Computational Optimization and Applications (2023) 86:383–420
https://doi.org/10.1007/s10589-023-00491-2

GLISp-r: a preference-based optimization algorithm with convergence guarantees

Davide Previtali[1] · Mirko Mazzoleni[1] · Antonio Ferramosca[1] · Fabio Previdi[1]

Received: 29 November 2022 / Accepted: 26 April 2023 / Published online: 12 May 2023

Abstract

Preference-based optimization algorithms are iterative procedures that seek the optimal calibration of a decision vector based only on comparisons between couples of different tunings. At each iteration, a human decision-maker expresses a preference between two calibrations (samples), highlighting which one, if any, is better than the other. The optimization procedure must use the observed preferences to find the tuning of the decision vector that is most preferred by the decision-maker, while also minimizing the number of comparisons. In this work, we formulate the preference-based optimization problem from a utility theory perspective. Then, we propose GLISp-r, an extension of a recent preference-based optimization procedure called GLISp. The latter uses a Radial Basis Function surrogate to describe the tastes of the decision-maker. Iteratively, GLISp proposes new samples to compare with the best calibration available by trading off exploitation of the surrogate model and exploration of the decision space. In GLISp-r, we propose a different criterion to use when looking for new candidate samples that is inspired by MSRS, a popular procedure in the black-box optimization framework. Compared to GLISp, GLISp-r is less likely to get stuck on local optima of the preference-based optimization problem. We motivate this claim theoretically, with a proof of global convergence, and empirically, by comparing the performances of GLISp and GLISp-r on several benchmark optimization problems.

✉ Davide Previtali
davide.previtali@unibg.it

Mirko Mazzoleni
mirko.mazzoleni@unibg.it

Antonio Ferramosca
antonio.ferramosca@unibg.it

Fabio Previdi
fabio.previdi@unibg.it

[1] Department of Management, Information and Production Engineering, University of Bergamo, Via G. Marconi 5, 24044 Dalmine, BG, Italy

Keywords Global optimization · Preference-based optimization · Surrogate-based methods · Active preference learning · Utility theory

1 Introduction

Preference-Based Optimization (PBO) algorithms seek the global solution of an optimization problem whose objective function is unknown and unmeasurable. Instead, the "goodness" of a *decision vector* is assessed by a human *Decision-Maker (DM)*, who compares different samples (i.e. tunings of the decision vector) and states which of them he/she prefers. In this work, we consider *queries* to the DM that involve two options. The output of a query is the preference between the two calibrations of the decision vector expressed by the decision-maker (e.g. "this tuning is better than that one") [3].

Preference-based optimization is closely related to industry practice. In the context of control systems, often the performances achieved by a regulator tuning are evaluated by a decision-maker, who expresses his/her judgment by observing the behavior of the system under control. Multiple experiments must be performed until the DM is satisfied by the closed-loop performances. If a trial-and-error approach is adopted, then there is no guarantee that the final tuning is optimal in some sense. Moreover, the calibration process might be quite time-consuming since possibly many combinations of the controller parameters need to be tested. PBO algorithms constitute a better alternative to the trial-and-error methodology due to the fact that they drive the experiments by exploiting the information carried by the preferences expressed by the DM. The goal is to *seek the calibration of the decision vector that is most preferred by the decision-maker while also minimizing the number of queries*, thus performing fewer experiments. Successful applications of preference-based optimization procedures include [39], where the authors use algorithm `GLISp` [3] to tune the controller for a continuous stirring tank reactor and for autonomous driving vehicles. The same algorithm has been employed in [30] to calibrate the parameters of a velocity planner for robotic sealing tasks.

Although preference-based optimization is a valuable tool for solving those calibration tasks that involve human decision-makers, its applicability is broader. In multi-objective optimization, PBO can be employed to scalarize the multi-objective optimization problem into a single objective one by tuning the weights for the weighted sum method, or to choose which, among a set of Pareto optimal solutions, is the most suited for the DM [3, 21]. PBO can also be used to perform active preference learning. In general, preference learning methods estimate a predictive model that describes the tastes of a human decision-maker [12]. Preference-based optimization methods also rely on a predictive model (called surrogate model), which is updated after each query to the DM, but its prediction accuracy is not the main concern. Rather, the sole purpose of the surrogate model is to drive the search towards the most preferred tuning of the decision vector.

The preference-based optimization problem can be formalized by applying notions of *utility theory* [27]. Suppose that there exists a binary relation, called the *preference relation*, which describes the tastes of the decision-maker (i.e. the outputs of the

queries). If the preference relation of the DM exhibits certain properties, then it is possible to represent it with a continuous (latent) *utility function*[1] [8], which assigns an abstract degree of "goodness" to all possible calibrations of the decision vector. The tuning that is most preferred by the decision-maker is the one with the highest utility and corresponds to the optimizer of the PBO problem.

As previously mentioned, many preference-based optimization algorithms rely on a *surrogate model*, which is an approximation of the latent utility function built from the preference information at hand. In general, any procedure that uses a surrogate model when solving an optimization problem is said to be a *surrogate-based* (or *response surface*) *method*. These algorithms are mostly used for black-box optimization problems, where the objective function is unknown but measurable (see [19, 36]). Nevertheless, surrogate-based methods can also be employed for PBO problems, provided that the surrogate model is properly defined. In practice, preference-based response surface methods iteratively propose new samples to be compared to the available best candidate by trading off *exploitation* (or local search), i.e. selecting samples that are likely to offer an improvement based on the data at our disposal, and *exploration* (or global search), i.e. finding samples in regions of the decision space of which we have little to no knowledge. Typically, new candidate samples are sought by minimizing or maximizing an *acquisition function* that encapsulates these two aspects.

Most surrogate models either rely on Gaussian Processes ($\mathcal{GP}$s) [38], giving rise to preferential Bayesian optimization [13], or Radial Basis Functions (RBFs) [15]. For example, in [7] the authors propose a predictive model for the latent utility function that is based on $\mathcal{GP}$s. The latter is used as a surrogate model in [6] to carry out preference-based optimization. Alternative preferential Bayesian optimization algorithms are proposed in [4] and in [13]. Recently, in [3] the authors developed a preference-based optimization method, called `GLISp`, which is based on a RBF approximation of the latent utility function.

In this work, *we propose an extension of the* `GLISp` [3] *algorithm (that we will refer to as* `GLISp-r`*) which is more robust in finding the global solution of the preference-based optimization problem*. To do so, we:

1. Address some limitations of the acquisition function used in [3], which can cause the procedure to miss the global solution and get stuck on local optima;
2. Dynamically vary the exploration–exploitation trade-off weight, allowing `GLISp-r` to alternate between local and global search. This is commonly done in the black-box optimization framework, see for example [15, 28, 29, 37], but it has not been tried for `GLISp` [3];
3. Provide a *proof of global convergence* for `GLISp-r`, furtherly motivating its robustness. Currently, no such proof is available for `GLISp` [3] (or for any of the most popular preference-based response surface methods, cf. also [4, 6, 13]).

Regarding this last point, as a minor contribution of this work, we formalize the preference-based optimization problem from a utility theory perspective, allowing us to analyze the existence of a solution and, ultimately, to prove the global convergence of `GLISp-r`.

[1] As a matter of fact, the goal of some preference learning methods is to estimate the utility function of the decision-maker [12].

This paper is organized as follows. In Sect. 2, we introduce the preference-based optimization problem. Section 3 addresses how to build the surrogate model (as in GLISp [3]) and look for the next sample to evaluate, keeping in mind the exploration–exploitation dilemma. We also briefly cover the exploration function used in [3]. The latter is thoroughly analyzed in Sect. 4, where we propose a solution to the shortcomings that we have encountered in GLISp [3]. Section 5 describes algorithm GLISp-r and addresses its convergence. Then, Sect. 6 compares the performances of GLISp-r with GLISp [3] and C-GLISp [40], a revisitation of the latter method by the same authors, on several benchmark optimization problems. Finally, Sect. 7 gives some concluding remarks.

2 Problem formulation

Let $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} = \left[x^{(1)} \ \dots \ x^{(n)}\right]^\top$, be the n-dimensional decision vector and suppose that we are interested in finding its calibration that is most preferred by the decision-maker within a subset of $\mathbb{R}^n$, namely $\Omega \subset \mathbb{R}^n$. In particular, we define the *constraint set* Ω as:

$$\Omega = \left\{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \boldsymbol{g_{ineq}}(\boldsymbol{x}) \leq \boldsymbol{0}_{q_{ineq}}, \boldsymbol{g_{eq}}(\boldsymbol{x}) = \boldsymbol{0}_{q_{eq}}\right\}. \tag{1}$$

In (1), $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^n, \boldsymbol{l} \leq \boldsymbol{u}$, are the lower and upper bounds on the decision vector while $\boldsymbol{g_{ineq}} : \mathbb{R}^n \to \mathbb{R}^{q_{ineq}}$ and $\boldsymbol{g_{eq}} : \mathbb{R}^n \to \mathbb{R}^{q_{eq}}$ are the constraints functions associated to the inequality and equality constraints respectively (which are $q_{ineq} \in \mathbb{N} \cup \{0\}$ and $q_{eq} \in \mathbb{N} \cup \{0\}$). Notation-wise, $\boldsymbol{0}_{q_{ineq}}$ represents the q_{ineq}-dimensional zero column vector (and similarly for $\boldsymbol{0}_{q_{eq}}$). We suppose that: (i) all of the constraints in (1) are completely known and (ii) Ω includes, at least, the bound constraints $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$ (the remaining equality and inequality constraints can be omitted, resulting in $q_{ineq} = q_{eq} = 0$).

In this work, we formalize the preference-based optimization problem from a *utility theory* [27] perspective. We start by introducing preference relations, a particular kind of binary relations that play a key role in PBO.

Definition 1 (*Binary relation* [27]) Consider the constraint set Ω in (1); we define a generic binary relation $\mathcal{R}$ on Ω as a subset $\mathcal{R} \subseteq \Omega \times \Omega$.

Notation-wise, given two samples $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$, we denote the ordered pairs for which the binary relation holds, $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{R}$, as $\boldsymbol{x}_i \mathcal{R} \boldsymbol{x}_j$.

Definition 2 (*Preference relation* [27]) A preference relation, $\succsim \subseteq \Omega \times \Omega$, is a binary relation that describes the tastes of a human decision-maker.

In the context of utility theory, $\boldsymbol{x}_i \succsim \boldsymbol{x}_j$ implies that the DM with preference relation $\succsim$ on Ω deems the alternative $\boldsymbol{x}_i$ at least as good as $\boldsymbol{x}_j$. We say that the decision-maker is rational (in an economics sense) if his/her preference relation exhibits certain properties, as highlighted by the following Definition.

Definition 3 (*Rational decision-maker* [27]) Consider a decision-maker with preference relation $\succsim$ on Ω. We say that the DM is rational if $\succsim$ is a reflexive, transitive and complete binary relation on Ω.

Now, we briefly review each of the aforementioned properties and give some insights as to why they characterize the rationality of an individual (see [8, 10, 27] for more details):

- *Reflexivity* of $\succsim$ on Ω implies that, for the DM, any alternative is as good as itself, i.e. $\boldsymbol{x}_i \succsim \boldsymbol{x}_i$ for each $\boldsymbol{x}_i \in \Omega$;
- A decision-maker whose preference relation $\succsim$ on Ω is *transitive* is able to express his/her preferences coherently since if $\boldsymbol{x}_i \succsim \boldsymbol{x}_j$ and $\boldsymbol{x}_j \succsim \boldsymbol{x}_k$ hold, then $\boldsymbol{x}_i \succsim \boldsymbol{x}_k$, for any $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \in \Omega$;
- *Completeness* of $\succsim$ on Ω implies that the DM is able to express a preference between any two alternatives in Ω, i.e. either $\boldsymbol{x}_i \succsim \boldsymbol{x}_j$ or $\boldsymbol{x}_j \succsim \boldsymbol{x}_i$ holds for each $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$.

The preference relation $\succsim$ on Ω of a rational decision-maker is usually "split" into two transitive binary relations [27]:

- The *strict preference relation* $\succ$ on Ω, i.e. $\boldsymbol{x}_i \succ \boldsymbol{x}_j$ if and only if $\boldsymbol{x}_i \succsim \boldsymbol{x}_j$ but not $\boldsymbol{x}_j \succsim \boldsymbol{x}_i$ ($\boldsymbol{x}_i$ is "better than" $\boldsymbol{x}_j$ or, equivalently, $\boldsymbol{x}_j$ is "worse than" $\boldsymbol{x}_i$), and
- The *indifference relation* $\sim$ on Ω, i.e. $\boldsymbol{x}_i \sim \boldsymbol{x}_j$ if and only if $\boldsymbol{x}_i \succsim \boldsymbol{x}_j$ and $\boldsymbol{x}_j \succsim \boldsymbol{x}_i$ ($\boldsymbol{x}_i$ is "as good as" $\boldsymbol{x}_j$).

One last relevant property for preference relations is continuity.

Definition 4 (*Continuous preference relation* [27]) A preference relation $\succsim$ on Ω is continuous if the strict upper and lower $\succsim$-contour sets:

$$\begin{aligned} \mathcal{U}_{\succ}(\boldsymbol{x}) &= \{\tilde{\boldsymbol{x}} : \tilde{\boldsymbol{x}} \in \Omega, \tilde{\boldsymbol{x}} \succ \boldsymbol{x}\} \quad \text{and} \\ \mathcal{L}_{\succ}(\boldsymbol{x}) &= \{\tilde{\boldsymbol{x}} : \tilde{\boldsymbol{x}} \in \Omega, \boldsymbol{x} \succ \tilde{\boldsymbol{x}}\} \quad \text{respectively,} \end{aligned}$$

are open subsets of Ω for each $\boldsymbol{x} \in \Omega$.

Intuitively speaking, if $\succsim$ on Ω is continuous and $\boldsymbol{x}_i \succ \boldsymbol{x}_j$ holds, then an alternative $\boldsymbol{x}_k$ which is "very close" to $\boldsymbol{x}_j$ should also be deemed strictly worse than $\boldsymbol{x}_i$ by the decision-maker, i.e. $\boldsymbol{x}_i \succ \boldsymbol{x}_k$.

Having defined the preference relation $\succsim$ on Ω thoroughly, we can finally state the goal of preference-based optimization:

$$find\ \boldsymbol{x}^* \in \Omega\ such\ that\ \boldsymbol{x}^* \succsim \boldsymbol{x}, \forall \boldsymbol{x} \in \Omega. \tag{2}$$

Formally, $\boldsymbol{x}^*$ is called the $\succsim$-*maximum of* Ω [27], i.e. the sample that is most preferred by the decision-maker with preference relation $\succsim$ on Ω. Concerning the existence of $\boldsymbol{x}^*$, we can state the following Proposition, which can be seen as a generalization of the Extreme Value Theorem [1] for preference relations.

Proposition 1 (Existence of a $\succsim$-maximum of Ω [27]) *A $\succsim$-maximum of Ω is guaranteed to exist if Ω is a compact subset of a metric space (in our case $\Omega \subset \mathbb{R}^n$) and $\succsim$ is a continuous preference relation on Ω of a rational decision-maker (see Definitions 3 and 4).*

Proposition 1 will be relevant when proving the convergence of the proposed algorithm, in Sect. 5. Lastly, in order to write Problem (2) as a typical global optimization problem, we need to state one of the most important results in utility theory.

Theorem 1 (Debreu's utility representation Theorem for $\mathbb{R}^n$ [8]) *Let Ω be any nonempty subset of $\mathbb{R}^n$ and $\succsim$ be a preference relation on Ω of a rational decision-maker (as in Definition 3). If $\succsim$ on Ω is continuous, then it can be represented by a continuous utility function $u_{\succsim} : \Omega \to \mathbb{R}$ such that, for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in \Omega$:*

$$\begin{aligned} \boldsymbol{x}_i \succsim \boldsymbol{x}_j \quad &\text{if and only if} \quad u_{\succsim}(\boldsymbol{x}_i) \geq u_{\succsim}\left(\boldsymbol{x}_j\right), \\ \boldsymbol{x}_i \succ \boldsymbol{x}_j \quad &\text{if and only if} \quad u_{\succsim}(\boldsymbol{x}_i) > u_{\succsim}\left(\boldsymbol{x}_j\right), \\ \boldsymbol{x}_i \sim \boldsymbol{x}_j \quad &\text{if and only if} \quad u_{\succsim}(\boldsymbol{x}_i) = u_{\succsim}\left(\boldsymbol{x}_j\right). \end{aligned}$$

Using Theorem 1, we can build an optimization problem to find the $\succsim$-maximum of Ω. In particular, we define the *scoring function*, $f : \mathbb{R}^n \to \mathbb{R}$, as $f(\boldsymbol{x}) = -u_{\succsim}(\boldsymbol{x})$ and re-write Problem (2) as:

$$\begin{aligned} \boldsymbol{x}^* &= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ \text{s.t.} \quad &\boldsymbol{x} \in \Omega. \end{aligned} \tag{3}$$

Remark 1 Formally, there could be more than one $\succsim$-maximum of Ω, i.e. Problem (3) could admit multiple global solutions, as described by the set:

$$\mathcal{X}^* = \left\{\boldsymbol{x}_i^* : \boldsymbol{x}_i^* \in \Omega \text{ such that } \nexists \boldsymbol{x} : \boldsymbol{x} \succ \boldsymbol{x}_i^*\right\}.$$

In this work, without loss of generality, we assume that there exists only one global solution $\boldsymbol{x}^*$ as in (3). In practice, as we will see in Sect. 5, any globally convergent preference-based optimization procedure generates a set of samples that is dense in Ω and thus, at least asymptotically, it actually finds all the global minimizers in $\mathcal{X}^*$. We do not make any assumptions on the local solutions of (3), which can be more than one.

On a side note, we could view preference-based optimization as a particular instance of black-box optimization [19, 36] where the cost function is not measurable in any way. Instead, information on $f(\boldsymbol{x})$ in (3) comes in the form of preferences, as described in the next section.

2.1 Data available to preference-based optimization procedures

In `GLISp` [3], instead of considering the preference relation $\succsim$ on Ω explicitly, the authors describe the outputs of the queries to the DM using the *preference function*

$\pi_{\succsim} : \mathbb{R}^n \times \mathbb{R}^n \to \{-1, 0, 1\}$, defined as:

$$\pi_{\succsim}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \begin{cases} -1 & \text{if } \boldsymbol{x}_i \succ \boldsymbol{x}_j \Leftrightarrow f\left(\boldsymbol{x}_i\right) < f\left(\boldsymbol{x}_j\right) \\ 0 & \text{if } \boldsymbol{x}_i \sim \boldsymbol{x}_j \Leftrightarrow f\left(\boldsymbol{x}_i\right) = f\left(\boldsymbol{x}_j\right) \\ 1 & \text{if } \boldsymbol{x}_j \succ \boldsymbol{x}_i \Leftrightarrow f\left(\boldsymbol{x}_i\right) > f\left(\boldsymbol{x}_j\right) \end{cases}. \tag{4}$$

In light of the just reviewed utility theory literature, we can see that $\pi_{\succsim}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in (4) is obtained from the utility representation of the preference relation $\succsim$ on Ω (see Theorem 1) and from the fact that $f(\boldsymbol{x}) = -u_{\succsim}(\boldsymbol{x})$. In the case of rational decision-makers (Definition 3), reflexivity and transitivity of the preference relation are highlighted by the following properties of the preference function:

- $\pi_{\succsim}(\boldsymbol{x}_i, \boldsymbol{x}_i) = 0$, for each $\boldsymbol{x}_i \in \mathbb{R}^n$,
- $\pi_{\succsim}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \pi_{\succsim}(\boldsymbol{x}_j, \boldsymbol{x}_k) = b \Rightarrow \pi_{\succsim}(\boldsymbol{x}_i, \boldsymbol{x}_k) = b$, for any $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \in \mathbb{R}^n$.

In the context of PBO, surrogate-based methods aim solve Problem (3) starting from a set of $N \in \mathbb{N}$ *distinct samples* of the decision vector:

$$\mathcal{X} = \left\{\boldsymbol{x}_i : i = 1, \ldots, N, \boldsymbol{x}_i \in \Omega, \boldsymbol{x}_i \neq \boldsymbol{x}_j, \forall i \neq j\right\} \tag{5}$$

and a set of $M \in \mathbb{N}$ *preferences* expressed by the decision-maker:

$$\mathcal{B} = \{b_h : h = 1, \ldots, M, b_h \in \{-1, 0, 1\}\}. \tag{6}$$

The term b_h in (6) is the output of the h-th query, where the decision-maker was asked to compare two samples in $\mathcal{X}$, as highlighted by the following *mapping set*:

$$\mathcal{S} = \Big\{ (\ell(h), \kappa(h)) : h = 1, \ldots, M, \ell(h), \kappa(h) \in \mathbb{N}, b_h = \pi_{\succsim}\left(\boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right), \\ b_h \in \mathcal{B}, \boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)} \in \mathcal{X} \Big\}. \tag{7}$$

In (7), $\ell : \mathbb{N} \to \mathbb{N}$ and $\kappa : \mathbb{N} \to \mathbb{N}$ are two mapping functions that associate the indexes of the samples, contained inside $\mathcal{X}$, to their respective preferences in $\mathcal{B}$. The cardinalities of these sets are $|\mathcal{X}| = N$ and $|\mathcal{B}| = |\mathcal{S}| = M$. Also note that $1 \leq M \leq \binom{N}{2}$.

3 Handling exploration and exploitation

In this section, we review some key concepts that are common in most preference-based optimization algorithms. We also cover briefly how exploration and exploitation are handled by algorithm GLISp [3].

Preference-based response surface methods iteratively propose new samples to evaluate with the objective of solving Problem (3) while also minimizing the number of queries. Suppose that, at iteration k, we have at our disposal the set of samples $\mathcal{X}$

in (5), $|\mathcal{X}| = N$, and the sets $\mathcal{B}$ in (6) and $\mathcal{S}$ in (7), $|\mathcal{B}| = |\mathcal{S}| = M$. We define the best sample found so far as:

$$\boldsymbol{x_{best}}(N) \in \Omega : \boldsymbol{x_{best}}(N) \in \mathcal{X}, |\mathcal{X}| = N, \quad \text{and} \quad \boldsymbol{x_{best}}(N) \succsim \boldsymbol{x}_i, \forall \boldsymbol{x}_i \in \mathcal{X}.$$

The new candidate sample,

$$\boldsymbol{x}_{N+1} \in \Omega,$$

is obtained by solving an additional global optimization problem:

$$\begin{aligned} \boldsymbol{x}_{N+1} &= \arg \min_{\boldsymbol{x}} a_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega, \end{aligned} \tag{8}$$

where $a_N : \mathbb{R}^n \to \mathbb{R}$ is a properly defined *acquisition function* which trades off exploration and exploitation.

In practice, once $\boldsymbol{x}_{N+1}$ has been computed, we let the decision-maker express a preference between the best sample found so far and the new one, obtaining:

$$b_{M+1} = \pi_{\succsim}(\boldsymbol{x}_{N+1}, \boldsymbol{x_{best}}(N)).$$

After that, $\boldsymbol{x}_{N+1}$ is added to the set $\mathcal{X}$ and, similarly, the sets $\mathcal{B}$ and $\mathcal{S}$ are also updated with the new preference b_{M+1}. The process is iterated until a certain condition is met. Typically, a *budget*, or rather a maximum number of samples $N_{max} \in \mathbb{N}$, is set and the optimization procedure is stopped once it is reached.

In our case, $a_N(\boldsymbol{x})$ in (8) is defined starting from a surrogate model $\hat{f}_N : \mathbb{R}^n \to \mathbb{R}$, which approximates the scoring function $f(\boldsymbol{x})$ of Problem (3), and a function $z_N : \mathbb{R}^n \to \mathbb{R}$ that promotes the exploration of those regions of Ω where fewer samples have been evaluated. The acquisition function that we will propose in this work is a weighted sum of these two contributions (see Sect. 4.3). $\hat{f}_N(\boldsymbol{x})$ and $z_N(\boldsymbol{x})$ are defined as in `GLISp` [3], which we now review.

3.1 Surrogate model

Given N samples $\boldsymbol{x}_i \in \mathcal{X}$ in (5), we define the surrogate model $\hat{f}_N : \mathbb{R}^n \to \mathbb{R}$ as the *radial basis function expansion* [9]:

$$\begin{aligned} \hat{f}_N(\boldsymbol{x}) &= \sum_{i=1}^{N} \beta^{(i)} \cdot \varphi(\epsilon \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2) \\ &= \sum_{i=1}^{N} \beta^{(i)} \cdot \phi_i(\boldsymbol{x}) \\ &= \boldsymbol{\phi}(\boldsymbol{x})^\top \cdot \boldsymbol{\beta}, \end{aligned} \tag{9}$$

where $\varphi : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a properly chosen *radial function*, $\epsilon \in \mathbb{R}_{>0}$ is the so-called *shape parameter* (which needs to be tuned) and $\phi_i : \mathbb{R}^n \to \mathbb{R}$ is the *radial basis*

function originated from $\varphi(\cdot)$ and center $\boldsymbol{x}_i \in \mathcal{X}$, namely $\phi_i(\boldsymbol{x}) = \varphi(\epsilon \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2)$. Moreover, $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^N$, $\boldsymbol{\phi}(\boldsymbol{x}) = \left[\phi_1(\boldsymbol{x}) \dots \phi_N(\boldsymbol{x})\right]^\top$, is the radial basis function vector and $\boldsymbol{\beta} = \left[\beta^{(1)} \dots \beta^{(N)}\right]^\top \in \mathbb{R}^N$ is a vector of weights that has to be estimated from the preferences in $\mathcal{B}$ (6) and $\mathcal{S}$ (7). Given a distance $r = \|\boldsymbol{x} - \boldsymbol{x}_i\|_2$, some commonly used radial functions are [11]:

- Inverse quadratic: $\varphi(\epsilon \cdot r) = \frac{1}{1+(\epsilon \cdot r)^2}$;
- Multiquadratic: $\varphi(\epsilon \cdot r) = \sqrt{1 + (\epsilon \cdot r)^2}$;
- Linear: $\varphi(\epsilon \cdot r) = \epsilon \cdot r$;
- Gaussian: $\varphi(\epsilon \cdot r) = e^{-(\epsilon \cdot r)^2}$;
- Thin plate spline: $\varphi(\epsilon \cdot r) = (\epsilon \cdot r)^2 \cdot \log(\epsilon \cdot r)$;
- Inverse multiquadratic: $\varphi(\epsilon \cdot r) = \frac{1}{\sqrt{1+(\epsilon \cdot r)^2}}$.

One advantage of using (9) as the surrogate model is highlighted by the following Proposition.

Proposition 2 *$\hat{f}_N(\boldsymbol{x})$ in (9) is differentiable everywhere[2] if and only if the chosen radial basis function $\phi_i(\boldsymbol{x}) = \varphi(\epsilon \cdot \|\boldsymbol{x} - \boldsymbol{x}_i\|_2)$ is differentiable everywhere.*

The surrogate model in (9) can be used to define the *surrogate preference function* $\hat{\pi}_{\succsim_N} : \mathbb{R}^n \times \mathbb{R}^n \to \{-1, 0, 1\}$. Differently from $\pi_{\succsim}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in (4), we consider a tolerance $\sigma \in \mathbb{R}_{>0}$ to avoid using strict inequalities and equalities and define $\hat{\pi}_{\succsim_N}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as [3]:

$$\hat{\pi}_{\succsim_N}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} -1 & \text{if} \quad \hat{f}_N(\boldsymbol{x}_i) - \hat{f}_N(\boldsymbol{x}_j) \leq -\sigma \\ 0 & \text{if} \quad |\hat{f}_N(\boldsymbol{x}_i) - \hat{f}_N(\boldsymbol{x}_j)| \leq \sigma \\ 1 & \text{if} \quad \hat{f}_N(\boldsymbol{x}_i) - \hat{f}_N(\boldsymbol{x}_j) \geq \sigma \end{cases}. \tag{10}$$

Suppose now that we have at our disposal the sets $\mathcal{X}$ in (5), $\mathcal{B}$ in (6) and $\mathcal{S}$ in (7). Then, we are interested in a surrogate model $\hat{f}_N(\boldsymbol{x})$ in (9) that correctly describes the preferences expressed by the decision-maker, i.e. we would like the corresponding surrogate preference function $\hat{\pi}_{\succsim_N}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in (10) to be such that:

$$b_h = \hat{\pi}_{\succsim_N}\left(\boldsymbol{x}_{\ell(h)}, \boldsymbol{x}_{\kappa(h)}\right), \quad \forall b_h \in \mathcal{B}, \quad (\ell(h), \kappa(h)) \in \mathcal{S}, \quad h = 1, \dots, M.$$

This, in turn, translates into some constraints on $\hat{f}_N(\boldsymbol{x})$, which can be used to estimate $\boldsymbol{\beta}$. To do so, the authors of GLISp [3] define the following convex optimization

[2] Note that, whenever we say that a multivariable function, such as $\hat{f}_N(\boldsymbol{x})$ in (9), is "differentiable everywhere" we imply that it is differentiable at each $\boldsymbol{x} \in \mathbb{R}^n$.

problem:

$$
\begin{aligned}
& \arg \min_{\boldsymbol{\varepsilon}, \boldsymbol{\beta}} \frac{\lambda}{2} \cdot \boldsymbol{\beta}^{\top} \cdot \boldsymbol{\beta} + \boldsymbol{r}^{\top} \cdot \boldsymbol{\varepsilon} \\
\text{s.t.} \quad & \hat{f}_N\left(\boldsymbol{x}_{\ell(h)}\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}\right) \leq -\sigma + \varepsilon^{(h)} \qquad \forall h : b_h = -1 \\
& \left|\hat{f}_N\left(\boldsymbol{x}_{\ell(h)}\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}\right)\right| \leq \sigma + \varepsilon^{(h)} \qquad \forall h : b_h = 0 \\
& \hat{f}_N\left(\boldsymbol{x}_{\ell(h)}\right) - \hat{f}_N\left(\boldsymbol{x}_{\kappa(h)}\right) \geq \sigma - \varepsilon^{(h)} \qquad \forall h : b_h = 1 \\
& \varepsilon^{(h)} \geq 0 \\
& h = 1, \ldots, M,
\end{aligned} \tag{11}
$$

where:

- $\boldsymbol{\varepsilon} = \left[\varepsilon^{(1)} \ \ldots \ \varepsilon^{(M)}\right]^{\top} \in \mathbb{R}^{M}_{\geq 0}$ is a vector of *slack variables* (one for each preference) which takes into consideration that: (i) there might be some outliers in $\mathcal{B}$ and $\mathcal{S}$ if the decision-maker expresses the preferences in an inconsistent way, and (ii) the selected radial function and/or shape parameter for $\hat{f}_N(\boldsymbol{x})$ in (9) do not allow a proper approximation of the scoring function $f(\boldsymbol{x})$;
- $\boldsymbol{r} = \left[r^{(1)} \ \ldots \ r^{(M)}\right]^{\top} \in \mathbb{R}^{M}_{>0}$ is a vector of weights that can be used to penalize more some slacks related to the most important preferences. In `GLISp` [3], the authors weigh more the preferences associated to the current best candidate and define $\boldsymbol{r}$ as follows:

$$
\begin{aligned}
& r^{(h)} = 1, \quad \forall h : (\ell(h), \kappa(h)) \in \mathcal{S}, \boldsymbol{x}_{\ell(h)} \neq \boldsymbol{x}_{\boldsymbol{best}}(N) \quad \text{and} \quad \boldsymbol{x}_{\kappa(h)} \neq \boldsymbol{x}_{\boldsymbol{best}}(N), \\
& r^{(h)} = 10, \quad \forall h : (\ell(h), \kappa(h)) \in \mathcal{S}, \boldsymbol{x}_{\ell(h)} = \boldsymbol{x}_{\boldsymbol{best}}(N) \quad \text{or} \quad \boldsymbol{x}_{\kappa(h)} = \boldsymbol{x}_{\boldsymbol{best}}(N).
\end{aligned}
$$

- $\lambda \in \mathbb{R}_{\geq 0}$ plays the role of a *regularization parameter*. It is easy to see that, for $\lambda = 0$, Problem (11) is a Linear Program (LP) while, for $\lambda > 0$, it is a Quadratic Program (QP).

Problem (11) ensures that, at least approximately, $\hat{f}_N(\boldsymbol{x})$ in (9) is a suitable representation of the unknown preference relation $\succsim$ on Ω which produced the data described in Sect. 2.1 (see Theorem 1).

3.2 Exploration function

Consider a sample $\boldsymbol{x}_i \in \mathcal{X}$ in (5). Its corresponding *Inverse Distance Weighting (IDW) function* $w_i : \mathbb{R}^n \setminus \{\boldsymbol{x}_i\} \to \mathbb{R}_{>0}$ is defined as [31]:

$$
w_i(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2}. \tag{12}
$$

GLISp [3] uses the so-called *IDW distance function* $z_N : \mathbb{R}^n \to (-1, 0]$,

$$z_N(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \mathcal{X} \\ -\frac{2}{\pi} \cdot \arctan\left(\frac{1}{\sum_{i=1}^N w_i(\boldsymbol{x})}\right) & \text{otherwise} \end{cases}, \tag{13}$$

to promote the exploration in those regions of $\mathbb{R}^n$ where fewer samples have been evaluated. It is possible to prove the following Proposition and Lemma.

Proposition 3 *The IDW distance function $z_N(\boldsymbol{x})$ in (13) is differentiable everywhere.*

The proof of Proposition 3 is given in [2]. Here, we derive the gradient of $z_N(\boldsymbol{x})$ in (13) (see "Appendix A" for its derivation).

Lemma 1 *The gradient of the IDW distance function $z_N(\boldsymbol{x})$ in (13) is:*

$$\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) = \begin{cases} \mathbf{0}_n & \text{if } \boldsymbol{x} \in \mathcal{X} \\ -\frac{4}{\pi} \cdot \frac{\sum_{i=1}^N (\boldsymbol{x}-\boldsymbol{x}_i) \cdot w_i(\boldsymbol{x})^2}{1+\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2} & \text{otherwise} \end{cases}. \tag{14}$$

The gradient $\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x})$ in (14) will be particularly relevant in the following section.

4 Next candidate sample search

As previously mentioned in Sect. 3, a key aspect of surrogate-based methods is the exploration–exploitation dilemma. Typically, new candidate samples are sought by minimizing an acquisition function $a_N(\boldsymbol{x})$ that is a weighted sum between the surrogate model and the exploration function. In practice, $\hat{f}_N(\boldsymbol{x})$ in (9) and $z_N(\boldsymbol{x})$ in (13) often exhibit different ranges and need to be rescaled. In particular, GLISp [3] adopts the following $a_N(\boldsymbol{x})$:

$$a_N(\boldsymbol{x}) = \frac{\hat{f}_N(\boldsymbol{x})}{\Delta \hat{F}} + \delta \cdot z_N(\boldsymbol{x}), \tag{15}$$

where $\delta \in \mathbb{R}_{\geq 0}$ is the exploration–exploitation trade-off weight. The division by

$$\Delta \hat{F} = \max_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i) - \min_{\boldsymbol{x}_i \in \mathcal{X}} \hat{f}_N(\boldsymbol{x}_i) \tag{16}$$

aims to rescale the surrogate model to make it assume a range that is comparable to that of the IDW distance function in (13), which is $(-1, 0]$.

In this section, we address some limitations of $a_N(\boldsymbol{x})$ in (15), which might prevent GLISp [3] from reaching the global minimizer of Problem (3), and define an alternative acquisition function. Furthermore, we propose a strategy to iteratively vary the exploration–exploitation trade-off.

4.1 Shortcomings of GLISp

There are two shortcomings of $a_N(\boldsymbol{x})$ in (15) that limit the exploratory capabilities of GLISp [3]. First, the rescaling of $\hat{f}_N(\boldsymbol{x})$ in (15), which relies on $\Delta\hat{F}$ in (16), only takes into account the previously evaluated samples inside $\mathcal{X}$ in (5) and thus it can be ineffective in making $\frac{\hat{f}_N(\boldsymbol{x})}{\Delta\hat{F}}$ and $z_N(\boldsymbol{x})$ comparable over all Ω (see Problem (8)). Second, the IDW distance function in (13) exhibits two characteristics that can make its contribution negligible in $a_N(\boldsymbol{x})$ in (15) and complicate the selection of δ:

1. Even though the range of $z_N(\boldsymbol{x})$ is $(-1, 0]$, what we are really interested in when solving Problem (8) and, ultimately, Problem (3), are the values that it assumes for $\boldsymbol{x} \in \Omega$ and not on its whole domain $\mathbb{R}^n$. In particular, there are some situations for which $|z_N(\boldsymbol{x})| \ll 1, \forall \boldsymbol{x} \in \Omega$. Consider, for example, the case $\mathcal{X} = \{\boldsymbol{x}_1\}$ $(N = 1)$; then, $\forall \boldsymbol{x} \in \mathbb{R}^n \backslash \mathcal{X}$, the IDW distance function simply becomes:

$$z_1(\boldsymbol{x}) = -\frac{2}{\pi} \cdot \arctan\left(\|\boldsymbol{x} - \boldsymbol{x}_1\|_2^2\right).$$

   Suppose that Problem (3) is only bound constrained, i.e. $\Omega = \{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$. Then, $z_1(\boldsymbol{x})$ assumes its lowest value at one (or more) of the vertices of the box defined by the constraints in Ω. Define $\boldsymbol{v}_\Omega \in \Omega$ as one of such vertices; if $\|\boldsymbol{v}_\Omega - \boldsymbol{x}_1\|_2$ is close to zero, then $|z_N(\boldsymbol{x})| \ll 1, \forall \boldsymbol{x} \in \Omega$. Thus, unless $\delta \gg 1$ in (15), $\hat{f}_N(\boldsymbol{x})$ in (9) and $z_N(\boldsymbol{x})$ in (13) might not be comparable.
2. The (absolute) values assumed by the IDW distance function decrease as the number of samples increases. To clarify this, consider two sets of samples:

$$\begin{aligned} \mathcal{X}' &= \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}, \quad |\mathcal{X}'| = N, \\ \mathcal{X}'' &= \mathcal{X}' \cup \{\boldsymbol{x}_{N+1}\}, \quad |\mathcal{X}''| = N + 1. \end{aligned}$$

   Given any point $\tilde{\boldsymbol{x}} \in \mathbb{R}^n \backslash \mathcal{X}''$, the IDW distance functions obtained from the previously defined sets are:

$$\begin{aligned} z_N(\tilde{\boldsymbol{x}}) &= -\frac{2}{\pi} \cdot \arctan\left(\frac{1}{\sum_{i=1}^{N} w_i(\tilde{\boldsymbol{x}})}\right), \\ z_{N+1}(\tilde{\boldsymbol{x}}) &= -\frac{2}{\pi} \cdot \arctan\left(\frac{1}{\sum_{i=1}^{N+1} w_i(\tilde{\boldsymbol{x}})}\right). \end{aligned}$$

   Note that $w_i(\tilde{\boldsymbol{x}}) > 0, \forall \tilde{\boldsymbol{x}} \in \mathbb{R}^n \backslash \mathcal{X}''$ and $i = 1, \ldots, N + 1$ (see (12)). Hence:

$$|z_N(\tilde{\boldsymbol{x}})| > |z_{N+1}(\tilde{\boldsymbol{x}})| > 0,$$

   proving the above point. In practice, unless δ in (15) is progressively increased as the iterations go on, GLISp [3] will explore the constraint set Ω of Problem (3) less as the number of samples increases, regardless of whether a region that contains the global minimizer $\boldsymbol{x}^*$ has been located.

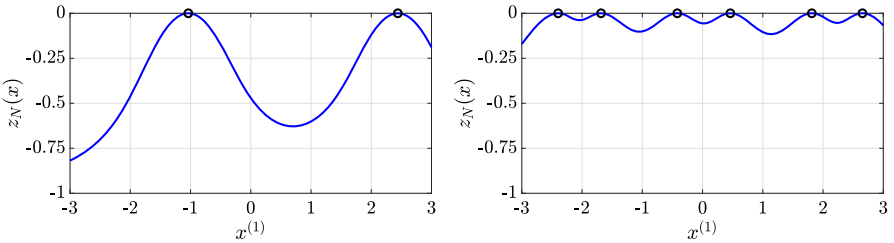


Fig. 1 Examples of the IDW distance function $z_N(x)$ in (13) for different numbers of points ($N = 2$ on the left, $N = 6$ on the right) and $-3 = l \leq x \leq u = 3$. Notice how $z_N(x)$ does not cover its whole range $(-1, 0]$, at least inside the bound constraints, and its absolute values decrease as the number of samples increases

A visualization of these two characteristics of $z_N(\boldsymbol{x})$ in (13) is presented in Fig. 1.

In this work, we overcome the limitations of $a_N(\boldsymbol{x})$ in (15) by defining an acquisition function that is similar to the one used by the Metric Stochastic Response Surface (MSRS [29]) algorithm, a popular black-box optimization procedure. In MSRS [29], the surrogate $\hat{f}_N(\boldsymbol{x})$ and the exploration function $z_N(\boldsymbol{x})$ are made comparable by randomly sampling Ω and rescaling the two using min–max normalization. Then, Problem (8) is not solved explicitly but by choosing the generated random sample that achieves the lowest value for the acquisition function. Here, we propose to rescale $z_N(\boldsymbol{x})$ in (13) and $\hat{f}_N(\boldsymbol{x})$ in (9) using some insights on the stationary points of the IDW distance function and solve Problem (8) explicitly, using a proper global optimization solver.

4.2 Novel rescaling strategy

In this section, we derive the approximate locations of the stationary points of the IDW distance function in (13) and use them to define an augmented set of samples $\mathcal{X}_{aug} \supset \mathcal{X}$ that is suited for the min–max normalization of both $z_N(\boldsymbol{x})$ and $\hat{f}_N(\boldsymbol{x})$.

4.2.1 Stationary points of the IDW distance function

The locations of the global maximizers of $z_N(\boldsymbol{x})$ can be deduced immediately from (13) and Lemma 1, as stated by the following Proposition.

Proposition 4 *Each $\boldsymbol{x}_i \in \mathcal{X}$ in (5) is a global maximizer of $z_N(\boldsymbol{x})$ in (13).*

Proof Recall that:

(i) $\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}_i) = \mathbf{0}_n, \forall \boldsymbol{x}_i \in \mathcal{X}$, see (14);
(ii) $z_N(\boldsymbol{x}) < 0, \forall \boldsymbol{x} \in \mathbb{R}^n \backslash \mathcal{X}$, see (13);
(iii) $z_N(\boldsymbol{x}_i) = 0, \forall \boldsymbol{x}_i \in \mathcal{X}$, see (13).

From Item (i) we deduce that each $\boldsymbol{x}_i \in \mathcal{X}$ is a stationary point of $z_N(\boldsymbol{x})$. Item (ii), in conjunction with Item (iii), implies that such samples are local maximizers of the IDW distance function in (13) since there exists a neighborhood of $\boldsymbol{x}_i \in \mathcal{X}$, denoted as $\mathcal{N}(\boldsymbol{x}_i)$, such that $z_N(\boldsymbol{x}) \leq z_N(\boldsymbol{x}_i), \forall \boldsymbol{x} \in \mathcal{N}(\boldsymbol{x}_i)$. Moreover, due to Item (ii),

$z_N(\boldsymbol{x}) \leq z_N(\boldsymbol{x}_i), \forall \boldsymbol{x} \in \mathbb{R}^n$ and not just in a neighborhood of $\boldsymbol{x}_i \in \mathcal{X}$. Hence, each $\boldsymbol{x}_i \in \mathcal{X}$ is a global maximizer of $z_N(\boldsymbol{x})$ in (13). □

Reaching similar conclusions for the minimizers of $z_N(\boldsymbol{x})$ is much harder; however, we can consider some simplified situations. Note that we are not necessarily interested in finding the minimizers of the IDW distance function in (13) with high accuracy, but rather to gain some insights on where they are likely to be located so that we can rescale both $z_N(\boldsymbol{x})$ in (13) and $\hat{f}_N(\boldsymbol{x})$ in (9) sufficiently enough to make them comparable. Moreover, their approximate locations can be used to solve the following global optimization problem (*pure exploration*):

$$\begin{aligned} \boldsymbol{x}_{N+1} &= \arg\min_{\boldsymbol{x}} z_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega \end{aligned} \tag{17}$$

by using a multi-start derivative-based optimization method with warm-start [23, 26] (recall that $z_N(\boldsymbol{x})$ is differentiable everywhere, see Proposition 3). Problem (17) is quite relevant for the global convergence of the algorithm that we will propose in Sect. 5.

Remark 2 In the following Paragraphs, we analyze where the local minimizers of $z_N(\boldsymbol{x})$ in (13) and the solution(s) of the simplified problem:

$$\begin{aligned} \boldsymbol{x}_{N+1} &= \arg\min_{\boldsymbol{x}} z_N(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \end{aligned} \tag{18}$$

are located in some specific cases. Note that $\{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\} \supseteq \Omega$ (see (1)) and thus the global minimum of Problem (18) is lower than or at most equal to the global minimum of Problem (17). Therefore, the minimizers of Problem (18) are better suited to perform min–max rescaling of $z_N(\boldsymbol{x})$ than those of Problem (17).

Case $\mathcal{X} = \{\boldsymbol{x}_1\}$ $(N = 1)$. The IDW distance function and its gradient $\forall \boldsymbol{x} \in \mathbb{R}^n \backslash \mathcal{X}$ are:

$$\begin{aligned} z_N(\boldsymbol{x}) &= -\frac{2}{\pi} \cdot \arctan\left(\|\boldsymbol{x} - \boldsymbol{x}_1\|_2^2\right), \\ \nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) &= -\frac{4}{\pi} \cdot (\boldsymbol{x} - \boldsymbol{x}_1) \cdot \frac{w_1(\boldsymbol{x})^2}{1 + w_1(\boldsymbol{x})^2}. \end{aligned}$$

Clearly, $\forall \boldsymbol{x} \in \mathbb{R}^n \backslash \mathcal{X}$, the gradient is never zero since $w_1(\boldsymbol{x}) > 0$. Therefore, the only stationary point is the global maximizer $\boldsymbol{x}_1$ (see Proposition 4). However, if we were to consider Problem (18), then its solution would be located at one of the vertices of the box defined by the bound constraints $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$.

Case $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ $(N = 2)$. The gradient of the IDW distance function $\forall \boldsymbol{x} \in \mathbb{R}^n \backslash \mathcal{X}$ is:

$$\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) = -\frac{4}{\pi} \cdot \frac{(\boldsymbol{x} - \boldsymbol{x}_1) \cdot w_1(\boldsymbol{x})^2 + (\boldsymbol{x} - \boldsymbol{x}_2) \cdot w_2(\boldsymbol{x})^2}{1 + [w_1(\boldsymbol{x}) + w_2(\boldsymbol{x})]^2}.$$

Let us consider the midpoint $\boldsymbol{x}_{\boldsymbol{\mu}} = \frac{\boldsymbol{x}_1+\boldsymbol{x}_2}{2}$, that is such that $\left\|\boldsymbol{x}_{\boldsymbol{\mu}} - \boldsymbol{x}_1\right\|_2 = \left\|\boldsymbol{x}_{\boldsymbol{\mu}} - \boldsymbol{x}_2\right\|_2$ and for which $w_1\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right) = w_2\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right)$. If we substitute it in the previous expression, we obtain:

$$\nabla_{\boldsymbol{x}} z_N\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right) = \mathbf{0}_n,$$

which means that $\boldsymbol{x}_{\boldsymbol{\mu}}$ is a stationary point for $z_N(\boldsymbol{x})$ in (13). It is easy to see by visual inspection that such point is actually a local minimizer for the IDW distance function (see for example Fig. 2). However, note that $\boldsymbol{x}_{\boldsymbol{\mu}}$ is not necessarily the global solution of Problem (18), it might just be a local one.

Case $\mathcal{X} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)}$ $(N > 2)$. Suppose now that the samples contained in $\mathcal{X}$ (5) can be partitioned into two clusters:

- $\mathcal{X}^{(1)} = \left\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_1}\right\}$ $(|\mathcal{X}^{(1)}| = N_1)$,
- $\mathcal{X}^{(2)} = \left\{\boldsymbol{x}_{N_1+1}, \ldots, \boldsymbol{x}_N\right\}$ $(|\mathcal{X}^{(2)}| = N - N_1)$,

such that $\mathcal{X}^{(1)} \cap \mathcal{X}^{(2)} = \emptyset$ and $\mathcal{X}^{(1)} \cup \mathcal{X}^{(2)} = \mathcal{X}$. Consider the midpoint between the centroids of each cluster:

$$\boldsymbol{x}_{\boldsymbol{\mu}} = \frac{1}{2} \cdot \left[\frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(1)}} \boldsymbol{x}_i}{N_1} + \frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(2)}} \boldsymbol{x}_i}{N - N_1}\right]. \tag{19}$$

We make the simplifying assumption that all the points contained inside each cluster are quite close to each other, i.e. $\boldsymbol{x}_1 \approx \boldsymbol{x}_2 \approx \cdots \approx \boldsymbol{x}_{N_1}$ and $\boldsymbol{x}_{N_1+1} \approx \boldsymbol{x}_{N_1+2} \approx \cdots \approx \boldsymbol{x}_N$. Then, the midpoint in (19) is approximately equal to $\boldsymbol{x}_{\boldsymbol{\mu}} \approx \frac{\boldsymbol{x}_1+\boldsymbol{x}_N}{2}$. Moreover, we have that $w_1\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right) \approx \cdots \approx w_{N_1}\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right) \approx w_{N_1+1}\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right) \approx \cdots \approx w_N\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right)$. Thus, the gradient of the IDW distance function at $\boldsymbol{x}_{\boldsymbol{\mu}}$ is approximately equal to:

$$\begin{aligned}
\nabla_{\boldsymbol{x}} z_N\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right) &= -\frac{4}{\pi} \cdot \frac{\sum_{i=1}^N \left(\boldsymbol{x}_{\boldsymbol{\mu}} - \boldsymbol{x}_i\right) \cdot w_i\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right)^2}{1 + \left[\sum_{i=1}^N w_i\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right)\right]^2} \\
&\approx -\frac{4}{\pi} \cdot \frac{w_1\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right)^2}{1 + \left[N \cdot w_1\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right)\right]^2} \cdot \left[\left(\frac{N}{2} - N_1\right) \cdot \boldsymbol{x}_1 + \left(-\frac{N}{2} + N_1\right) \cdot \boldsymbol{x}_N\right].
\end{aligned}$$

Clearly, if the clusters are nearly equally sized, i.e. $N_1 \approx N - N_1 \approx \frac{N}{2}$, then:

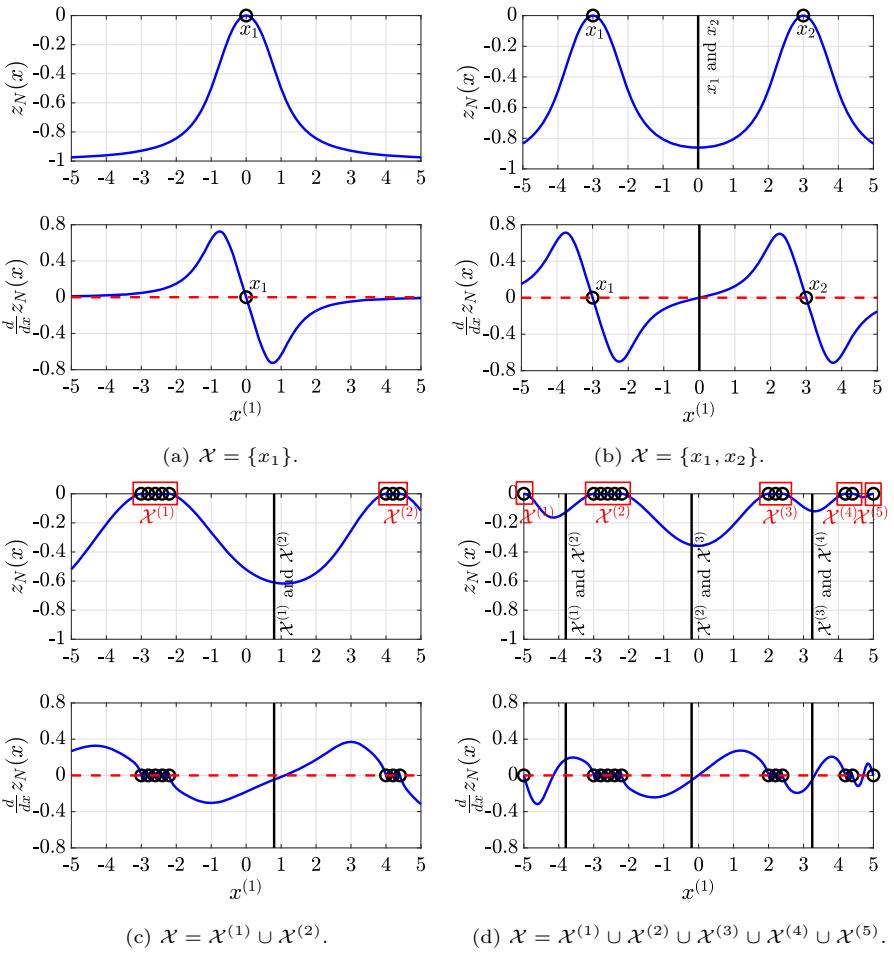
$$\nabla_{\boldsymbol{x}} z_N\left(\boldsymbol{x}_{\boldsymbol{\mu}}\right) \approx \mathbf{0}_n,$$

reaching a similar result to the one that we have seen for the case $N = 2$.

General case $(N > 2)$. Any set of samples $\mathcal{X}$ in (5) can be partitioned into an arbitrary number of disjoint clusters, say $K \in \mathbb{N}$, $K \leq |\mathcal{X}| = N$, i.e.:

$$\mathcal{X} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)} \cup \cdots \cup \mathcal{X}^{(K)}, \quad \text{such that} \quad \mathcal{X}^{(i)} \cap \mathcal{X}^{(j)} = \emptyset, \forall i \neq j.$$

In this case, finding the local solutions of Problem (18) explicitly, or even approximately, is quite complex. Heuristically speaking, if the clusters are "well spread" (i.e. all the points contained inside each cluster $\mathcal{X}^{(i)}$ are sufficiently far away from the



(a) $\mathcal{X} = \{x_1\}$. (b) $\mathcal{X} = \{x_1, x_2\}$.

(c) $\mathcal{X} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)}$. (d) $\mathcal{X} = \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)} \cup \mathcal{X}^{(3)} \cup \mathcal{X}^{(4)} \cup \mathcal{X}^{(5)}$.

Fig. 2 One-dimensional examples of the IDW distance function $z_N(x)$ in (13) and its gradient $\nabla_x z_N(x)$ in (14) in the four analyzed cases. The red boxes mark the different clusters while the red dashed lines highlight the values of x for which the first derivative of $z_N(x)$ is zero. Finally, the black vertical lines mark the midpoints (either between points or centroids of the clusters). Only a portion of all possible midpoints between centroids has been reported in the general case. Notice that the midpoints in Fig. 2c and d are quite close to the local minimizers of $z_N(\boldsymbol{x})$, while the midpoint in Fig. 2b is an exact local solution of Problem (18) (color figure online)

others in $\mathcal{X}^{(j)}$, $j = 1, \ldots, K$, $j \neq i$), then we can approximately deduce where the local minimizers of $z_N(\boldsymbol{x})$ in (13) are located.

For instance, Fig. 3 depicts a set of samples $\mathcal{X}$ that has been partitioned into three clusters, $\mathcal{X}^{(1)}$, $\mathcal{X}^{(2)}$ and $\mathcal{X}^{(3)}$, and for which the previous hypothesis is satisfied.

In the general case, given the clusters $\mathcal{X}^{(i)}$ and $\mathcal{X}^{(j)}$, we compute their centroids $\boldsymbol{x}_{\boldsymbol{c}}^{(i)}$, $\boldsymbol{x}_{\boldsymbol{c}}^{(j)}$, and the corresponding midpoint $\boldsymbol{x}_{\boldsymbol{\mu}}$ between them as:

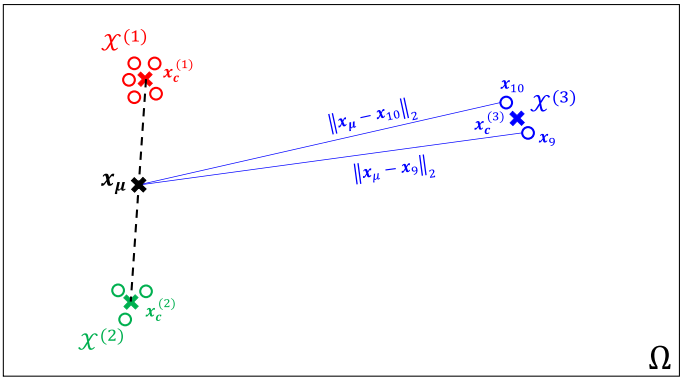


Fig. 3 Two-dimensional example of "well spread" clusters, highlighted with different colors. The circles denote the points contained in $\mathcal{X}$ while the crosses represent the centroids of $\mathcal{X}^{(1)}$, $\mathcal{X}^{(2)}$ and $\mathcal{X}^{(3)}$. $\boldsymbol{x_\mu}$ is the midpoint between the centroids of clusters $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$. Finally, the blue lines highlight the distances between the samples of cluster $\mathcal{X}^{(3)}$ and $\boldsymbol{x_\mu}$ (color figure online)

$$\boldsymbol{x}_{\boldsymbol{c}}^{(k)} = \frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(k)}} \boldsymbol{x}_i}{|\mathcal{X}^{(k)}|} \quad \text{(centroid of } k\text{-th cluster)}, \tag{20a}$$

$$\boldsymbol{x_\mu} = \frac{\boldsymbol{x}_{\boldsymbol{c}}^{(i)} + \boldsymbol{x}_{\boldsymbol{c}}^{(j)}}{2} \quad \text{(midpoint)}. \tag{20b}$$

Going back to the example depicted in Fig. 3, if we consider the midpoint $\boldsymbol{x_\mu}$ between the centroids of $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$, due to the "well spread" hypothesis we can say that $\left\|\boldsymbol{x_\mu} - \boldsymbol{x}_i\right\|_2 \gg 0, \forall \boldsymbol{x}_i \in \mathcal{X}^{(3)}$, making the contributions of the points inside the third cluster negligible when evaluating $z_N(\boldsymbol{x})$ in (13) at $\boldsymbol{x_\mu}$. Therefore, the IDW distance function at $\boldsymbol{x_\mu}$ is approximately equal to:

$$\begin{aligned} z_N(\boldsymbol{x_\mu}) &= -\frac{2}{\pi} \cdot \arctan\left\{\left[\sum_{k=1}^{3}\left(\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(k)}} \frac{1}{\left\|\boldsymbol{x_\mu} - \boldsymbol{x}_i\right\|_2^2}\right)\right]^{-1}\right\} \\ &\approx -\frac{2}{\pi} \cdot \arctan\left[\left(\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(1)}} \frac{1}{\left\|\boldsymbol{x_\mu} - \boldsymbol{x}_i\right\|_2^2} + \sum_{\boldsymbol{x}_i \in \mathcal{X}^{(2)}} \frac{1}{\left\|\boldsymbol{x_\mu} - \boldsymbol{x}_i\right\|_2^2}\right)^{-1}\right]. \end{aligned}$$

In general, given K clusters, if these are "well spread", then we can consider each possible couple of clusters separately and neglect the contributions of the remaining ones. Approximately speaking, we could split the general case into $\binom{K}{2}$ distinct problems that read as follows: find the stationary points of the IDW distance function $z_{N_{i\cup j}}(\boldsymbol{x})$ in (13) defined from the set of samples $\mathcal{X}^{(i)} \cup \mathcal{X}^{(j)}, i \neq j$ and $N_{i\cup j} = |\mathcal{X}^{(i)} \cup \mathcal{X}^{(j)}|$. Hence, rough locations of the stationary points of $z_{N_{i\cup j}}(\boldsymbol{x})$ can be found by following the same rationale proposed for the previously analyzed cases.

Some one-dimensional examples of all the previously analyzed situations are reported in Fig. 2.

4.2.2 Min–max rescaling and augmented sample set

Given a generic set of samples $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and a multivariable function $h : \mathbb{R}^n \to \mathbb{R}$, *min–max rescaling* (or normalization) [16] rescales $h(\boldsymbol{x})$ as:

$$\bar{h}(\boldsymbol{x}; \mathcal{X}) = \frac{h(\boldsymbol{x}) - h_{min}(\mathcal{X})}{\Delta H(\mathcal{X})}, \tag{21}$$

where:[3]

$$h_{min}(\mathcal{X}) = \min_{\boldsymbol{x}_i \in \mathcal{X}} h(\boldsymbol{x}_i), \tag{22a}$$

$$h_{max}(\mathcal{X}) = \max_{\boldsymbol{x}_i \in \mathcal{X}} h(\boldsymbol{x}_i), \tag{22b}$$

$$\Delta H(\mathcal{X}) = h_{max}(\mathcal{X}) - h_{min}(\mathcal{X}). \tag{22c}$$

The objective of min–max rescaling is to obtain a function with range [0, 1], i.e. we would like to have $\bar{h} : \mathbb{R}^n \to [0, 1]$. Clearly, the quality of the normalization depends on the information brought by the samples contained inside $\mathcal{X}$, as pointed out in the following Remark.

Remark 3 We can observe that:

1. If $\mathcal{X}$ contains the global minimizer(s) and maximizer(s) of $h(\boldsymbol{x})$, then $\bar{h}(\boldsymbol{x})$ defined as in (21) effectively has codomain [0, 1],
2. Otherwise, we can only ensure that $0 \leq \bar{h}(\boldsymbol{x}_i) \leq 1, \forall \boldsymbol{x}_i \in \mathcal{X}$.
3. In general, if we increase the amount of distinct samples in $\mathcal{X}$, then the rescaling of $h(\boldsymbol{x})$ gets better (or, worst case, stays the same).

Going back to the problem of rescaling the IDW distance function $z_N(\boldsymbol{x})$ in (13), if we were to apply (21) using the set of previously evaluated samples $\mathcal{X}$ in (5), then it would not be effective since $z_N(\boldsymbol{x}_i) = 0, \forall \boldsymbol{x}_i \in \mathcal{X}$ (see Proposition 4). Instead, we have opted to generate a sufficiently expressive *augmented sample set* $\mathcal{X}_{aug} \supset \mathcal{X}$ and perform min–max normalization using $\mathcal{X}_{aug}$ instead of $\mathcal{X}$.

Consider the general case described in Sect. 4.2.1. Then, the augmented sample set $\mathcal{X}_{aug}$ can be built in the following fashion:

1. Partition the points in $\mathcal{X}$ (5) into different clusters. Here, for simplicity, we fix a-priori the number $K_{aug} \in \mathbb{N}$ of clusters and apply K-means clustering [5, 17, 22] to obtain the sets $\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(K_{aug})}$;
2. Compute the centroids of each cluster, using (20a), and group them inside the set $\mathcal{X}_c = \left\{\boldsymbol{x}_{\boldsymbol{c}}^{(1)}, \ldots, \boldsymbol{x}_{\boldsymbol{c}}^{(K_{aug})}\right\}$;

[3] Note that, to avoid dividing by zero in (21), $\Delta H(\mathcal{X})$ can be set to $h_{max}(\mathcal{X})$ or 1 whenever $h_{min}(\mathcal{X}) = h_{max}(\mathcal{X}) \neq 0$ or $h_{min}(\mathcal{X}) = h_{max}(\mathcal{X}) = 0$ respectively.

3. Calculate all the midpoints $\boldsymbol{x}_{\boldsymbol{\mu}}$ between each possible couple of centroids $\boldsymbol{x}_{\boldsymbol{c}}^{(i)}, \boldsymbol{x}_{\boldsymbol{c}}^{(j)} \in \mathcal{X}_c, \boldsymbol{x}_{\boldsymbol{c}}^{(i)} \neq \boldsymbol{x}_{\boldsymbol{c}}^{(j)}$, using (20b);
4. Build the augmented sample set as $\mathcal{X}_{aug} = \mathcal{X} \cup \mathcal{X}_{\mu}$, where $\mathcal{X}_{\mu}$ is the set which groups all the previously computed midpoints.

Clearly, as highlighted by (21) and Remark 3, if $\mathcal{X}_{aug}$ contains points that are close (or equal) to the global minimizer(s) and maximizer(s) of $z_N(\boldsymbol{x})$ in (13), then the quality of the min–max rescaling of the IDW distance function improves.

Algorithm 1 formalizes these steps while also taking into consideration the case $|\mathcal{X}| \leq K_{aug}$ (for which no clustering is performed). Note that we also include the bounds $\boldsymbol{l}$ and $\boldsymbol{u}$ inside $\mathcal{X}_c$ and $\mathcal{X}_{aug}$ for two reasons: (i) $\boldsymbol{l}$ or $\boldsymbol{u}$ might actually be the solutions of Problem (18)[4] and (ii) given that we also want to rescale $\hat{f}_N(\boldsymbol{x})$ in (9), adding additional points to the augmented sample set improves the quality of min–max normalization (see Remark 3). Notice that the number of points contained inside $\mathcal{X}_{aug}$ obtained from Algorithm 1 is:

$$|\mathcal{X}_{aug}| = |\mathcal{X}| + \binom{K_{aug}+2}{2} + 2.$$

Therefore, to avoid excessively large augmented sample sets, K_{aug} needs to be chosen appropriately.

As a final remark, we point out that we could perform min–max normalization in (21) by using the real minima and maxima of $z_N(\boldsymbol{x})$ in (13) and $\hat{f}_N(\boldsymbol{x})$ in (9), which can be obtained by solving four additional global optimization problems. However, we have preferred to stick with the proposed heuristic way since we are not interested in an extremely accurate rescaling and, also, to avoid potentially large overhead times due to solving additional global optimization problems.

4.3 Definition of the acquisition function

In this section, we take advantage of the results on the stationary points of $z_N(\boldsymbol{x})$ presented in Sect. 4.2.1 to rescale the surrogate model and the exploration function. In particular, we define the following acquisition function:

$$a_N(\boldsymbol{x}) = \delta \cdot \hat{\bar{f}}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right) + (1-\delta) \cdot \bar{z}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right), \tag{23}$$

where $\hat{f}_N(\boldsymbol{x})$ in (9) and $z_N(\boldsymbol{x})$ in (13) have been rescaled using min–max normalization as in (21) and $\mathcal{X}_{aug}$ is generated by Algorithm 1. $\delta \in [0,1]$ is the exploration–exploitation trade-off weight; also note that $\delta = 0$ corresponds to pure exploration, while $\delta = 1$ results in pure exploitation. $a_N(\boldsymbol{x})$ in (23) is similar to the acquisition function of MSRS [29] (for black-box optimization) but here we use an ad-hoc augmented sample set instead of a randomly generated one and a different

[4] We could add all 2^n vertices of the box defined by the bound constraints $\{\boldsymbol{x} : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$. However, we have preferred to include only $\boldsymbol{l}$ and $\boldsymbol{u}$ to avoid increasing the cardinality of the augmented sample set, especially in the case of high-dimensional problems.

Algorithm 1 Computation of $\mathcal{X}_{aug}$ for min-max rescaling

Input: (i) Set of samples $\mathcal{X}$ in (5); (ii) Number of clusters $K_{aug} \in \mathbb{N}$; (iii) Lower bounds $\boldsymbol{l} \in \mathbb{R}^n$ and upper bounds $\boldsymbol{u} \in \mathbb{R}^n$ of Problem (3).
Output: (i) Augmented sample set $\mathcal{X}_{aug} \supset \mathcal{X}$.

1: **if** $|\mathcal{X}| > K_{aug}$ **then**
2: Perform K-means clustering [5, 17, 22] to group the samples in $\mathcal{X}$ into K_{aug} clusters $\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(K_{aug})}$
3: Compute the set of centroids $\mathcal{X}_c$ using (20a):

$$\mathcal{X}_c = \left\{ \boldsymbol{x}_c^{(k)} : \boldsymbol{x}_c^{(k)} = \frac{\sum_{\boldsymbol{x}_i \in \mathcal{X}^{(k)}} \boldsymbol{x}_i}{|\mathcal{X}^{(k)}|}, k = 1, \ldots, K_{aug} \right\}$$

4: **else**
5: Set $\mathcal{X}_c = \mathcal{X}$
6: **end if**
7: Add the bounds to $\mathcal{X}_c$: $\mathcal{X}_c = \mathcal{X}_c \cup \{\boldsymbol{l}, \boldsymbol{u}\}$
8: Group all possible couples of $\mathcal{X}_c$ (without repetition):

$$\mathcal{X}_{couples} = \left\{ \left(\boldsymbol{x}_c^{(i)}, \boldsymbol{x}_c^{(j)} \right) : \boldsymbol{x}_c^{(i)}, \boldsymbol{x}_c^{(j)} \in \mathcal{X}_c, \boldsymbol{x}_c^{(i)} \neq \boldsymbol{x}_c^{(j)} \right\}$$

9: Calculate the midpoints between all the couples inside $\mathcal{X}_{couples}$, obtaining the set:

$$\mathcal{X}_\mu = \left\{ \boldsymbol{x}_\mu : \boldsymbol{x}_\mu = \frac{\boldsymbol{x}_c^{(i)} + \boldsymbol{x}_c^{(j)}}{2}, \left(\boldsymbol{x}_c^{(i)}, \boldsymbol{x}_c^{(j)} \right) \in \mathcal{X}_{couples} \right\}$$

10: Build the augmented sample set as $\mathcal{X}_{aug} = \mathcal{X} \cup \mathcal{X}_\mu \cup \{\boldsymbol{l}, \boldsymbol{u}\}$

exploration function. We will refer to the algorithm that we will propose in Sect. 5, which uses $a_N(\boldsymbol{x})$ in (23), as `GLISp-r`, where the "`r`" highlights the min–max rescaling performed for the acquisition function.

A comparison between the terms of the acquisition functions in (23) (`GLISp-r`) and in (15) (`GLISp` [3]) is depicted in Fig. 4. As the number of samples N increases, the absolute values of $z_N(\boldsymbol{x})$ in (13) get progressively smaller (see Sect. 4.1) and simply dividing $\hat{f}_N(\boldsymbol{x})$ by $\Delta\hat{F}$ as in (15) is not enough to make the exploration and exploitation contributions comparable. Thus, unless δ in (15) is dynamically varied in between iterations of `GLISp` [3], solving Problem (8) with $a_N(\boldsymbol{x})$ in (15) becomes similar to performing pure exploitation. This, in turn, can make `GLISp` [3] more prone to getting stuck on local minima of Problem (3) with no way of escaping (especially if the surrogate model is not expressive enough to capture the location of the global minimizer). Vice-versa, by performing min–max rescaling as proposed in (23), the exploration and exploitation contributions stay comparable throughout the whole optimization process and approximately assume the same range. For this reason, it is also more straightforward to define δ in (23) compared to the weight in (15).

From Propositions 2 and 3, we can immediately deduce the following results on the differentiability of $a_N(\boldsymbol{x})$ in (23).

Proposition 5 *The acquisition function $a_N(\boldsymbol{x})$ in (23) is differentiable everywhere provided that the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (9) is differentiable everywhere.*

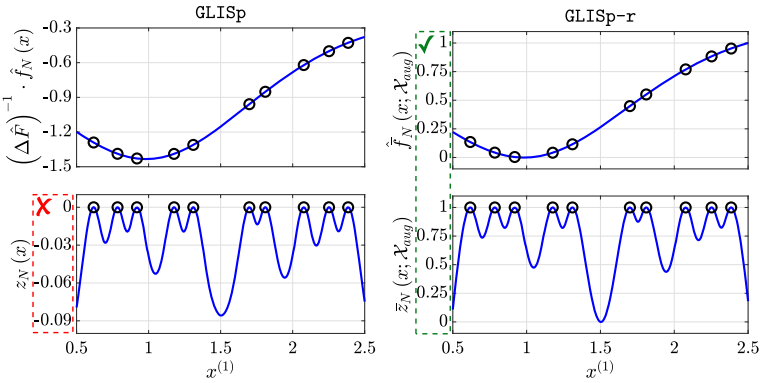


Fig. 4 Comparison between the terms of the acquisition functions in (15) (left, `GLISp` [3]) and in (23) (right, `GLISp-r`). The scoring function $f(\boldsymbol{x})$ is the `gramacy-lee` [14] function while $|\mathcal{X}| = N = 10$. For `GLISp-r`, the number of clusters used to build $\mathcal{X}_{aug}$ through Algorithm 1 is $K_{aug} = 5$. Notice how, for `GLISp` [3], the exploration function $z_N(\boldsymbol{x})$ is not comparable with the rescaled surrogate $\frac{\hat{f}_N(\boldsymbol{x})}{\Delta\hat{F}}$ since it assumes values that are one to two orders of magnitude lower

At each iteration of `GLISp-r` we find the next candidate for evaluation, i.e. $\boldsymbol{x}_{N+1}$, by solving Problem (8) with the acquisition function in (23). It is possible to use derivative-based optimization solvers since $a_N(\boldsymbol{x})$ in (23) is differentiable everywhere. In general, the acquisition function is *multimodal* and thus it is better to employ a global optimization procedure. Moreover, $a_N(\boldsymbol{x})$ is cheap to evaluate; therefore, we are not particularly concerned on its number of function evaluations.

4.3.1 Greedy δ-cycling

Many black-box optimization algorithms explicitly vary their emphasis on exploration and exploitation in between iterations. Just to cite a few:

- `Gutmann-RBF` [15] uses an acquisition function that is a measure of "bumpiness" of the RBF surrogate that depends upon a target value τ to aim for.
  The author suggests to cycle the values of τ between $\tau = \min_{\boldsymbol{x}\in\Omega}\hat{f}_N(\boldsymbol{x})$ (local search) and $\tau = -\infty$ (global search).
- The authors of `MSRS` [29], which uses an acquisition function that is similar to (23), propose to cycle between different values of δ as to prioritize exploration or exploitation more.
- In algorithm `SO-SA` [37], which is a revisitation of `MSRS` [29], the weight δ is chosen in a random fashion at each iteration. Moreover, the authors adopt a greedy strategy, i.e. δ is kept unaltered until it fails to find a significantly better solution.

In `GLISp` [3], the weight δ for $a_N(\boldsymbol{x})$ in (15) is kept constant throughout the whole optimization process. Also, defining some form of cycling for such hyper-parameter can be quite complex since the additive terms that compose the acquisition function are not always comparable. In this work, we propose a strategy that is in between that of `MSRS` [29] and `SO-SA` [37], which we refer to as *greedy δ-cycling*. We define a

sequence of $N_{cycle} \in \mathbb{N}$ weights to cycle:

$$\Delta_{cycle} = \langle \delta_0, \ldots, \delta_{N_{cycle}-1} \rangle. \tag{24}$$

Δ_{cycle} should contain values that are well spread within the $[0, 1]$ range as to properly alternate between local and global search. Greedy δ-cycling operates as follows. Suppose that, at iteration k of `GLISp-r`, we have at our disposal $|\mathcal{X}| = N$ samples and denote the trade-off weight δ in (23) as $\delta(k)$ to highlight the iteration number. Furthermore, assume $\delta(k) = \delta_j \in \Delta_{cycle}$, which is used to find the new candidate sample $\boldsymbol{x}_{N+1}$ at iteration k by solving Problem (8). Then, if $\boldsymbol{x}_{N+1} \succ \boldsymbol{x_{best}}(N)$ (i.e. there has been some improvement), the trade-off weight is kept unchanged, $\delta(k+1) = \delta(k) = \delta_j$. Otherwise, we cycle the values in Δ_{cycle}, obtaining $\delta(k+1) = \delta_{(j+1) \bmod N_{cycle}}$. Thus:

$$\delta(k+1) = \begin{cases} \delta_j & \text{if } \boldsymbol{x}_{N+1} \succ \boldsymbol{x_{best}}(N) \\ \delta_{(j+1) \bmod N_{cycle}} & \text{if } \boldsymbol{x_{best}}(N) \succsim \boldsymbol{x}_{N+1} \end{cases} \tag{25}$$

In Sect. 5, we will discuss the choice of the cycling sequence in (24) more in detail and also cover its relationship with the global convergence of `GLISp-r`.

5 Algorithm `GLISp-r` and convergence

Algorithm 2 describes each step of the `GLISp-r` procedure. As with any surrogate-based method, `GLISp-r` starts from an initial set of samples $\mathcal{X}$, $|\mathcal{X}| = N_{init} \in \mathbb{N}$, $N_{init} \geq 2$, generated by a *space-filling experimental design* [36], such as a Latin Hypercube Design (LHD) [24]. The sets $\mathcal{B}$ in (6) and $\mathcal{S}$ in (7), as well as the initial best candidate $\boldsymbol{x_{best}}(N_{init})$, are obtained by asking the decision-maker to compare the samples in $\mathcal{X}$ (5) as proposed in Algorithm 3, which prompts $M = N_{init} - 1$ queries. Once the initial sampling phase is concluded, the new candidate samples are obtained by solving Problem (8). The procedure is stopped once $|\mathcal{X}| = N_{max}$, where $N_{max} \in \mathbb{N}$ is a budget specified by the user. Overall, the decision-maker is queried $N_{max} - 1$ times.

`GLISp-r` follows the same scheme of `GLISp` [3] but, additionally, at each iteration, builds the augmented sample set $\mathcal{X}_{aug}$ using Algorithm 1. New candidate samples are found by minimizing the acquisition function in (23) instead of $a_N(\boldsymbol{x})$ in (15). Moreover, δ is cycled as proposed in Sect. 4.3.1. Similarly to `GLISp` [3], the shape parameter ϵ of the surrogate model in (9) is recalibrated at certain iterations of the algorithm, as specified by the set $\mathcal{K}_R \subseteq \{1, \ldots, N_{max} - N_{init}\}$, using grid-search Leave One Out Cross-Validation (LOOCV). In particular, at each iteration $k \in \mathcal{K}_R$, ϵ for $\hat{f}_N(\boldsymbol{x})$ in (9) is selected among a set $\mathcal{E}_{LOOCV}$ of possible shape parameters as the one whose corresponding surrogate preference function $\hat{\pi}_{\succsim_N}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in (10) classifies (out-of-sample) most of the preferences in $\mathcal{B}$ and S correctly [3]. Lastly, consistently with `GLISp` [3], Problem (3) is rescaled so that each decision variable assumes the $[-1, 1]$ range (at least inside Ω).

Algorithm 2 GLISp-r

Input: (i) Constraint set Ω of Problem (3); (ii) Number of initial samples $N_{init} \in \mathbb{N}$, $N_{init} \geq 2$; (iii) Budget $N_{max} \in \mathbb{N}$, $N_{max} > N_{init}$; (iv) Hyper-parameters for the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (9), i.e. shape parameter $\epsilon \in \mathbb{R}_{>0}$, radial function $\varphi(\cdot)$, regularization parameter $\lambda \in \mathbb{R}_{\geq 0}$ and tolerance $\sigma \in \mathbb{R}_{>0}$; (v) Cycling sequence Δ_{cycle} in (24) for the acquisition function $a_N(\boldsymbol{x})$ in (23); (vi) Number of clusters $K_{aug} \in \mathbb{N}$ for the augmented sample set $\mathcal{X}_{aug}$ generated by Algorithm 1; (vii) Possible shape parameters $\mathcal{E}_{LOOCV}$ for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (9); (viii) Set of indexes for the recalibration of the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (9), i.e. $\mathcal{K}_R \subseteq \{1, \ldots, N_{max} - N_{init}\}$.
Output: (i) Best sample obtained by the procedure $\boldsymbol{x_{best}}(N_{max})$.

Rescale Problem (3) as in GLISp [3]
Generate a set $\mathcal{X}$ in (5) of N_{init} starting points using a LHD [24]
Evaluate the samples in $\mathcal{X}$ by querying the decision-maker as in Algorithm 3, obtaining the sets $\mathcal{B}$ in (6) and $\mathcal{S}$ in (7), as well as the best candidate $\boldsymbol{x_{best}}(N_{init})$
Set $N = N_{init}$ and $M = |\mathcal{B}|$
Set $\delta = \delta_0 \in \Delta_{cycle}$ and $j = 0$
for $k = 1, 2, \ldots, N_{max} - N_{init}$ **do**
  if $k \in \mathcal{K}_R$ **then** recalibrate the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (9) as in GLISp [3]
  Build the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (9) from $\mathcal{X}$, $\mathcal{B}$ and $\mathcal{S}$ by solving Problem (11)
  Generate the augmented sample set $\mathcal{X}_{aug}$ through Algorithm 1
  Look for the next candidate sample $\boldsymbol{x}_{N+1}$ by solving Problem (8) with $a_N(\boldsymbol{x})$ in (23)
  Let the decision-maker express the preference $b_{M+1} = \pi_{\succsim}\left(\boldsymbol{x}_{N+1}, \boldsymbol{x_{best}}(N)\right)$
  if $b_{M+1} = -1$ (improvement, $\boldsymbol{x}_{N+1} \succ \boldsymbol{x_{best}}(N)$) **then**
    Set $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x}_{N+1}$
  else
    Keep $\boldsymbol{x_{best}}(N+1) = \boldsymbol{x_{best}}(N)$
    Set $\delta = \delta_{(j+1) \bmod N_{cycle}} \in \Delta_{cycle}$ (greedy δ-cycling) and $j = j + 1$
  end if
  Update the set of samples $\mathcal{X}$ and the preference information in the sets $\mathcal{B}$ and $\mathcal{S}$
  Set $N = N + 1$ and $M = M + 1$
end for

Algorithm 3 Initial queries for preference-based optimization

Input: (i) Initial set of samples $\mathcal{X}$, $|\mathcal{X}| = N_{init} \in \mathbb{N}$, $N_{init} \geq 2$, in (5).
Output: (i) Set of preferences $\mathcal{B}$ in (6); (ii) Mapping set $\mathcal{S}$ in (7); (iii) Initial best sample $\boldsymbol{x_{best}}(N_{init})$.

Initialize the best candidate as $\boldsymbol{x_{best}}(1) = \boldsymbol{x}_1$, $i_{best} = 1$
Initialize the sets $\mathcal{B}$ and $\mathcal{S}$: $\mathcal{B} = \emptyset$ and $\mathcal{S} = \emptyset$
for $i = 2$ to $|\mathcal{X}| = N_{init}$ **do**
  Let the decision-maker express a preference between $\boldsymbol{x_{best}}(i-1)$ and $\boldsymbol{x}_i$, obtaining $b = \pi\left(\boldsymbol{x_{best}}(i-1), \boldsymbol{x}_i\right)$
  Update the sets $\mathcal{B}$ and $\mathcal{S}$: $\mathcal{B} = \mathcal{B} \cup \{b\}$ and $\mathcal{S} = \mathcal{S} \cup \{(i_{best}, i)\}$
  if $b = 1$ (i.e. $\boldsymbol{x}_i \succ \boldsymbol{x_{best}}(i-1)$) **then**
    Update the best candidate, $\boldsymbol{x_{best}}(i) = \boldsymbol{x}_i$ and $i_{best} = i$
  else
    Keep the best candidate unaltered, $\boldsymbol{x_{best}}(i) = \boldsymbol{x_{best}}(i-1)$
  end if
end for

5.1 Global convergence of GLISp-r

Whenever we are dealing with any global optimization algorithm, it is possible to guarantee its convergence to the global minimizer of Problem (3) by checking if the conditions of the following Theorem hold.

Theorem 2 (Convergence of a global optimization algorithm [33]) *Let $\Omega \subset \mathbb{R}^n$ be a compact set. An algorithm converges to the global minimum of every continuous function $f : \mathbb{R}^n \to \mathbb{R}$ over Ω if and only if its sequence of iterates,*

$$\langle \boldsymbol{x}_i \rangle_{i \geq 1} = \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \rangle,$$

is dense in Ω.

In what follows, for the sake of clarity, we denote:

- $\mathcal{X}_\infty$ as the set containing all the elements of $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ (infinite sequence),
- $\mathcal{X}_k \subseteq \mathcal{X}_\infty$ as the set containing all the elements of $\langle \boldsymbol{x}_i \rangle_{i=1}^{k}$, which is a subsequence of $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ composed of its first $k \in \mathbb{N}$ entries.

To prove the convergence of `GLISp-r`, we also make use of the following Theorem, which gives us a sufficient condition that ensures the denseness of the sequence of iterates produced by any global optimization algorithm.

Theorem 3 (A sufficient condition for the denseness of $\mathcal{X}_\infty$ [28]) *Let Ω be a compact subset of $\mathbb{R}^n$ and let $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ be the sequence of iterates generated by an algorithm `A` (when run indefinitely). Suppose that there exists a strictly increasing sequence of positive integers $\langle i_t \rangle_{t \geq 1}$, $i_t \in \mathbb{N}$, such that $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ satisfies the following condition for some $\alpha \in (0, 1]$:*

$$\min_{1 \leq i \leq i_t - 1} \left\| \boldsymbol{x}_{i_t} - \boldsymbol{x}_i \right\|_2 \geq \alpha \cdot d_\Omega \left(\mathcal{X}_{i_t - 1} \right), \quad \forall t \in \mathbb{N}, \tag{26}$$

where:

$$d_\Omega \left(\mathcal{X}_{i_t - 1} \right) = \max_{\boldsymbol{x} \in \Omega} \min_{1 \leq i \leq i_t - 1} \left\| \boldsymbol{x} - \boldsymbol{x}_i \right\|_2 . \tag{27}$$

Then, $\mathcal{X}_\infty$ generated by `A` is dense in Ω.

The aforementioned Theorem has been used to prove the global convergence of `CORS` [28], a black-box optimization procedure. Clearly, if Ω is compact and (26) holds for some $\alpha \in (0, 1]$ (making $\mathcal{X}_\infty$ dense in Ω) then, due to Theorem 2, algorithm `A` converges to the global minimum of every continuous function $f(\boldsymbol{x})$ over Ω. For what concerns the preference-based framework, we need to ensure that the scoring function $f(\boldsymbol{x})$, which represents the preference relation $\succsim$ on Ω, is continuous. Theorem 1 gives us necessary conditions on $\succsim$ to achieve such property. Furthermore, Proposition 1 can be used to check the existence of a $\succsim$-maximum of Ω. The next Theorem addresses the global convergence of `GLISp-r` (Algorithm 2).

Theorem 4 (Convergence of `GLISp-r`) *Let $\Omega \subset \mathbb{R}^n$ be a compact set and $\succsim$ be a continuous preference relation on Ω of a rational (as in Definition 3) human decision-maker. Then, provided that $\exists \delta_j \in \Delta_{cycle}$ in (24) such that $\delta_j = 0$ and $N_{max} \to \infty$, `GLISp-r` converges to the global minimum of Problem (3) for any choice of its remaining hyper-parameters.*[5]

[5] Formally, we should also ensure that the surrogate model $\hat{f}_N(\boldsymbol{x})$ in (9) is continuous. However, that is the case for any of the radial basis functions reported in Sect. 3.1.

Proof Compactness of Ω, continuity of $\succsim$ on Ω and rationality of the decision-maker are conditions that ensure the existence of a solution for Problem (3) (cf. Theorem 1 and Proposition 1).

Consider the sequence of iterates $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ produced by Algorithm 2. The first $N_{init} \in \mathbb{N}$, $N_{init} \geq 2$, elements of $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ are obtained by the LHD [24]. Instead, each $\boldsymbol{x}_i \in \mathcal{X}_\infty$, $i > N_{init}$, is selected as the solution of Problem (8) with $a_N(\boldsymbol{x})$ in (23).

Now, suppose that δ in (23) is cycled regardless of the improvement that the new candidate samples might bring (non-greedy cycling). Denote the exploration–exploitation trade-off weight at iteration k of Algorithm 2 as $\delta(k)$ and assume that $\delta(k) = \delta_j \in \Delta_{cycle}$. Then, the cycling is performed as:

$$\delta(k+1) = \delta_{(j+1) \bmod N_{cycle}}, \quad \forall k \in \mathbb{N}, \tag{28}$$

instead of (25). Without loss of generality, suppose that Δ_{cycle} in (24) is defined as:

$$\delta_j \neq 0, \forall j = 0, \ldots, N_{cycle} - 2, \quad \text{and} \quad \delta_{N_{cycle}-1} = 0.$$

Then, every N_{cycle} iterations Algorithm 2 looks for a new candidate sample by minimizing the (min–max rescaled) IDW distance function in (13), regardless of the surrogate model in (9), see $a_N(\boldsymbol{x})$ in (23) and Problem (8). In practice, minimizing $\bar{z}_N\left(\boldsymbol{x}; \mathcal{X}_{aug}\right)$ over Ω is equivalent to solving $\arg\min_{\boldsymbol{x} \in \Omega} z_N(\boldsymbol{x})$ since scaling and shifting the IDW distance function does not change its minimizers [26]. We define the following strictly increasing sequence of positive integers:

$$\langle i_{t'} \rangle_{t' \geq 1} = \langle N_{init} + t' \cdot N_{cycle} \rangle_{t' \geq 1}, \tag{29}$$

which is such that:

$$\begin{aligned} \boldsymbol{x}_{i_{t'}} &= \arg\min_{\boldsymbol{x}} z_{i_{t'}-1}(\boldsymbol{x}), \quad \forall t' \in \mathbb{N} \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega. \end{aligned} \tag{30}$$

Now, recall from Proposition 4 that each $\boldsymbol{x}_i \in \mathcal{X}_{i_{t'}-1}$ is a global maximizer of Problem (30). Furthermore, $z_{i_{t'}-1}(\boldsymbol{x})$ in (13) is differentiable everywhere and thus continuous (see Proposition 3). Then, by the Extreme Value Theorem [1], Problem (30) admits at least a solution. Hence, we can conclude that:

$$\boldsymbol{x}_{i_{t'}} \notin \mathcal{X}_{i_{t'}-1} \implies \exists \tilde{\alpha}' \in \mathbb{R}_{>0} \text{ such that } \min_{1 \leq i \leq i_{t'}-1} \left\| \boldsymbol{x}_{i_{t'}} - \boldsymbol{x}_i \right\|_2 \geq \tilde{\alpha}', \quad \forall t' \in \mathbb{N}. \tag{31}$$

Clearly, due to how new candidate samples are sought (i.e. by minimizing some acquisition function over Ω), we have that (recall (27)):

$$\alpha' \cdot d_\Omega\left(\mathcal{X}_{i_{t'}-1}\right) \leq \min_{1 \leq i \leq i_{t'}-1} \left\| \boldsymbol{x}_{i_{t'}} - \boldsymbol{x}_i \right\|_2 \leq d_\Omega\left(\mathcal{X}_{i_{t'}-1}\right), \quad \forall t' \in \mathbb{N}, \tag{32}$$

for some $\alpha' \in (0, 1]$. Then, by combining (31) and (32), we get:

$$0 < \tilde{\alpha}' \leq d_{\Omega}\left(\mathcal{X}_{i_{t'}-1}\right), \quad \forall t' \in \mathbb{N}. \tag{33}$$

Therefore, $\exists \alpha' \in (0, 1]$ such that $\tilde{\alpha}' = \alpha' \cdot d_{\Omega}\left(\mathcal{X}_{i_{t'}-1}\right)$ which satisfies the condition (26) of Theorem 3, $\forall t' \in \mathbb{N}$. Thus, Algorithm 2 with δ cycled as in (28) produces a sequence of iterates that is dense in Ω.

Next, consider the greedy δ-cycling strategy proposed in Sect. 4.3.1 and for an arbitrary choice of Δ_{cycle} in (24). Let us examine the case in (25) when δ is kept unchanged from an iteration of Algorithm 2 to the other. Denote as $\langle i_{t''}\rangle_{t''=1}^{t''_{max}}$, $t''_{max} \in \mathbb{N}$, the sequence of indexes of those samples that improve upon the current best candidate, resulting in no change in the exploration–exploitation trade-off weight. We have that:

$$\boldsymbol{x}_{i_{t''}} \succ \boldsymbol{x}_{\boldsymbol{best}}\left(i_{t''}-1\right), \quad \forall t'' : 1 \leq t'' \leq t''_{max}.$$

Clearly, $\boldsymbol{x}_{i_{t''}} \notin \mathcal{X}_{i_{t''}-1}$ since it is strictly preferred to all the other samples in $\mathcal{X}_{i_{t''}-1}$. Thus, we could define a positive constant $\tilde{\alpha}'' \in \mathbb{R}_{>0}$ analogously to (31):

$$\begin{aligned} \boldsymbol{x}_{i_{t''}} \notin \mathcal{X}_{i_{t''}-1} \implies \exists \tilde{\alpha}'' \in \mathbb{R}_{>0} \text{ such that } \min_{1 \leq i \leq i_{t''}-1} \left\| \boldsymbol{x}_{i_{t''}} - \boldsymbol{x}_i \right\|_2 \geq \tilde{\alpha}'', \\ \forall t'' : 1 \leq t'', \leq t''_{max}. \end{aligned} \tag{34}$$

Finally, let us consider the greedy δ-cycling strategy in (25) as whole and assume, as in Theorem 4, that $\exists \delta_j \in \Delta_{cycle}$ in (24) such that $\delta_j = 0$. We can build a strictly increasing sequence of positive integers $\langle i_t \rangle_{t \geq 1}$ by merging:

- The elements of the sequence $\langle i_{t''}\rangle_{t''=1}^{t''_{max}}$, which are the indexes of those samples that improve upon the best candidates $\boldsymbol{x}_{\boldsymbol{best}}\left(i_{t''}-1\right), \forall t'' : 1 \leq t'' \leq t''_{max}$;
- The elements of the sequence $\langle i_{t'} \rangle_{t' \geq 1}$, which constitute the indexes of those samples found by solving the pure exploration problem in (30). Note that, unless Algorithm 2 always improves upon its current best candidate (in which case $\langle i_{t''}\rangle_{t''=1}^{t''_{max}}$ is actually infinite and hence $\mathcal{X}_{\infty}$ is dense in Ω), Problem (8) with $a_N(\boldsymbol{x})$ in (23) is solved using $\delta = 0$ infinitely often, although not necessarily every N_{cycle} iterations as in (29).

Hence, we can select a positive constant $\tilde{\alpha} \in \mathbb{R}_{>0}$ as $\tilde{\alpha} = \min\left\{\tilde{\alpha}', \tilde{\alpha}''\right\}$ which, analogously to (33), is such that:

$$0 < \tilde{\alpha} \leq d_{\Omega}\left(\mathcal{X}_{i_t-1}\right), \quad \forall t \in \mathbb{N}. \tag{35}$$

Therefore, $\exists \alpha \in (0, 1]$ such that $\tilde{\alpha} = \alpha \cdot d_{\Omega}\left(\mathcal{X}_{i_t-1}\right)$ which satisfies the condition (26) of Theorem 3, $\forall t \in \mathbb{N}$. Thus, `GLISp-r` with δ in (23) cycled following the greedy δ-cycling strategy in (25) converges to the global minimum of Problem (3). □

Most preference-based response surface methods, such as the ones in [3, 4, 6, 13], do not address their convergence to the global minimum of Problem (3). In this work,

we have shown that, by leveraging some results from the utility theory literature (see Sect. 2), we can find sufficient conditions on the preference relation of the human decision-maker ($\succsim$ on Ω) that guarantee the existence of a solution for Problem (3) and allow us to analyze the convergence of any preference-based procedure as we would in the global optimization framework.

We conclude this section with two Remarks on Theorem 4. The first deals with the importance of adding a zero entry inside Δ_{cycle} in (24), while the second addresses the selection of the cycling set.

Remark 4 The omission of a zero entry inside Δ_{cycle} in (24) does not necessarily preclude the global convergence of Algorithm 2 on all possible preference-based optimization problems. For example, if $f(\boldsymbol{x})$ is a constant function then, after we evaluate the first sample $\boldsymbol{x}_1$, any other point brings no improvement (i.e. we would have $\boldsymbol{x}_i \sim \boldsymbol{x}_1, \forall i > 1$). The caveat is that, if $\nexists \delta_j \in \Delta_{cycle}$ such that $\delta_j = 0$, then the sequence $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}$ for which (34) holds is likely to be finite (i.e. GLISp-r does not improve upon its current best candidate infinitely often). Moreover, differently from Problem (30), we have no guarantee that the solutions of:

$$\begin{aligned} \boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}} & a_N(\boldsymbol{x}), \quad \forall N : N \neq i_{t''} - 1, 1 \leq t'' \leq t''_{max} \\ \text{s.t.} \quad & \boldsymbol{x} \in \Omega, \end{aligned}$$

with $a_N(\boldsymbol{x})$ defined as in (23) and for $\delta \neq 0$, are not already present in $\mathcal{X}_N$.

Therefore, we cannot ensure the existence of a strictly increasing sequence of positive integers that is infinite and for which (26) holds. Instead, the result in Theorem 3 does not apply for $\langle i_{t''} \rangle_{t''=1}^{t''_{max}}$, since the sequence is finite. Consequently, Algorithm 2 does not necessarily produce a sequence of iterates $\langle \boldsymbol{x}_i \rangle_{i \geq 1}$ that is dense in Ω, preventing its convergence on some (but not all) preference-based optimization problems.

Remark 5 Theorem 4 guarantees that, under some hypotheses, GLISp-r converges to the global minimum of Problem (3), however it does not give any indication on its convergence rate. In particular, if Δ_{cycle} is actually $\langle 0 \rangle$, Algorithm 2 amounts to performing an exhaustive search without considering the information carried by the preferences in $\mathcal{B}$ (6) and $\mathcal{S}$ (7), which is quite inefficient [1]. Therefore, it is best to include some δ_j's in Δ_{cycle} that allow the surrogate model to be taken into consideration. For this reason, we suggest including terms that are well spread within the [0, 1] range, including a zero entry to ensure the result in Theorem 4. Intuitively, the rate of convergence will be dependent on how well $\hat{f}_N(\boldsymbol{x})$ in (9) approximates $f(\boldsymbol{x})$ as well as on the choice of Δ_{cycle} in (24).

6 Empirical results

In this section, we compare the performances of algorithms GLISp-r and GLISp [3] on a variety of benchmark bound-constrained global optimization problems taken from [2, 14, 18, 25]. Consistently with the preference-based optimization literature, we

stick to benchmark problems with less than $n = 10$ decision variables [3, 4, 6, 13]. We also consider the revisited version of the IDW distance function in (13) employed by C-GLISp [40], which is an extension of GLISp [3] proposed by the same authors.[6] In particular, in C-GLISp [40]:

$$z_N(\boldsymbol{x}) = \left(\frac{N}{N_{max}} - 1\right) \cdot \arctan\left(\frac{\sum_{i=1, i\neq i_{best}(N)}^{N} w_i\left(\boldsymbol{x_{best}}\left(N\right)\right)}{\sum_{i=1}^{N} w_i\left(\boldsymbol{x}\right)}\right) + \\ - \frac{N}{N_{max}} \cdot \arctan\left(\frac{1}{\sum_{i=1}^{N} w_i\left(\boldsymbol{x}\right)}\right), \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}, \tag{36}$$

while $z_N(\boldsymbol{x}) = 0, \forall \boldsymbol{x} \in \mathcal{X}$. In (36), $i_{best}\left(N\right) \in \mathbb{N}, 1 \leq i_{best}\left(N\right) \leq N$, represents the index of the best-found candidate when $|\mathcal{X}| = N$. In practice, $z_N\left(\boldsymbol{x}\right)$ in (36) improves the exploratory capabilities of GLISp [3] without the need to define an alternative acquisition function from the one in (15) (see [40]).

We point out that we could also consider the preferential Bayesian optimization algorithm in [6] as an additional competitor for GLISp-r. However, in [3], the authors show that GLISp outperforms the aforementioned method. As we will see in the next sections, GLISp-r exhibits convergence speeds that are similar to those of GLISp [3] and hence we have decided to omit the algorithm in [6] from our analysis. Moreover, after preliminary testing, the latter method was proven to not be on par w.r.t. the other competitors.

6.1 Experimental setup

All benchmark optimization problems have been solved on a machine with two Intel Xeon E5-2687W @3.00 GHz CPUs and 128 GB of RAM. GLISp-r has been implemented in MATLAB. Similarly, we have used the MATLAB code for GLISp provided in [3] (formally, version 2.4 of the software package) and the one for C-GLISp supplied in [40] (version 3.0 of the same code package). For all the procedures, Problem (8) has been solved using Particle Swarm Optimization (PSWARM). In particular, we have used its MATLAB implementation provided by [20, 34, 35].

To achieve a fair comparison, we have chosen the same hyper-parameters for both GLISp-r and GLISp/C-GLISp [3, 40], whenever possible. This applies, for example, to the shape parameter ϵ, which is initialized to $\epsilon = 1$, and the radial function $\varphi\left(\cdot\right)$, that is an inverse quadratic. Furthermore, the shape parameter for the surrogate model $\hat{f}_N\left(\boldsymbol{x}\right)$ in (9) is recalibrated using LOOCV (see GLISp [3]) at the iterations $\mathcal{K}_R = \{1, 50, 100\}$. Its possible values are $\mathcal{E}_{LOOCV} = \{0.1000,\ 0.1668,\ 0.2783,\ 0.4642,\ 0.7743,\ 1,\ 1.2915,\ 2.1544,\ 3.5938,\ 5.9948,\ 10\}$. The remaining hyper-parameters shared by GLISp/C-GLISp [3, 40] and GLISp-r are set to $\lambda = 10^{-6}$ and $\sigma = 10^{-2}$. Regarding GLISp-r, we have chosen $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0\rangle$,

[6] Note that, in this work, we use C-GLISp [40] only to test how the IDW distance function in (36) compares to the other formulation in (13). However, C-GLISp [40] has been developed mainly to extend GLISp [3] in order to handle black-box constraints. The different definition of $z_N\left(\boldsymbol{x}\right)$ is only a minor detail of such paper.

where we have included a zero term to comply with the convergence result in Theorem 4. The reasoning behind this cycling sequence is that, after the initial sampling phase, we give priority to the surrogate as to drive the algorithm towards more promising regions of Ω, for example where local minima are located. In practice, if $f(\boldsymbol{x})$ is a function that can be approximated well by $\hat{f}_N(\boldsymbol{x})$ with little data, starting with a δ in (23) that is close to 1 might lead the procedure to converge quite faster. If that is not the case, the remaining terms contained inside Δ_{cycle} promote the exploration of other zones of the constraint set, either dependently or independently from $\hat{f}_N(\boldsymbol{x})$. For the sake of completeness, we also analyze the performances of GLISp-r when equipped with two particular cycling sequences: $\Delta_{cycle} = \langle 0.95 \rangle$ ("pure" exploitation) and $\Delta_{cycle} = \langle 0 \rangle$ (pure exploration). Concerning GLISp/C-GLISp [3, 40], which use $a_N(\boldsymbol{x})$ in (15), we set $\delta = 2$, as proposed by the authors in [3]. Lastly, for GLISp-r, we select $K_{aug} = 5$ for all optimization problems since, empirically, after several preliminary experiments, it has proven to be good enough to rescale $z_N(\boldsymbol{x})$ in (13) and $\hat{f}_N(\boldsymbol{x})$ in (9) in most cases (see for example Fig. 4).

We run the procedures using a fixed budget of $N_{max} = 200$ samples and solve each benchmark optimization problem $N_{trial} = 100$ times, starting from $N_{init} = 4 \cdot n$ points generated by LHDs [24] with different random seeds. To ensure fairness of comparison, all the algorithms are started from the same samples.

For the sake of clarity, we point out that the preferences between the couples of samples are expressed using the preference function in (4), where $f(\boldsymbol{x})$ is the corresponding cost function for the considered benchmark optimization problem. Therefore, the preferences are always expressed consistently, allowing us to compare GLISp-r and GLISp/C-GLISp [3, 40] properly. That would not necessarily be the case if a human decision-maker was involved.

6.2 Results

We compare the performances of GLISp-r and GLISp/C-GLISp [3, 40] on each benchmark optimization problem by means of *convergence plots* and *data profiles* [1]. Convergence plots depict the median, best and worst case performances over the N_{trial} instances and with respect to the cost function values achieved by $\boldsymbol{x_{best}}(N)$, as N increases. Data profiles are one of the most popular tools for assessing efficiency and robustness of global optimization methods: efficient surrogate-based methods exhibit steep slopes (i.e. fast convergences speeds), while robust algorithms are able to solve more (instances of) problems within the budget N_{max}. In general, no method is both efficient and robust: a trade-off must be made [32]. Typically, data profiles are used to visualize the performances of several algorithms on multiple benchmark optimization problems simultaneously. However, due to the stochastic nature of LHDs [24], here we will also depict the data profiles for GLISp-r, GLISp [3] and C-GLISp [40] on each benchmark optimization problem on its own, highlighting how the algorithms behave when started from different samples. In practice, data profiles show, for $1 \leq N \leq N_{max}$, how many among the N_{trial} instances of one (or several) benchmark optimization problem(s) have been solved to a prescribed *accuracy*, defined as [1]:

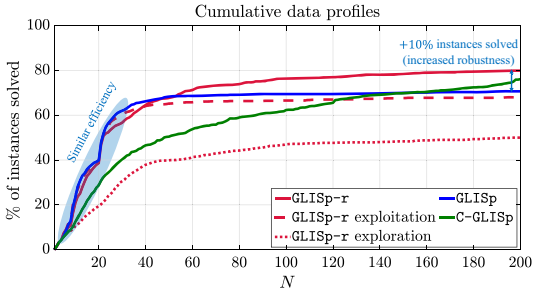


Fig. 5 Cumulative data profiles ($acc\,(N) > 95\%$) of the considered preference-based optimization algorithms. `GLISp-r` is depicted in red ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0\rangle$ continuous line, $\Delta_{cycle} = \langle 0.95\rangle$ dashed line, $\Delta_{cycle} = \langle 0\rangle$ dotted line), `GLISp` [3] in blue and `C-GLISp` [40] in green (color figure online)

$$acc\,(N) = \frac{f\,(\boldsymbol{x}_{\boldsymbol{best}}\,(N)) - f\,(\boldsymbol{x}_1)}{f^* - f\,(\boldsymbol{x}_1)}, \tag{37}$$

where $f^* = \min_{\boldsymbol{x}\in\Omega} f\,(\boldsymbol{x})$. In particular, here we consider a benchmark optimization problem to be solved by some algorithm when $acc\,(N) > t$, $t = 0.95$. In what follows, the results achieved by `GLISp-r` equipped with $\Delta_{cycle} = \langle 0.95\rangle$ and $\Delta_{cycle} = \langle 0\rangle$ are reported only in the data profiles (and not in the convergence plots), to make the graphs more readable.

Figure 5 shows the cumulative data profiles of `GLISp-r`, `GLISp` [3] and `C-GLISp` [40], which result from considering all the benchmark optimization problems simultaneously. Instead, Figs. 6 and 7 depict the convergence plots and the data profiles achieved by the considered algorithms on each benchmark. In Table 1 we report the number of samples required to reach the accuracy $t = 0.95$, i.e.:

$$N_{acc>t} = \min_{1\le N\le N_{max}} N \quad \text{such that } acc\,(N) > t. \tag{38}$$

In practice, given that each benchmark optimization problem is solved multiple times, $N_{acc>t}$ in Table 1 is assessed median-wise (over the N_{trial} instances), giving us an indication on the efficiency of each method. In the same table, we also report the percentages of instances of problems solved by `GLISp-r`, `GLISp` [3] and `C-GLISp` [40] (which is an indicator of robustness) and the average execution times of each algorithm.

From our experiments, we gather that:

- `GLISp-r` ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0\rangle$) *can be notably more robust than* `GLISp` [3] *without excessively compromising its convergence speed*. On several occasions, the latter algorithm gets stuck on local minima of the benchmark optimization problems. That is particularly evident on the `bemporad` [2] and `gramacy-lee` [14] benchmarks, in which cases `GLISp` [3] solves, respectively, only 70% and 31% of the N_{trial} instances. Instead, `GLISp-r` ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0\rangle$) is able to solve both of them to the prescribed accuracy. In practice, `GLISp` [3] shines when exploitation is better suited for the benchmark optimization problem at hand. That is particularly relevant for the `bukin 6` [18] problem, on which both `GLISp`

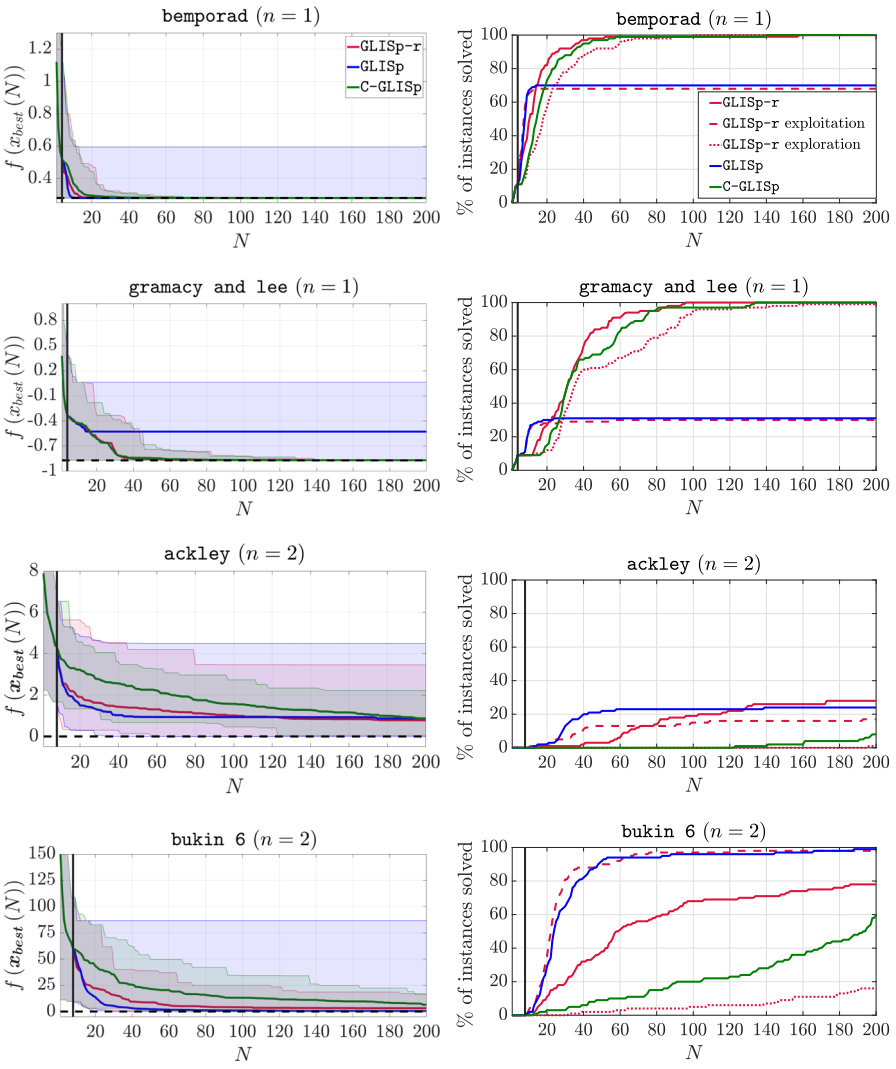


Fig. 6 Performances achieved by the considered preference-based optimization algorithms on the benchmark optimization problems: convergence plots on the left and data profiles ($acc(N) > 95\%$) on the right. GLISp-r (with $\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) is depicted in red, GLISp [3] in blue and C-GLISp [40] in green. The dashed black-line in the convergence plots represents f^*. We also show the number of initial samples, N_{init}, with a black vertical line. The results obtained by GLISp-r with $\Delta_{cycle} = \langle 0.95 \rangle$ (dashed red line) and GLISp-r with $\Delta_{cycle} = \langle 0 \rangle$ (dotted red line) are shown only in the data profiles (color figure online)

[3] and GLISp-r with $\Delta_{cycle} = \langle 0.95 \rangle$ ("pure" exploitation) perform quite well. Lastly, C-GLISp [40] is as robust as GLISp-r ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) but is the least efficient among the analyzed procedures.

- The pure exploration strategy (i.e. GLISp-r with $\Delta_{cycle} = \langle 0 \rangle$) performs poorly, even for $n = 2$. When $n = 5$, it is unable solve any problem (in fact, the data profiles

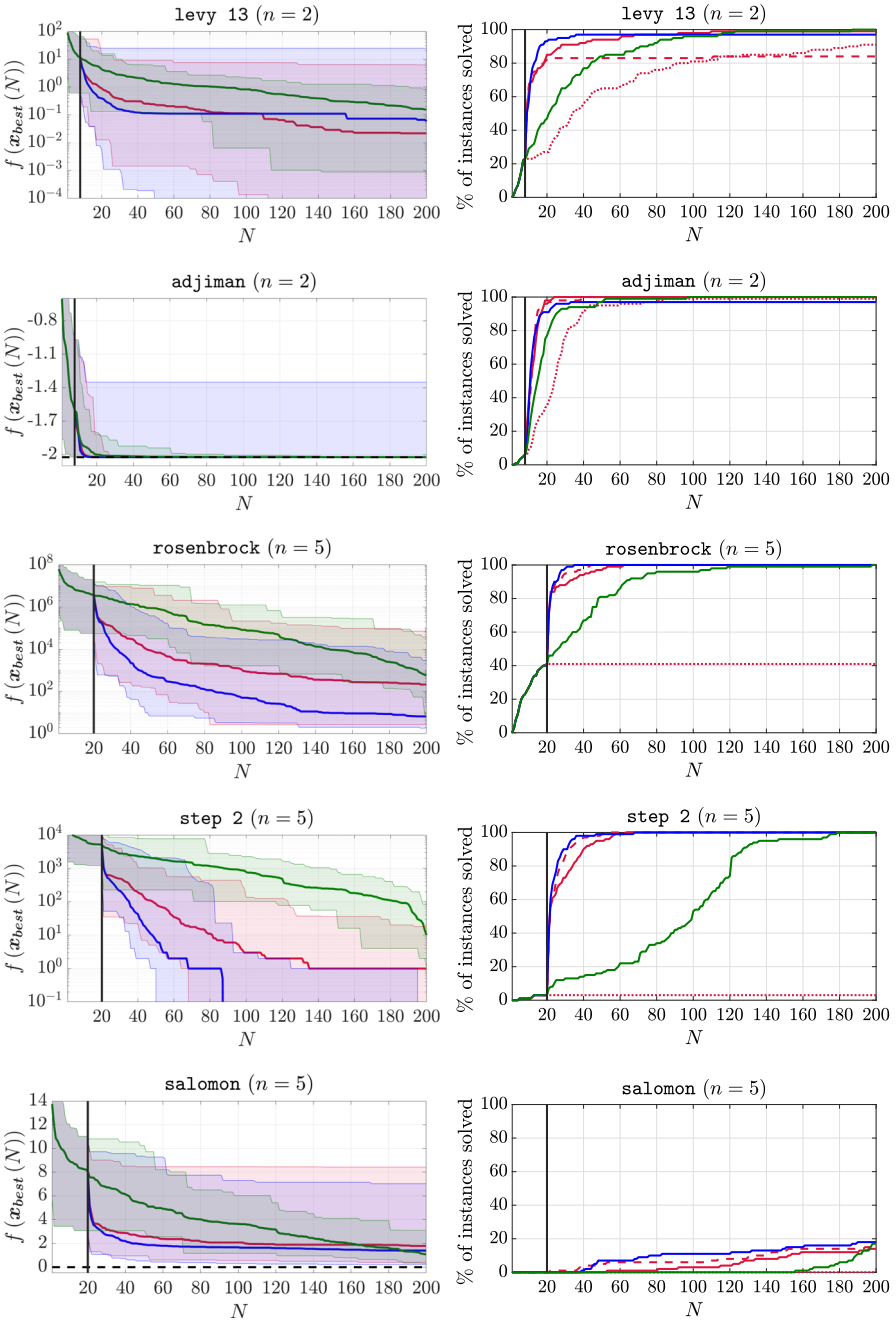


Fig. 7 Figure 6 cont'd

stay flat after the initial sampling phase). Only for $n = 1$ the pure exploration strategy is quite robust and relatively efficient.

- Vice-versa, a "pure" exploitatory approach (i.e. GLISp-r with $\Delta_{cycle} = \langle 0.95 \rangle$), although not necessarily globally convergent (see Theorem 4), can actually be successful on some benchmark optimization problems. Often, such strategy exhibits a slightly lower $N_{acc>t}$ (median-wise) than the other procedures, see Table 1. Notably, the data profiles of GLISp-r ($\Delta_{cycle} = \langle 0.95 \rangle$) can be quite similar to those of GLISp [3]. Therefore, we could say that GLISp [3] has limited exploratory capabilities.
- The main disadvantage of GLISp-r, compared to GLISp [3] and C-GLISp [40], is the increased computational time, as reported in Table 1. That is due to the computational overhead of Algorithm 1, which generates the augmented sample set $\mathcal{X}_{aug}$ for the proposed procedure by performing K-means clustering. On average, GLISp-r ($\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$) is 31% slower than GLISp [3] and 72% slower than C-GLISp [40]. However, we point out that these overheads are practically negligible when the queries to the decision-maker involve running computer simulations or performing real-world experiments which, contrary to the considered benchmark optimization problems, can take from a few minutes up to several hours. This is a common assumption made by surrogate-based methods: at each iteration, the most time-consuming operation is the query to the decision-maker (or, in the context of black-box optimization, the measure of the cost function [19, 36]).

7 Conclusions

In this work, we introduced the preference-based optimization problem from a utility theory perspective. Then, we extended algorithm GLISp [3], giving rise GLISp-r. In particular, we have addressed the shortcomings of $a_N(\boldsymbol{x})$ in (15), defined a new acquisition function and proposed the greedy δ-cycling strategy, which dynamically varies the exploration–exploitation weight δ in (23). Furthermore, we have proven the global convergence of GLISp-r, which is strictly related to the choice of the cycling sequence Δ_{cycle}. To the best of our knowledge, GLISp-r is the first preference-based surrogate-based method with a formal proof of convergence.

Compared to the original method, the proposed extension is less likely to get stuck on local minima of the scoring function and proves to be more robust on several benchmark optimization problems without particularly compromising its convergence speed. Moreover, we have also considered algorithm C-GLISp [40], which improves the exploratory capabilities of GLISp [3] by employing a different exploration function. In our experiments, we have observed that, even though C-GLISp [40] is as robust as GLISp-r, it often exhibits slower convergence rates compared to GLISp [3] and the proposed extension.

Further research is devoted to extending GLISp-r in order to handle black-box constraints, which involve functions whose analytical formulations are not available. One possibility is to follow the same reasoning behind C-GLISp [40], which does so by adding a term to the acquisition function that penalizes the exploration in those

Table 1 Comparison between the considered preference-based optimization algorithms

Benchmark	n	GLISp [3]			C-GLISp [40]			GLISp-r								
								$\Delta_{cycle} = \langle 0.95, 0.7, 0.35, 0 \rangle$			$\Delta_{cycle} = \langle 0.95 \rangle$			$\Delta_{cycle} = \langle 0 \rangle$		
		(a)	(b) (%)	(c)	(a)	(b) (%)	(c)	(a)	(b) (%)	(c)	(a)	(b) (%)	(c)	(a)	(b) (%)	(c)
bemporad [2]	1	**8**	70	79.8	15	**100**	**57.5**	11	**100**	79.4	**8**	68	109.7	18	**100**	71.4
gramacy-lee [14]	1	n.r.	31	74.4	32	**100**	**58.1**	**31**	**100**	67.5	n.r.	30	90.8	36	99	63.8
ackley [18]	2	n.r.	24	92.1	n.r.	8	**70.3**	n.r.	**28**	127.9	n.r.	17	122.4	n.r.	1	121.1
bukin 6 [18]	2	24	**99**	84.5	193	60	**73.3**	58	78	115.0	**23**	98	114.6	n.r.	16	123.6
levy 13 [25]	2	**9**	97	82.0	22	**100**	**69.3**	**9**	99	126.8	**9**	84	118.2	34	91	120.6
adjiman [18]	2	**11**	97	105.4	15	**100**	**71.6**	12	**100**	146.1	12	**100**	124.0	24	99	117.5
rosenbrock [18]	5	**21**	**100**	133.3	26	**100**	107.1	**21**	**100**	190.2	**21**	**100**	222.3	n.r.	41	**100.2**
step 2 [18]	5	**22**	**100**	148.1	100	**100**	104.2	**22**	**100**	190.6	**22**	**100**	216.7	n.r.	3	**98.9**
salomon [18]	5	n.r.	**18**	137.7	n.r.	17	103.5	n.r.	14	183.4	n.r.	16	219.1	n.r.	0	**100.8**
Average		**15.8**	70.7	104.1	57.6	76.1	**79.4**	23.4	**79.8**	136.3	**15.8**	68.1	148.6	28.0	56.3	102.0

Several indicators are taken into account: (a) *efficiency indicator*, i.e. number of samples required to solve the benchmark optimization problems to an accuracy of 0.95 (median-wise), as defined in (38), (b) *robustness indicator*, i.e. percentage of (instances of) problems solved and (c) average execution times (in seconds). The acronym n.r. stands for "not reached". The best results are highlighted in bold font. We also report the average performances over all the considered benchmark optimization problems. Note that the averages for indicator (a) do not consider those benchmarks for which the algorithms did not reach the desired accuracy

regions of Ω where the black-box constraints are likely to be violated. Additionally, a compelling line of research concerns preference-based optimization problems in the case where the human decision-maker is "irrational" (in a sense that some of the properties in Definition 3, typically completeness, do not hold). From a practical perspective, when the preference relation $\succsim$ on Ω of the DM is not complete, we would need a surrogate model that is able to handle the answer "I do not know" when the decision-maker cannot decide which, among two calibrations, he/she prefers.

Funding Open access funding provided by Università degli studi di Bergamo within the CRUI-CARE Agreement.

Data availibility The benchmark optimization problems considered in Sect. 6 are reported in [2, 14, 18, 25]. The MATLAB code for algorithms `GLISp` [3] and `C-GLISp` [40] is provided in the corresponding papers. The MATLAB code for the proposed method, `GLISp-r`, is supplied in the supplementary material of this paper.

Declarations

Conflict of interest All the authors declare that they have no conflict of interest.

Appendix A: Additional proofs

Proof of Lemma 1 The IDW distance function in (13) is differentiable at any $\boldsymbol{x} \in \mathbb{R}^n$ (see Proposition 3); thus, its gradient $\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x})$ is defined everywhere and can be computed by repeatedly applying the chain rule.

Consider the case $\boldsymbol{x} \in \mathbb{R}^n \backslash \mathcal{X}$. First of all, it is easy to prove that the IDW function $w_i(\boldsymbol{x})$ in (12) is differentiable at any $\boldsymbol{x} \in \mathbb{R}^n \backslash \{\boldsymbol{x}_i\}$. That is because the squared Euclidean norm $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2$ is differentiable everywhere and also:

$$\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 \neq 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{x}_i\}.$$

Therefore, due to the reciprocal rule, the IDW function is differentiable at any $\boldsymbol{x} \in \mathbb{R}^n \backslash \{\boldsymbol{x}_i\}$. In order to compute the gradient of $w_i(\boldsymbol{x})$ in (12), recall that the gradient of the squared Euclidean norm is equal to:

$$\nabla_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 = 2 \cdot (\boldsymbol{x} - \boldsymbol{x}_i), \quad \forall \boldsymbol{x} \in \mathbb{R}^n. \tag{A1}$$

Using (A1), it is easy to prove that:

$$\nabla_{\boldsymbol{x}} w_i(\boldsymbol{x}) = -2 \cdot \frac{\boldsymbol{x} - \boldsymbol{x}_i}{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^4} = -2 \cdot (\boldsymbol{x} - \boldsymbol{x}_i) \cdot w_i(\boldsymbol{x})^2, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{x}_i\}. \tag{A2}$$

Now, let us consider the argument of the arctan $(\cdot)$ function in $z_N(\boldsymbol{x})$, which is (see (13)):

$$h(\boldsymbol{x}) = \frac{1}{\sum_{i=1}^N w_i(\boldsymbol{x})}, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}.$$

We can find the gradient of $h(\boldsymbol{x})$ by applying the chain rule in combination with (A2):

$$\nabla_{\boldsymbol{x}} h(\boldsymbol{x}) = 2 \cdot \frac{\sum_{i=1}^N (\boldsymbol{x} - \boldsymbol{x}_i) \cdot w_i(\boldsymbol{x})^2}{\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2}, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}. \tag{A3}$$

Finally, using (A3) and applying the chain rule one last time, we can compute the gradient of the IDW distance function in (13):

$$\begin{aligned} \nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}) &= \frac{d}{dt}\left[-\frac{2}{\pi} \cdot \arctan(t)\right]\Big|_{t=h(\boldsymbol{x})} \cdot \nabla_{\boldsymbol{x}} h(\boldsymbol{x}) \\ &= -\frac{4}{\pi} \cdot \frac{1}{1 + \left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^{-2}} \cdot \frac{\sum_{i=1}^N (\boldsymbol{x} - \boldsymbol{x}_i) \cdot w_i(\boldsymbol{x})^2}{\left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2} \\ &= -\frac{4}{\pi} \cdot \frac{\sum_{i=1}^N (\boldsymbol{x} - \boldsymbol{x}_i) \cdot w_i(\boldsymbol{x})^2}{1 + \left[\sum_{i=1}^N w_i(\boldsymbol{x})\right]^2}, \quad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \mathcal{X}. \end{aligned} \tag{A4}$$

Now, let us consider the case $\boldsymbol{x}_i \in \mathcal{X}$. In [2], the authors have proven that all the partial derivatives of $z_N(\boldsymbol{x})$ in (13) are zero at each $\boldsymbol{x}_i \in \mathcal{X}$, i.e.

$$\nabla_{\boldsymbol{x}} z_N(\boldsymbol{x}_i) = \mathbf{0}_n, \quad \forall \boldsymbol{x}_i \in \mathcal{X}. \tag{A5}$$

Lastly, combining (A4) and (A5), we obtain the expression for the gradient of the IDW distance function $\forall \boldsymbol{x} \in \mathbb{R}^n$, as reported in (14). □

References

1. Audet, C., Hare, W.: Derivative-Free and Blackbox Optimization. Springer, Berlin (2017). (**ISBN 9783319886800**)
2. Bemporad, A.: Global optimization via inverse distance weighting and radial basis functions. Comput. Optim. Appl. **77**(2), 571–595 (2020). https://doi.org/10.1007/s10589-020-00215-w. (**ISSN 1573-2894**)
3. Bemporad, A., Piga, D.: Global optimization based on active preference learning with radial basis functions. Mach. Learn. **110**(2), 417–448 (2021). https://doi.org/10.1007/s10994-020-05935-y. (**ISSN 1573-0565**)

4. Benavoli, A., Azzimonti, D., Piga, D.: Preferential Bayesian optimisation with skew gaussian processes. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1842–1850 (2021). https://doi.org/10.1145/3449726.3463128
5. Bishop, C.M., Nasrabadi, N.M.: Pattern Recognition and Machine Learning. Springer, Berlin (2006)
6. Brochu, E., De Freitas, N., Ghosh, A.: Active preference learning with discrete choice data. In: Advances in Neural Information Processing Systems 20 (NIPS 2007), pp. 409–416 (2007). https://papers.nips.cc/paper/2007/hash/b6a1085a27ab7bff7550f8a3bd017df8-Abstract.html
7. Chu, W., Ghahramani, Z.: Preference learning with gaussian processes. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 137–144 (2005). https://doi.org/10.1145/1102351.1102369
8. Debreu, G.: Theory of value: An Axiomatic Analysis of Economic Equilibrium, Vol. 17. Yale University Press (1971) (**ISBN 0300015593**)
9. Fasshauer, G.E.: Meshfree Approximation Methods with MATLAB, vol. 6. World Scientific, Singapore (2007). (**ISBN 9789812706348**)
10. Feldman, A. M., Serrano, R.: Welfare Economics and Social Choice Theory. Springer Science & Business Media (2006). (**ISBN 9780387293677**)
11. Fornberg, B., Flyer, N.: A Primer on Radial Basis Functions with Applications to the Geosciences. SIAM, Philadelphia (2015). (**ISBN 9781611974027**)
12. Fürnkranz, J., Hüllermeier, E.: Preference Learning and Ranking by Pairwise Comparison. Springer, Berlin (2010). (**ISBN 9783642141249**)
13. González, J., Dai, Z., Damianou, A., Lawrence, N. D.: Preferential Bayesian optimization. In: International Conference on Machine Learning, pp. 1282–1291 (2017). https://assets.amazon.science/da/1e/24ad7c354431bd0c2f64a6049269/preferential-bayesian-optimization.pdf
14. Gramacy, R.B., Lee, H.K.H.: Cases for the nugget in modeling computer experiments. Stat. Comput. **22**(3), 713–722 (2012). https://doi.org/10.1007/s11222-010-9224-x
15. Gutmann, H.-M.: A radial basis function method for global optimization. J. Glob. Optim. **19**(3), 201–227 (2001). https://doi.org/10.1023/A:1011255519438
16. Han, J., Pei, J., Kamber, M.: Data Mining: Concepts and Techniques. Elsevier, Amsterdam (2011). (**ISBN 9789380931913**)
17. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, vol. 2. Springer, Berlin (2009). (**ISBN 9780387848570**)
18. Jamil, M., Yang, X.-S.: A literature survey of benchmark functions for global optimisation problems. Int. J. Math. Modell. Numer. Optim. **4**(2), 150–194 (2013). https://doi.org/10.1504/IJMMNO.2013.055204
19. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Glob. Optim. **21**(4), 345–383 (2001). https://doi.org/10.1023/A:1012771025575
20. Le, H.A., Thi, A.I., Vaz, F., Vicente, L.N.: Optimizing radial basis functions by DC programming and its use in direct search for global derivative-free optimization. TOP **20**(1), 190–214 (2012). https://doi.org/10.1007/s11750-011-0193-9
21. Liu, G.P., Yang, J.B., Whidborn, J.F.: Multiobjective Optimisation and Control. Research Studies Press Ltd, Baldock (2002). (**ISBN 9780863802645**)
22. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982). https://doi.org/10.1109/TIT.1982.1056489
23. Martí, R., Lozano, J. A., Mendiburu, A., Hernando, L.: Multi-start methods. In: Handbook of Heuristics, pp. 155–175 (2018). https://doi.org/10.1007/0-306-48056-5_12
24. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **42**(1), 55–61 (2000). https://doi.org/10.1080/00401706.2000.10485979
25. Mishra, S. K.: Some new test functions for global optimization and performance of repulsive particle swarm method. Available at SSRN 926132. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=926132 (2006)
26. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, Berlin (1999). (**ISBN 0387987932**)
27. Ok, E.A.: Real Analysis with Economic Applications. Princeton University Press, Princeton (2011). (**ISBN 9780691117683**)
28. 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). https://doi.org/10.1007/s10898-004-0570-0

29. Regis, R.G., Shoemaker, C.A.: A stochastic radial basis function method for the global optimization of expensive functions. INFORMS J. Comput. **19**(4), 497–509 (2007). https://doi.org/10.1287/ijoc.1060.0182
30. Roveda, L., Maggioni, B., Marescotti, E., Shahid, A.A., Zanchettin, A.M., Bemporad, A., Piga, D.: Pairwise preferences-based optimization of a path-based velocity planner in robotic sealing tasks. IEEE Robot. Autom. Lett. **6**(4), 6632–6639 (2021). https://doi.org/10.1109/LRA.2021.3094479
31. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In Proceedings of the 1968 23rd ACM National Conference, pp. 517–524 (1968). https://doi.org/10.1145/800186.810616
32. Thanedar, P.B., Arora, J.S., Li, G.Y., Lin, T.C.: Robustness, generality and efficiency of optimization algorithms for practical applications. Struct. Optim. **2**(4), 203–212 (1990). https://doi.org/10.1007/BF01748225
33. Torn, A., Zilinskas, A.: Global optimization. In: Lecture Notes in Computer Science (1989). (**ISBN 9783540508717**)
34. 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). https://doi.org/10.1007/s10898-007-9133-5
35. Vaz, A.I.F., Vicente, L.N.: PSwarm: a hybrid solver for linearly constrained global derivative-free optimization. Optim. Methods Softw. **24**(4–5), 669–685 (2009). https://doi.org/10.1080/10556780902909948
36. Vu, K.K., d'Ambrosio, C., Hamadi, Y., Liberti, L.: Surrogate-based methods for black-box optimization. Int. Trans. Oper. Res. **24**(3), 393–424 (2017). https://doi.org/10.1111/itor.12292
37. Wang, Y., Shoemaker, C. A.: A general stochastic algorithmic framework for minimizing expensive black box objective functions based on surrogate models and sensitivity analysis. arXiv:1410.6271 (2014). https://doi.org/10.48550/arXiv.1410.6271
38. Williams, C.K., Rasmussen, C.E.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2005)
39. Zhu, M., Bemporad, A., Piga, D.: Preference-based MPC calibration. arXiv:2003.11294 (2020). https://doi.org/10.48550/arXiv.2003.11294
40. Zhu, M., Piga, D., Bemporad, A.: C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration. IEEE Trans. Control Syst. Technol. (2021). https://doi.org/10.1109/TCST.2021.3136711