $S \in \mathbb{R}^{q \times p}$, and we assume $Q = Q' \succ 0$.

Problems of the form (21) arise for example in model predictive control applications [2, 3], where z represents a sequence of future control inputs to optimize and θ collects signals that change continuously at runtime depending on measurements and set-point values. ADMM can be used effectively to solve QP problems (21), see for example the solver described in [1]. A very simple ADMM formulation for QP is summarized in Algorithm 3.

Algorithm 3 ADMM for QP

Input: Matrices Q, c, F, A, b, S, parameter θ, ADMM hyperparameters $\bar{\rho}, \bar{\alpha}$, number N of ADMM iterations.

---

1. $M_A \leftarrow (\frac{1}{\bar{\rho}}Q + A'A)^{-1}A'$; $m_\theta \leftarrow (\frac{1}{\bar{\rho}}Q + A'A)^{-1}(c + F\theta)$; $b_\theta \leftarrow b + S\theta$;
2. $s \leftarrow 0$, $u \leftarrow 0$;
3. **for** $i = 1, \ldots, N$ **do**:
   3.1. $z \leftarrow M_A(s - u) - m_\theta$;
   3.2. $w \leftarrow \bar{\alpha}Az + (1 - \bar{\alpha})s$;
   3.3. $s \leftarrow \min(w + u, b_\theta)$;
   3.4. $u \leftarrow u + w - s$;
4. **End**.

Output: Optimal solution $z^* = z$.

We consider a randomly generated QP test problem with $n = 5$, $q = 10$, $p = 3$ that is feasible for all $\theta \in [-1, 1]^3$, whose matrices are reported in "Appendix B" for reference. We set $N = 100$ in Algorithm 3, and generate $M = 2000$ samples θ_j uniformly distributed in $[-1, 1]^3$. The aim is to find the hyperparameters $x = [\bar{\rho} \ \bar{\alpha}]'$ that provide the best QP solution quality after N ADMM iterations. Quality is expressed by the following objective function

$$
f(x) = \log\left(\frac{1}{M} \sum_{j=1}^{M} \max\left\{ \frac{\phi_j^*(x) - \phi^*(x)}{1 + |\phi^*(x)|}, 0 \right\} \right.
$$
$$
\left. + \bar{\beta} \max\left\{ \max_i \left\{ \frac{A_i z_j^*(x) - b_i - S_i x}{1 + |b_i + S_i x|} \right\}, 0 \right\} \right) \tag{22}
$$

where $\phi_j^*(x)$, $z_j^*(x)$ are the optimal value and optimizer found at run #j, respectively, $\phi^*(x)$ is the solution of the QP problem obtained by running the very fast and accurate ODYS QP solver [10]. The first term in (22) measures relative optimality, the second term relative violation of the constraints, and we set $\bar{\beta} = 1$ to equally weight relative optimality versus relative accuracy. Function f in (22) is minimized for $\ell = \begin{bmatrix} 0.01 \\ 0.01 \end{bmatrix}$ and $u = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$ using GLIS with the same parameters used in Sect. 6.2 and, for comparison, by Bayesian optimization. Due to the fact that ADMM provides suboptimal solutions, when acquiring the samples f_i the argument of the logarithm in (22) is always found positive in our tests. The test is repeated $N_{\text{test}} = 100$ times and the results are depicted in Fig. 11. It is apparent that GLIS attains slightly better function values for the same number of functions evaluations than BO, both on
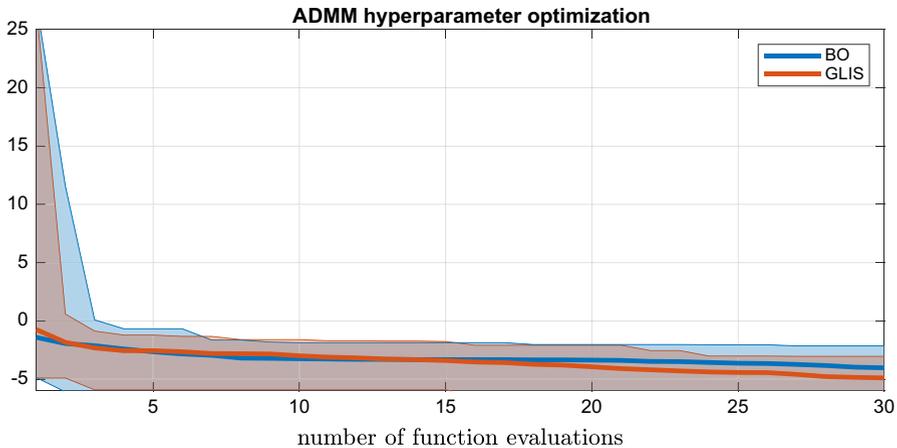
Fig. 11 Hyperparameter optimization for ADMM. Average performance over $N_{\text{test}} = 100$ runs and best/worst-case instance bands

average and in the worst-case. The resulting hyperparameter tuning that minimized the selected ADMM performance index (22) is $\bar{\rho} = 0.1566, \bar{\alpha} = 1.9498$.

7 Conclusions

This paper has proposed an approach based on surrogate functions to address global optimization problems whose objective function is expensive to evaluate, possibly under constraints that are inexpensive to evaluate. Contrarily to Bayesian optimization methods, the approach is driven by deterministic arguments based on radial basis functions (or inverse distance weighting) to create the surrogate, and on inverse distance weighting to characterize the uncertainty between the surrogate and the black-box function to optimize, as well as to promote the exploration of the feasible space. The computational burden associated with the algorithm is lighter then the one of Bayesian optimization while performance is comparable. Clearly, the main limitation of the algorithm is related to the dimension of the optimization vector it can cope with, as many other black-box global optimization algorithms.

Current research is devoted to extend the approach to include constraints that are also expensive to evaluate, and to explore if performance can be improved by adapting the parameters α and δ during the search. Future research should address theoretical issues of convergence of the approach, by investigating assumptions on the black-box function f and on the parameters $\alpha, \delta, \epsilon_{\text{SVD}}, \epsilon_{\Delta F} > 0$ of the algorithm, so to allow guaranteeing convergence, for example using the arguments in [15] based on the results in [40].

Appendices

A Proofs

Proof of Lemma 1 Property P1 easily follows from (4). Property P2 also easily follows from the fact that for all $x \in \mathbb{R}^n$ the values $v_i(x) \in [0, 1]$, $\forall i = 1, \dots, N$, and $\sum_{i=1}^{N} v_i(x) = 1$, so that

$$\min_{j}\{f_j\} = \sum_{i=1}^{N} v_i(x) \min_{j}\{f_j\} \le \sum_{i=1}^{N} v_i(x) f_i = \hat{f}(x) \le \sum_{i=1}^{N} v_i(x) \max_{j}\{f_j\} = \max_{j}\{f_j\}$$

Regarding differentiability, we first prove that for all $i = 1, \dots, N$, functions v_i are differentiable everywhere on $\mathbb{R}^n$, and that in particular $\nabla v_i(x_j) = 0$ for all $j = 1, \dots, N$. Clearly, functions v_i are differentiable for all $x \notin \{x_1, \dots, x_N\}$, $\forall i = 1, \dots, N$. Let e^h be the hth column of the identity matrix of order n. Consider first the case in which $w_i(x)$ are given by (3b). The partial derivatives of v_i at x_i are

$$\begin{aligned}\frac{\partial v_i(x_i)}{\partial x^h} &= \lim_{t \to 0} \frac{1}{t}\left(\frac{w_i(x_i + te^h)}{\sum_{j=1}^{N} w_j(x_i + te^h)} - 1\right) = \lim_{t \to 0} -\frac{\sum_{j \ne i} w_j(x_i + te^h)}{t \sum_{j=1}^{N} w_j(x_i + te^h)} \\ &= \lim_{t \to 0} -\frac{\sum_{j \ne i} w_j(x_i)}{t \frac{e^{-\|x_i + te^h - x_i\|^2}}{\|x_i + te^h - x_i\|^2}} = \lim_{t \to 0} -t \frac{\sum_{j \ne i} w_j(x_i)}{e^{-t^2}} = 0\end{aligned}$$

and similarly at x_j, $j \ne i$ are

$$\frac{\partial v_i(x_j)}{\partial x^h} = \lim_{t \to 0} \frac{1}{t}\left(\frac{w_i(x_j + te^h)}{\sum_{k=1}^{N} w_k(x_j + te^h)} - 0\right) = \lim_{t \to 0} \frac{1}{t} \frac{w_i(x_j)}{\frac{e^{-t^2}}{t^2} + \sum_{k \ne j} w_k(x_j)} = 0$$

In case $w_i(x)$ given by (3a) differentiability follows similarly, with e^{-t^2} replaced by 1. Therefore $\hat{f}$ is differentiable and

$$\nabla \hat{f}(x_j) = \sum_{i=1}^{N} f_i \nabla v_i(x_j) = 0$$

$\square$

Proof of Lemma 2 As by Lemma 1 functions $\hat{f}$ and v_i are differentiable, $\forall i = 1, \ldots, N$, it follows immediately that $s(x)$ is differentiable. Regarding differentiability of z, clearly it is differentiable for all $x \notin \{x_1, \ldots, x_N\}$, $\forall i = 1, \ldots, N$. Let e^h be the hth column of the identity matrix of order n. Consider first the case in which $w_i(x)$ are given by (3b). The partial derivatives of z at x_i are

$$
\frac{\partial z(x_i)}{\partial x^h} = \lim_{t \to 0} \frac{1}{t} \left(\frac{2}{\pi} \tan^{-1} \left(\frac{1}{\sum_{j=1}^{N} w_j(x_i + te^h)} \right) - 0 \right)
$$

$$
= \lim_{t \to 0} \frac{2}{\pi t} \tan^{-1} \left(\frac{1}{\frac{e^{-t^2}}{t^2} + \sum_{j \neq i} w_j(x_i)} \right) = \lim_{t \to 0} \frac{2}{\pi t} \tan^{-1} \left(\frac{t^2}{1 + \sum_{j \neq i} w_j(x_i)} \right) = 0
$$

In case $w_i(x)$ are given by (3a) differentiability follows similarly, with e^{-t^2} replaced by 1. Therefore the acquisition function a is differentiable for all $\alpha, \delta \geq 0$. $\square$

B Matrices of parametric QP considered in Sect. 6.3

$$
Q = \begin{bmatrix}
6.6067 & -1.6361 & 2.8198 & 0.3776 & 3.1448 \\
-1.6361 & 0.9943 & -0.9998 & -0.4786 & -0.5198 \\
2.8198 & -0.9998 & 4.0749 & 0.2183 & 0.2714 \\
0.3776 & -0.4786 & 0.2183 & 0.7310 & 0.1689 \\
3.1448 & -0.5198 & 0.2714 & 0.1689 & 2.1716
\end{bmatrix}
\quad
c = \begin{bmatrix}
-11.4795 \\
1.0487 \\
7.2225 \\
25.8549 \\
-6.6689
\end{bmatrix}
$$

$$
A = \begin{bmatrix}
-0.8637 & -1.0891 & -0.6156 & 1.4193 & -1.0000 \\
0.0774 & 0.0326 & 0.7481 & 0.2916 & -1.0000 \\
-1.2141 & 0.5525 & -0.1924 & 0.1978 & -1.0000 \\
-1.1135 & 1.1006 & 0.8886 & 1.5877 & -1.0000 \\
-0.0068 & 1.5442 & -0.7648 & -0.8045 & -1.0000 \\
1.5326 & 0.0859 & -1.4023 & 0.6966 & -1.0000 \\
-0.7697 & -1.4916 & -1.4224 & 0.8351 & -1.0000 \\
0.3714 & -0.7423 & 0.4882 & -0.2437 & -1.0000 \\
-0.2256 & -1.0616 & -0.1774 & 0.2157 & -1.0000 \\
1.1174 & 2.3505 & -0.1961 & -1.1658 & -1.0000
\end{bmatrix}
\quad
b = \begin{bmatrix}
0.0838 \\
0.2290 \\
0.9133 \\
0.1524 \\
0.8258 \\
0.5383 \\
0.9961 \\
0.0782 \\
0.4427 \\
0.1067
\end{bmatrix}
$$

$$
F = \begin{bmatrix}
1.8733 & 8.4038 & -6.0033 \\
-0.8249 & -8.8803 & 4.8997 \\
-19.3302 & 1.0009 & 7.3936 \\
-4.3897 & -5.4453 & 17.1189 \\
-17.9468 & 3.0352 & -1.9412
\end{bmatrix}
\quad
S = \begin{bmatrix}
2.9080 & -0.3538 & 0.0229 \\
0.8252 & -0.8236 & -0.2620 \\
1.3790 & -1.5771 & -1.7502 \\
-1.0582 & 0.5080 & -0.2857 \\
-0.4686 & 0.2820 & -0.8314 \\
-0.2725 & 0.0335 & -0.9792 \\
1.0984 & -1.3337 & -1.1564 \\
-0.2779 & 1.1275 & -0.5336 \\
0.7015 & 0.3502 & -2.0026 \\
-2.0518 & -0.2991 & 0.9642
\end{bmatrix}
$$

References

1. Banjac, G., Stellato, B., Moehle, N., Goulart, P., Bemporad, A., Boyd, S.: Embedded code generation using the OSQP solver. In: *Proc. 56th IEEE Conf. on Decision and Control*, pp. 1906–1911, Melbourne, Australia, 2017. https://github.com/oxfordcontrol/osqp

2. Bemporad, A.: Model-based predictive control design: new trends and tools. In: *Proc. 45th IEEE Conf. on Decision and Control*, pp. 6678–6683, San Diego, CA (2006)

3. Bemporad, A.: A multiparametric quadratic programming algorithm with polyhedral computations based on nonnegative least squares. IEEE Trans. Autom. Control **60**(11), 2892–2903 (2015)

4. Bemporad, A.: Global optimization via inverse distance weighting. 2019. Available on arXiv at arxiv:1906.06498. Code available at http://cse.lab.imtlucca.it/~bemporad/glis

5. Bemporad, A., Piga, D.: Active preference learning based on radial basis functions. 2019. Available on arXiv at arxiv:1909.13049. Code available at http://cse.lab.imtlucca.it/~bemporad/idwgopt

6. Blok, H.J.: The lhsdesigncon MATLAB function, 2014. https://github.com/rikblok/matlab-lhsdesigncon

7. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)

8. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York, NY, USA (2004). http://www.stanford.edu/~boyd/cvxbook.html

9. Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint* arXiv:1012.2599 (2010)

10. Cimini, G., Bemporad, A., Bernardini, D.: ODYS QP Solver. ODYS S.r.l. (https://odys.it/qp), September 2017

11. Costa, A., Nannicini, G.: Rbfopt: an open-source library for black-box optimization with costly function evaluations. Math. Program. Comput. **10**(4), 597–629 (2018)

12. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43. Nagoya (1995)

13. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F.: Auto-sklearn: efficient and robust automated machine learning. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Automated Machine Learning: Methods, Systems, Challenges, pp. 113–134. Springer International Publishing (2019)

14. Forgione, M., Piga, D., Bemporad, A.: Efficient calibration of embedded MPC. In: Proc. 21th IFAC World Congress. https://arxiv.org/abs/1911.13021 (2020)

15. Gutmann, H.-M.: A radial basis function method for global optimization. J. Glob. Optim. **19**, 201–2227 (2001)

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

17. Hardy, R.L.: Multiquadric equations of topography and other irregular surfaces. J. Geophys. Res. **76**(8), 1905–1915 (1971)

18. Huyer, W., Neumaier, A.: Global optimization by multilevel coordinate search. J. Glob. Optim. **14**(4), 331–355 (1999)

19. Jamil, M., Yang, X.-S.: A literature survey of benchmark functions for global optimisation problems. Int. J. Math. Model. Numer. Optim. **4**(2):150–194 (2013). arxiv:1308.4008.pdf

20. Johnson, S.G.: The NLopt nonlinear-optimization package. http://github.com/stevengj/nlopt

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

22. Jones, D.R.: DIRECT global optimization algorithm. *Encyclopedia of Optimization*, pages 725–735, (2009)

23. Jones, D.R., Schonlau, M., Matthias, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998)

24. Joseph, V.R., Kang, L.: Regression-based inverse distance weighting with applications to computer experiments. Technometrics **53**(3), 255–265 (2011)

25. Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. J. Basic Eng. **86**(1), 97–106 (1964)

26. Matheron, G.: Principles of geostatistics. Econ. Geol. **58**(8), 1246–1266 (1963)

27. McDonald, D.B., Grantham, W.J., Tabor, W.L., Murphy, M.J.: Global and local optimization using radial basis function response surface models. Appl. Math. Model. **31**(10), 2095–2110 (2007)
28. McKay, M.D., Beckman, R.J., Conover, W.J.: Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979)
29. Piga, D., Forgione, M., Formentin, S., Bemporad, A.: Performance-oriented model learning for data-driven MPC design. IEEE Control Systems Letters, 2019. Also in Proc. 58th IEEE Conf. Decision and Control, Nice (France) (2019). arxiv:1904.10839
30. Powell, M.J.D.: The NEWUOA software for unconstrained optimization without derivatives. In Large-scale nonlinear optimization, pp. 255–297. Springer (2006)
31. Regis, R.G., Shoemaker, C.A.: Constrained global optimization of expensive black box functions using radial basis functions. J. Glob. Optim. **31**(1), 153–171 (2005)
32. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Glob. Optim. **56**(3), 1247–1293 (2013)
33. Rippa, S.: An algorithm for selecting a good value for the parameter c in radial basis function interpolation. Adv. Comput. Math. **11**(2–3), 193–210 (1999)
34. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. 409–423 (1989)
35. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., De Freitas, N.: Taking the human out of the loop: a review of Bayesian optimization. Proc. IEEE **104**(1), 148–175 (2015)
36. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proc. ACM National Conference, pp. 517–524. New York (1968)
37. Snoek, J., Jasper, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, pp. 2951–2959 (2012)
38. The GPyOpt authors. GPyOpt: a Bayesian optimization framework in Python. http://github.com/SheffieldML/GPyOpt (2016)
39. The Mathworks, Inc. Statistics and Machine Learning Toolbox User's Guide (2019). https://www.mathworks.com/help/releases/R2019a/pdf_doc/stats/stats.pdf
40. Törn, A., Žilinskas, A.: Global Optimization, vol. 350. Springer, Berlin (1989)
41. Vaz, A.I.F., Vicente, L.N.: A particle swarm pattern search method for bound constrained global optimization. J. Glob. Optim. **39**(2), 197–219 (2007)
42. Vaz, A.I.F., Vicente, L.N.: PSwarm: a hybrid solver for linearly constrained global derivative-free optimization. Optim. Methods Softw. **24**, 669–685 (2009). http://www.norg.uminho.pt/aivaz/pswarm/
43. Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**(2), 65–85 (1994)