# D-GLISp: A Distributed Algorithm for Preference-Based Black-Box Optimization in Multi-Agent Systems

Loris Cannelli, Mengjia Zhu, Melanie Schranz, Alberto Bemporad, Dario Piga

***Abstract*—This paper proposes *D-GLISp*, a novel algorithm to solve black-box optimization problems over a multi-agent network, where agents cooperate to reach consensus on a decision that satisfies everyone to a reasonable extent. In this setting, each agent aims to optimize an unknown objective function that cannot be evaluated analytically or numerically, but only implicitly and qualitatively through pairwise preferences over candidate solutions. A key challenge lies in the distributed nature of the problem, where agents are not allowed to share their local data or preferences.**

**To address this, D-GLISp extends the GLISp framework to the distributed setting by combining local surrogate modeling based on preference data with an exploration mechanism that promotes coverage of the decision space. A distributed optimization strategy allows agents to collectively select new candidate solutions and converge toward a global consensus without exchanging sensitive information. We formally define the concept of global optimality in the context of distributed preferences and demonstrate the effectiveness of the proposed algorithm through simulation studies involving distributed multi-agent calibration of the parameters of model predictive controllers.**

## I. Introduction

Black-box optimization problems arise in a wide range of engineering and scientific domains, where the objective function to be optimized does not have an analytical closed-form expression and can only be evaluated through experiments or queries. This is common, for example, in engineering design, hyperparameter tuning of machine learning models, and physical model calibration, where each function evaluation may be costly, noisy, or time-consuming [1], [2].

The challenge of black-box optimization becomes even more pronounced when the objective function cannot be quantitatively measured, even through direct queries. In some of such scenarios, optimization can be guided by indirect and qualitative information, such as pairwise preferences between candidate solutions expressed by a human. This setting, known as *preference-based black-box optimization* [3], is particularly relevant in multi-objective design problems, where it is often difficult to quantify the relative importance of competing criteria. In these cases, it may be easier for a user to express a qualitative preference between two candidate solutions rather than assign explicit numerical scores. Applications of preference-based black-box optimization are increasingly diverse. For example, in user interface design, users may find it more natural to indicate which of two layouts they prefer rather than rate each one on an absolute scale [4]. In materials or molecule discovery, experimentalists may be able to compare two samples but not to provide precise quantitative evaluations [5]. In recommender systems, user preferences are often elicited as pairwise comparisons [6]. In robotics, movement trajectories can be iteratively optimized through simple comparisons between different behaviors [7]. Similarly, in control systems, controller parameters can be tuned by human operators comparing how different strategies produce more or less desirable system responses [8].

These scenarios motivate the development of algorithms that, by modeling the underlying and unknown utility function through statistical or machine learning methods, can efficiently optimize black-box objectives using only preference information. The most known algorithms are *Preferential Bayesian Optimization* [3] and their variants [9], [10], [11], which extends Bayesian optimization to settings with qualitative feedback by placing Gaussian process priors over latent utility functions and updating beliefs using observed preferences. Another promising approach is *GLISp* (**GL**obal optimization based on **I**nverse distance weighting and radial basis function **S**urrogates with **p**references) [12], which avoids probabilistic modeling in favor of simpler surrogate-based strategies and distance-based exploration terms.

The approaches mentioned above assume that preferences are expressed by a *single* agent. However, there are scenarios in which black-box optimization must be performed by collecting pairwise comparisons by *multiple* agents in a *distributed* setting, where agents collaborate to find consensus on an optimal solution without being able to share data with others due, for example, to communication constraints or privacy concerns. Examples include the distributed calibration of machine tools or control systems, where multiple calibrators work in parallel, each using its own qualitative criteria to express preferences over configurations, and must collectively agree on a final setting that satisfies everyone to a reasonable extent. Distributed black-box optimization is a compelling and increasingly relevant problem in other domains too, such as edge-fog-cloud computing [13], federated robotics [14], and industrial automation, where data is decentralized and full knowledge sharing is constrained by bandwidth, privacy, or

Loris Cannelli and Dario Piga are with the IDSIA Dalle Molle Institute for Artificial Intelligence, SUPSI, Lugano, Switzerland (e-mail:{loris.cannelli, dario.piga}@supsi.ch).

Mengjia Zhu is with the University of Liverpool, Liverpool, United Kingdom (e-mail: mengjia.zhu@liverpool.ac.uk).

Melanie Schranz is with Lakeside Labs GmbH, Klagenfurt, Austria (e-mail: schranz@lakeside-labs.com).

Alberto Bemporad is with IMT School for Advanced Studies, Lucca, Italy (e-mail: alberto.bemporad@imtlucca.it).

The activities of L. Cannelli and M. Schranz are partially supported by the European Union, Grant Agreement n. 101093126 (project ACES: Autopoiesis Cognitive Edge-cloud Services).

The activities of L. Cannelli and D. Piga are partially supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 22.00490 and by the Eurostars programme, project "E3093 - SLIMPEC: A Software suite for LearnIng-based embedded Model PrEdictive Control".

The activities of A. Bemporad are partially supported by the European Union (ERC Advanced Research Grant COMPACT, No. 101141351).

autonomy requirements.

Several contributions in the literature address the problem of distributed black-box optimization, by assuming access to quantitative evaluations, often ignoring the possibility of preference-based or qualitative feedback [15], [16], [17]. Other works have also explored distributed learning with preference information, such as federated learning from pairwise comparisons [18], decentralized dueling bandits [19], and collaborative preference aggregation in multi-agent systems [20]. However, these approaches primarily focus on ranking, online decision-making, or reward-based learning. In contrast, to the best of our knowledge, there is no contribution on solving a global black-box optimization problem in a distributed setting, where agents rely exclusively on local, qualitative feedback in the form of pairwise preferences.

In this work, we propose Distributed-GLISp (D-GLISp), a novel algorithm that extends the original (single-agent) GLISp approach [12] to a decentralized, communication-constrained setting, enabling multiple agents to reach consensus on a global solution through local interactions only, without the need of centralized data aggregation.

The remainder of this paper is organized as follows. In Section II, we formalize the problem of distributed preference-based black-box optimization and introduce several notions of global optimality in this setting. Section III presents a step-by-step description of the proposed D-GLISp algorithm, leveraging preferences, surrogate functions, and acquisition functions. Section IV illustrates how distributed optimization is incorporated into the proposed algorithmic framework. Section V presents numerical results on benchmark global optimization problems, and on decentralized preference-based multi-agent calibration of a Model Predictive Controller (MPC) laws for autonomous driving.

## II. Problem Setting

In the next paragraphs, we formalize the setting of distributed multi-agent preference-based optimization addressed in the paper, introducing the individual agents' optimization problems and the preference-based observation model, comparing alternative definitions of global optimality in the context of distributed multi-agent preferences.

### A. Preference-based optimization

Consider a network composed by $N$ computing units (agents), where each unit $i$ (with $i = 1, \dots, N$) aims at solving the following optimization problem:

$$x_i^\star \in \arg\min_{x \in \mathcal{X}} f_i(x), \tag{1}$$

where $x \in \mathbb{R}^n$, $\mathcal{X} \subseteq \mathbb{R}^n$ is the set of feasible solutions, supposed to be known and common to all agents, and $f_i : \mathbb{R}^n \to \mathbb{R}$ is the objective function that the $i$-th agent wish to minimize. Because the agents are connected through a network and cooperate towards a common objective, the aim is to compute a consensus solution $x^\star \in \mathcal{X}$ rather than the individual minimizers $x_i^\star$; the definition of $x^\star$ will be formally discussed in section II-C.

The values of $f_i(x)$ cannot be quantitatively evaluated. Indeed, according to common settings in preference-based optimization [12], we assume that $f_i$ can only be observed through comparisons between two values $f_i(x)$ and $f_i(y)$, for any pair $x, y \in \mathcal{X}$. For this purpose, for each $i = 1, \dots, N$, we introduce the *preference function* $\pi_i : \mathcal{X} \times \mathcal{X} \to \{-1, 0, 1\}$ defined as:

$$\pi_i(x, y) \triangleq \begin{cases} -1 & \text{if, for agent } i,\ x \text{ is "better" than } y, \\ 0 & \text{if, for agent } i,\ x \text{ is "as good as" } y, \\ 1 & \text{if, for agent } i,\ y \text{ is "better" than } x, \end{cases} \tag{2}$$

where for all $x, y, z \in \mathcal{X}$ it holds: i) $\pi_i(x, x) = 0$; ii) $\pi_i(x, y) = -\pi_i(y, x)$; and iii) the transitive property $\pi_i(x, y) = \pi_i(y, z) = -1 \Rightarrow \pi_i(x, z) = -1$. Given that agent $i$ aims at minimizing $f_i$, if the latter were known we could rewrite (2) as:

$$\pi_i(x, y) \triangleq \begin{cases} -1 & \text{if } f_i(x) < f_i(y), \\ 0 & \text{if } f_i(x) = f_i(y), \\ 1 & \text{if } f_i(x) > f_i(y). \end{cases} \tag{3}$$

Furthermore, we take into consideration realistic assumptions for a distributed environment. In particular, we assume that each agent does not share its preferences with the other agents, due to privacy, limited communication bandwidth, or the absence of a central coordinating entity. At the same time, the agents cooperate to meet a consensus on a global objective (defined later).

### B. Communication network

The communication network among the agents is modeled as a fixed, directed weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where $\mathcal{V} = \{1, \dots, N\}$ is the set of vertices/agents, $\mathcal{E} \subseteq \{1, \dots, N\} \times \{1, \dots, N\}$ is the set of edges/communication links, and $\mathcal{A} \in \mathbb{R}_+^{N \times N}$ is the weighted adjacency matrix of the graph. The edge $(i, j) \in \mathcal{E}$ models the fact that agent $i$ can send a message to agent $j$. $\mathcal{A}$ is compliant with the topology described by $\mathcal{E}$, i.e., for each $(i, j)-$entry $\alpha_{ij}$ of $\mathcal{A}$, $\alpha_{ij} > 0$ if $(i, j) \in \mathcal{E}$, and $\alpha_{ij} = 0$ otherwise. In the rest of this work, we assume that: i) $\mathcal{G}$ is strongly-connected and ii) $\mathcal{A}$ is doubly stochastic. These are two common minimal assumptions ensuring that the information of each agent influences the information of any other agent infinitely often in time (see, e.g., [21]).

### C. Optimality for distributed preferences

The main question we now want to address is: *How do we define a global optimum point $x^\star$ in terms of preferences, given that we have different agents expressing their preferences based on their own underlying functions $f_i$?*

A first intuitive definition of optimum preference point in a multi-agent environment could be the following one: $x^\star \in \mathcal{X}$ is a global optimum preference point iff

$$|\{i \in \mathcal{V} | \pi_i(x^\star, \bar{x}) \leq 0\}| \geq |\{i \in \mathcal{V} | \pi_i(\bar{x}, x^\star) \leq 0\}|, \ \forall \bar{x} \in \mathcal{X} \tag{4}$$

with $|\cdot|$ denoting the set cardinality.

In other words, the definition stated above means that $x^\star$ is the optimal point if and only if there is no other point $\bar{x}$

that receives a higher number of preferences when compared to $x^\star$. An issue with this definition of global optimality lies in the fact that such a point $x^\star$ might not exist at all, as the following example shows.

**Example 1**. Let us assume that each agent aims at solving problem (1), and only 3 feasible points $x, y, z \in \mathcal{X}$ exist. $N = 5$ agents, $\mathcal{V} = \{\text{A, B, C, D, E}\}$, are considered. The agents have the following preferences:

$$\text{A prefers } y \text{ to } x, \text{ and } x \text{ to } z.$$

$$\text{B prefers } y \text{ to } x, \text{ and } x \text{ to } z.$$

$$\text{C prefers } z \text{ to } y, \text{ and } x \text{ to } z.$$

$$\text{D prefers } z \text{ to } y, \text{ and } y \text{ to } x.$$

$$\text{E prefers } z \text{ to } y, \text{ and } y \text{ to } x.$$

The above preferences can be visualized in Table I, under assumption that transitive property holds.

TABLE I: Preferences of each agent on the comparisons between points $x, y$, and $z$: ✓ denotes that the statement in the column is satisfied by the agent in the row; x otherwise.

| | $x$ better than $y$ | $x$ better than $z$ | $y$ better than $z$ |
|---|---|---|---|
| A | x | ✓ | ✓ |
| B | x | ✓ | ✓ |
| C | ✓ | ✓ | x |
| D | x | x | x |
| E | x | x | x |

Counting the preferences in Table I, we obtain the summary reported in Table II.

TABLE II: Summary of the preferences of Example 1 expressed by the 5 agents on points $x, y$, and $z$, compared against each other.

| | $x$ | $y$ | $z$ |
|---|---|---|---|
| $\lvert\{i \in \mathcal{V} \lvert \pi_i(\cdot, x) \leq 0\}\rvert$ | 5 | 4 | 2 |
| $\lvert\{i \in \mathcal{V} \lvert \pi_i(\cdot, y) \leq 0\}\rvert$ | 1 | 5 | 3 |
| $\lvert\{i \in \mathcal{V} \lvert \pi_i(\cdot, z) \leq 0\}\rvert$ | 3 | 2 | 5 |

As it is possible to see from the example, none of the tested points satisfies definition (4), thus no global optimum exists. ■

An alternative operative possible definition for an optimum preference point, adopted in this paper, is given by:

$$x^\star \in \arg\max_{x \in \mathcal{X}} \frac{\int_{\mathcal{X}} |\{i \in \mathcal{V} | \pi_i(x, \tilde{x}) \leq 0\}| \mathrm{d}\tilde{x}}{\int_{\mathcal{X}} \mathrm{d}\tilde{x}}, \tag{5}$$

which means searching for the point $x^\star$ that on average is the most preferred (or neutral) by the agents on the feasible domain $\mathcal{X}$. In Example 1 above, $y$ would be the optimum preference point according to definition (5).

We point out that other definitions of global optimum might be also reasonable. For example, one may search for a point $x^\star$ that, differently from (5), does not behave better than other points on average, but it guarantees to always be preferred by a minimum number of agents, i.e.:

$$x^\star \in \arg\max_{x \in \mathcal{X}} \min_{\tilde{x} \in \mathcal{X}} |\{i \in \mathcal{V} | \pi_i(x, \tilde{x}) \leq 0\}|. \tag{6}$$

**Example 2**. As an additional example to show the definitions of global optimizers in (5) and (6), let us consider a network with $N = 11$ agents and evaluate 2 different points $x_1$ and $x_2$ with respect to other 6 points. The situation visualized in Table III occurs, where each cell shows the number of agents that prefer the point on the row with respect to the one in the column.

TABLE III: Preferences expressed by 11 agents on points $x_1$ and $x_2$, compared with 6 other points.

| | vs $x_1$ | vs $x_2$ | vs $x_3$ | vs $x_4$ | vs $x_5$ | vs $x_6$ |
|---|---|---|---|---|---|---|
| $x_1$ | - | 7 | 6 | 7 | 6 | 7 |
| $x_2$ | 4 | - | 10 | 11 | 9 | 9 |

In this case, $x_1$ will be chosen as the optimizer according to criterion (6) since $x_1$ always obtains at least 6 preferences, and thus is better than $x_2$ in the worst case scenario. On the other and, $x_2$ is the global optimum according to criterion (5) as $x_2$ obtains $8.6$ preferences on average with respect to the $6.6$ on average of $x_1$. ■

In the following section, we present the D-GLISp algorithm, which aims at computing the global optimum $x^\star$ according to criterion (5), through a decentralized strategy relying on the communication graph $\mathcal{G}$ and based on active preference learning.

## III. D-GLISp

The D-GLISp algorithm extends the original GLISp method [12] to a distributed, multi-agent setting for preference-based optimization. In this section, we first describe how each agent computes a local surrogate function $\hat{f}_i$ that approximates its unknown objective function $f_i$ (Section III-A). Since no information is shared among agents, the construction of these surrogates follows directly the original GLISp procedure, without any modification.

We then address the cooperative nature of the problem: the agents collectively aim to identify the global optimum $x^\star$ as defined in (5). To this end, we introduce an active learning strategy that iteratively refines the estimate of $x^\star$ using acquisition functions, constructed by combing the estimated surrogate functions for all agents as well as local exploration terms based on inverse distance weighting.

### A. Local surrogate functions

Assume that each agent $i$ has collected a set of local points $\mathcal{D}_i = \{x_j\}_{j=1}^{M_i}$, with $M_i \geq 2$ and $x_j \in \mathcal{X}$ for any $j$, and have evaluated a *preference vector* $b_i \in \{-1, 0, 1\}^{P_i}$, where $P_i$, $1 \leq P_i \leq \binom{M_i}{2}$, is the number of pairwise preferences evaluated by agent $i$ among the points in $\mathcal{D}_i$ according to the preference rule in (2). Each element $b_{i,h}$ (with $h = 1, \ldots, P_i$) of the *preference vector* $b_i$ represents the preference expressed by the agent and is formally defined as

$$b_{i,h} = \pi_i(x_{k(h)}, x_{j(h)}), \tag{7}$$

where $k(h), j(h) \in \{1, \ldots, M_i\}$, $k(h) \neq j(h)$.

Based on the available dataset $\mathcal{D}_i$ and the related preference vector $b_i$, each agent $i$ thus constructs a local surrogate $\hat{f}_i$

of $f_i$ modelling its preferences according to the criterion (3). Following the formulation of the original GLISp algorithm, the surrogate $\hat{f}_i$ is parametrized as a linear combination of *Radial Basis Functions* (RBFs) [22], [23]:

$$\hat{f}_i(x) = \sum_{j=1}^{M_i} \beta_j \phi(\epsilon d(x, x_j)), \tag{8}$$

where $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the squared Euclidean distance

$$d(x_1, x_2) = \|x_1 - x_2\|_2^2, \tag{9}$$

$\epsilon > 0$ is a scalar parameter, $\phi : \mathbb{R} \to \mathbb{R}$ is an RBF, and $\beta = [\beta_1 \ \dots \ \beta_{M_i}]^T$ are the unknown coefficients to be computed based on the preferences expressed in the preference vector $b_i$. The RBF function adopted in this paper is a Gaussian function, defined as $\phi(\epsilon d) = e^{-(\epsilon d)^2}$. Other examples of RBFs can be found in [22], [24].

According to the preference relation (3), the following constraints are imposed on $\hat{f}_i$:

$$\begin{array}{ll} \hat{f}_i(x_{k(h)}) \leq \hat{f}_i(x_{j(h)}) - \sigma + \varepsilon_h & \text{if } \pi_i(x_{k(h)}, x_{j(h)}) = -1 \\ \hat{f}_i(x_{k(h)}) \geq \hat{f}_i(x_{j(h)}) + \sigma - \varepsilon_h & \text{if } \pi_i(x_{k(h)}, x_{j(h)}) = 1 \\ |\hat{f}_i(x_{k(h)}) - \hat{f}_i(x_{j(h)})| \leq \sigma + \varepsilon_h & \text{if } \pi_i(x_{k(h)}, x_{j(h)}) = 0 \end{array} \tag{10}$$

for all $h = 1, \dots, P_i$, where $\sigma > 0$ is a given tolerance and $\varepsilon_h$ are positive slack variables which aim at softening the constraints in (3). Constraint infeasibility might be due to an inappropriate selection of the RBF (namely, poor flexibility in the parametric description of the surrogate $\hat{f}_i$) or because the agent expresses inconsistent preferences.

Based on the parametrization of $\hat{f}_i$ in (8), a coefficient vector $\beta$ satisfying the constraints (10) can be computed by solving the following convex QP problem

$$\begin{aligned} \min_{\beta,\varepsilon} \quad & \sum_{h=1}^{P_i} \varepsilon_h^2 + \frac{\lambda}{2} \sum_{j=1}^{M_i} \beta_j^2 \\ \text{s.t.} \quad & \sum_{i=1}^{M_i} (\phi(\epsilon d(x_{k(h)}, x_i) - \phi(\epsilon d(x_{j(h)}, x_i)) \beta_k \\ & \qquad \leq -\sigma + \varepsilon_h, \qquad \forall h: \ b_h = -1 \\ & \sum_{i=1}^{M_i} (\phi(\epsilon d(x_{k(h)}, x_i) - \phi(\epsilon d(x_{j(h)}, x_i)) \beta_k \\ & \qquad \geq \sigma - \varepsilon_h, \qquad \forall h: \ b_h = 1 \\ & \left| \sum_{i=1}^{M_i} (\phi(\epsilon d(x_{k(h)}, x_i) - \phi(\epsilon d(x_{j(h)}, x_i)) \beta_k \right| \\ & \qquad \leq \sigma + \varepsilon_h, \qquad \forall h: \ b_h = 0 \\ & h = 1, \dots, P_i \end{aligned} \tag{11}$$

where $\lambda > 0$ is a regularization parameter which guarantees that the cost function (11) is strictly convex and admits a unique solution.

In the following sections, we describe how to approximate the global optimum $x^*$ defined in (5) using the local surrogate functions $\hat{f}_i$ obtained by solving (11). Furthermore, in Section IV we will show how to compute the global optimum $x^*$ using distributed optimization strategies, since agents are assumed not to exchange their local surrogates $\hat{f}_i$.

### *B. Computing a global optimizer*

Definition (5) characterizes an optimal point $x^\star$ in terms of preferences, which represents the solution we aim to identify. Since the local functions $f_i$ are not available, we derive an approximation of (5) based on the local surrogates $\hat{f}_i$.

First, using the definitions of the preference function (3), we equivalently rewrite the optimal preference point (5), as follows:

$$x^\star \in \arg\max_{x \in \mathcal{X}} \frac{\int_{\mathcal{X}} \sum_{i=1}^{N} H\left(f_i(\tilde{x}) - f_i(x)\right) \mathrm{d}\tilde{x}}{\int_{\mathcal{X}} \mathrm{d}\tilde{x}}, \tag{12}$$

where $H(\cdot)$ denotes the Heaviside step function, $\forall x \in \mathbb{R}$:

$$H(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Let us approximate $H(\cdot)$ with the continuous and differentiable sigmoid function $s(x) = 1/(1 + e^{-x})$, $x \in \mathbb{R}$. By also using the local surrogate functions $\hat{f}_i$ instead of the unknown true ones $f_i$, the objective function in problem (12) can be approximated as:

$$\begin{aligned} & \frac{\int_{\mathcal{X}} \sum_{i=1}^{N} H\left(f_i(\tilde{x}) - f_i(x)\right) \mathrm{d}\tilde{x}}{\int_{\mathcal{X}} \mathrm{d}\tilde{x}} \simeq \\ \simeq & \frac{\int_{\mathcal{X}} \sum_{i=1}^{N} H\left(\hat{f}_i(\tilde{x}) - \hat{f}_i(x)\right) \mathrm{d}\tilde{x}}{\int_{\mathcal{X}} \mathrm{d}\tilde{x}} \simeq \\ \simeq & \frac{\int_{\mathcal{X}} \sum_{i=1}^{N} s\left(\hat{f}_i(\tilde{x}) - \hat{f}_i(x)\right) \mathrm{d}\tilde{x}}{\int_{\mathcal{X}} \mathrm{d}\tilde{x}} = \\ = & \sum_{i=1}^{N} \int_{\mathcal{X}} s\left(\hat{f}_i(\tilde{x}) - \hat{f}_i(x)\right) \mathrm{d}\tilde{x} = \\ = & \sum_{i=1}^{N} \hat{r}_i(x), \end{aligned} \tag{13}$$

where:

$$\hat{r}_i(x) \triangleq \int_{\mathcal{X}} s\left(\hat{f}_i(\tilde{x}) - \hat{f}_i(x)\right) \mathrm{d}\tilde{x}, \tag{14}$$

is a *preference surrogate function* that estimates how many preferences a point $x$ receives with respect to all points in the domain $\mathcal{X}$.

Thanks to the steps carried out in (13), the approximation $\hat{x}^\star$ of the optimal preference point $x^\star$ can be computed as:

$$\hat{x}^\star \in \arg\max_{x \in \mathcal{X}} \sum_{i=1}^{N} \hat{r}_i(x), \tag{15}$$

that is, by solving an optimization problem whose cost function has a sum-utility structure. This is the classical formulation used in distributed optimization algorithms over multi-agent networks [21], as we will present in Section IV.

However, relying solely on surrogate functions may lead to suboptimal solutions. Indeed, surrogate models built from

previously collected data are inherently uncertain in unexplored regions of the search domain. To balance the trade-off between exploitation and exploration, the original GLISp algorithm combines a preference-based surrogate model with an exploration term based on the distance from previously evaluated points. In the following, we describe how this principle is extended to the distributed setting.

### C. Local acquisition functions

As in [17], the IDW (Inverse Distance Weighting) function is employed to encourage exploration. For each agent $i$, this function is defined as:

$$z_i(x) = \begin{cases} 0 & x \in x_1, \dots, x_{M_i}, \\ \tan^{-1}\left(\frac{1}{\sum_{j=1}^{M_i} w_j(x)}\right) & \text{otherwise}, \end{cases}$$

where $w_j(x) = \frac{1}{\|x - x_j\|^2}$, and $x_1, \dots, x_{M_i}$ represent the samples stored in the local dataset $\mathcal{D}_i$. By construction, $z_i(x) = 0$ for any input previously evaluated by agent $i$, while $z_i(x) > 0$ elsewhere in $\mathbb{R}^n$. Furthermore, $z_i(x)$ increases with the distance between $x$ and the previously explored samples, thereby promoting sampling in less-visited regions of the input space.

To balance exploration and exploitation, each agent $i$ defines a local acquisition function $a_i : \mathcal{X} \to \mathbb{R}$, parameterized by an exploration weight $\delta \geq 0$. Although agent $i$ does not have access to the global preference surrogate $\hat{r}(x) \triangleq \sum_{i=1}^{N} \hat{r}_i(x)$, the acquisition function is conceptually defined as:

$$a_i(x) \triangleq \frac{\hat{r}(x)}{\Delta \hat{r}} + \delta z_i(x), \tag{16}$$

where $\Delta \hat{r} \triangleq \max_{x \in \mathcal{X}} \hat{r}(x) - \min_{x \in \mathcal{X}} \hat{r}(x)$ denotes the range of the surrogate and serves as a normalization factor to facilitate the tuning of the exploration parameter $\delta \in (0, 1]$.

During each D-GLISp iteration, a single agent—say agent $i$ with $i = 1, \dots, N$—is selected following a round-robin schedule. This agent then determines its next evaluation point $x^{i,\star}$ by maximizing the acquisition function $a_i$, i.e.,

$$x^{i,\star} \in \arg\max_{x \in \mathcal{X}} a_i(x), \tag{17}$$

where the superscript emphasizes that the selected input corresponds to agent $i$'s query. Note that in the definition of $a_i$, the global preference surrogate $\hat{r}$ guides the *exploitation* term, ensuring cooperation among agents toward searching the optimum preference point defined in (5) and approximated by (15). Meanwhile, the *exploration* term is driven by the local IDW function $z_i$, allowing each agent to independently probe unexplored areas of the domain $\mathcal{X}$ relevant to its local preference surrogate objective $f_i$.

*Remark 1:* Although the acquisition function $a_i$ is specific to the agent $i$, it inherently depends on the preference surrogate functions $\hat{r}_j$ of the other agents $j \neq i$. However, these surrogate functions are not shared among agents, as each agent has access only to its own. This necessitates the use of a distributed optimization strategy to solve problem (17), as discussed in Section IV. ■

### D. Next point selection

After selecting the new input $x^{i,\star}$, the following steps are performed: (i) agent $i$ evaluates at least one new preference between the new point $x^{i,*}$ and previous observations in $\mathcal{D}_i$, updating accordingly the preference vector $b_i$ and the dataset $\mathcal{D}_i$; (ii) both the local surrogate model $\hat{f}_i$ and the IDW-based function $z_i$ are updated; (iii) a new agent $j$ is chosen. This cycle is repeated until the iteration count reaches the predefined maximum number of iterations $T_{\max}$. At the end of the process, the final global objective $\hat{r}$ is minimized (disregarding the exploration component) to obtain a candidate optimal solution $\hat{x}^\star$. A structured overview of the D-GLISp algorithm is provided in Algorithm 1.

**Algorithm 1** D-GLISp algorithm

**Inputs**: maximum number of iterations $T_{\max}$; exploration parameter $\delta$; constraint set $\mathcal{X}$; initial datasets $\mathcal{D}_i = \{x_j\}_{j=1}^{M_i}$; initial preference vector $b_i$; and initial surrogate functions $\hat{f}_i$ and $\hat{r}_i$ for all agents $i = 1, \dots, N$.

1: **repeat**
2: select agent $i$ according to a cyclic rule;
3: agent $i$ builds the IDW function $z_i$;
4: compute $x^{i,*}$ in (17) via distributed optimization (Section IV);
5: agent $i$ updates $b_i$ with new preferences evaluated between $x^{i,*}$ and points in $\mathcal{D}_i$
6: agent $i$ updates the local dataset: $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{x^{i,*}\}$;
7: agent $i$ updates the local surrogate $\hat{f}_i(x)$ and $\hat{r}_i$ based on $\mathcal{D}_i$ and $b_i$.
8: **until** maximum numbers of iterations $T_{\max}$ is reached.
9: compute consensus $\hat{x}^\star$ in (15) via distributed optimization (Section IV).

**Output**: consensus $\hat{x}^\star$

*Remark 2:* It is possible to consider a *parallel* and fully decentralized version of Algorithm 1. In this case, all agents compute at each iteration their acquisition functions, solve in a distributed way at the same time $N$ optimization problems as (17), and finally update their datasets $\mathcal{D}_i$ and their preference vectors $b_i$. In this case, there is no need to have a central controller enforcing the cyclic selection rule. ■

*Remark 3:* Steps 4 and 9 of Algorithm 1 require each agent $i$ to construct the function $\hat{r}_i$, which is based on the local surrogate $\hat{f}_i$, as defined in (14). As can be seen by its expression, $\hat{r}_i$ has a complex structure due to the presence of the integral function. Furthermore, since Steps 4 and 9 involve distributed optimization, the gradient of $\hat{r}_i$ is explicitly required too, further motivating the need for a tractable approximation of the integral. In our implementation of D-GLISp, this integral is approximated using Monte Carlo integration. Alternative approximations of this integral based on heuristic approaches may also be valid, as discussed in the Remark 4 (below). However, in our implementations and case studies, the Monte Carlo approach proved to be both simple and accurate. ■

*Remark 4:* Following the discussion in Remark 3, we present here an alternative approach to approximate $\hat{r}_i$, which does not rely on Monte Carlo sampling.

By defining $g_i(\tilde{x}, x) \triangleq \hat{f}_i(\tilde{x}) - \hat{f}_i(x)$, we consider the first-order Taylor approximation of the integrand function $s\left(g_i(\tilde{x}, x)\right)$ in (14) with respect to the variable $\tilde{x}$ centered around $\tilde{x}_0 \in \mathcal{X}$.

$$\begin{aligned}
&\hat{r}_i(x) \approx \\
&\int_{\mathcal{X}} s(g_i(\tilde{x}_0, x)) + s'(g_i(\tilde{x}_0, x)) \left(\nabla_{\tilde{x}} g_i(\tilde{x}_0, x)\right)^{\mathrm{T}} (\tilde{x} - \tilde{x}_0) d\tilde{x} = \\
&= s(g_i(\tilde{x}_0, x)) \int_{\mathcal{X}} d\tilde{x} + \\
&\quad + s'(g_i(\tilde{x}_0, x)) \left(\nabla_{\tilde{x}} g_i(\tilde{x}_0, x)\right)^{T} \int_{\mathcal{X}} (\tilde{x} - \tilde{x}_0) d\tilde{x}, \qquad (18)
\end{aligned}$$

where $\nabla_{\tilde{x}} g_i(\tilde{x}_0, x)$ denotes the gradient of $g_i(\tilde{x}, x)$ computed with respect to $\tilde{x}$ and evaluated at $\tilde{x} = \tilde{x}_0$.

Considering that $\hat{f}_i$ is the sum of $M_i$ RBFs, as shown in (8), it follows that all the terms in (18) (namely, $g_i$ and its gradient) can be computed in closed form. The only challenge arises from the presence of the constraint set $\mathcal{X}$ as the integration domain. If $\mathcal{X}$ is too arbitrary or complex, numerical approximation techniques are required. However, if $\mathcal{X}$ exhibits a simple structure, closed-form solutions may be available. For instance, for the case where $\mathcal{X}$ is a box, i.e., $\mathcal{X} \triangleq \prod_{k=1}^{n} [l_k, u_k]$, with $l_k, u_k \in \mathbb{R}$ and $l_k \leq u_k$ for all $k$, we have:

$$\int_{\mathcal{X}} d\tilde{x} = \prod_{k=1}^{n} (u_k - l_k). \qquad (19)$$

Then, by defining the centroid of $\mathcal{X}$ as $\bar{x} \triangleq [\frac{l_1+u_1}{2}, \ldots, \frac{l_n+u_n}{2}]^{\mathrm{T}}$, we also have:

$$\int_{\mathcal{X}} (\tilde{x} - \tilde{x}_0) d\tilde{x} = (\bar{x} - \tilde{x}_0) \prod_{k=1}^{n} (u_k - l_k). \qquad (20)$$

Substituting (19) and (20) into (18) we obtain a closed-form expression for $\hat{r}_i(x)$ and its gradient that can be used in Steps 4 and 9 of Algorithm 1, instead of the Monte Carlo approximation discussed in Remark 3. Moreover, by selecting the midpoint approximation, i.e., $\tilde{x}_0 = \bar{x}$, the expression (18) simplifies further and becomes:

$$\hat{r}_i(x) = s(g_i(\bar{x}, x)) \prod_{k=1}^{n} (u_k - l_k).$$

■

## IV. Distributed Optimization

As discussed in the previous sections, in the D-GLISp algorithm, each generic agent $i$ optimizes its local acquisition function $a_i$, which inherently depends also on the preference surrogate functions $\hat{r}_j$ of the other agents $j \neq i$. However, these surrogate functions are not shared among agents, namely, each agent has access only to its own. To overcome this limitation, the maximization of the local acquisition function $a_i$ is formulated in a distributed optimization framework, where agents coordinate through communication with their neighbors to collaboratively optimize the local acquisition function $a_i$ of the $i$-th agent.

To address the optimization problem defined in (17) during Step 4 of Algorithm 1 (or, similarly, problem (15) in Step 9), the GTAdam algorithm [25], a decentralized extension of the widely used Adam optimizer [26], is adopted. While Adam is designed for centralized settings, where updates to the solution estimate $x^t$ are computed using a globally available gradient history, GTAdam extends this principle to a networked environment. It achieves this by embedding the gradient tracking mechanism introduced in [27] into the Adam framework, enabling fully decentralized execution without a central coordinator.

A key advantage of GTAdam lies in its communication efficiency, as the agents are not required to share their local models $\hat{r}_i$ or $\hat{f}_i$. Instead, at each iteration $t$, only the gradient of the local preference surrogate $\hat{r}_i$, evaluated at the current iterate $x^t$, is exchanged between directly connected agents. The strong connectivity of the communication graph $\mathcal{G}$ ensures that information from any agent can propagate through the network, while the doubly stochastic nature of the mixing matrix $\mathcal{A}$ guarantees consensus among the agents on the stationary solution. However, because of the non-convex nature of problem (17), no guarantees of convergence can be stated.

## V. Examples

In this section we show the performances of the D-GLISp algorithm on benchmark black-box optimization problems and on distributed design of an MPC controller for autonomous driving. All the codes are implemented in Python and run on the CPUs of an AMD EPYC 7742 server, with a base processor speed of 2.25 GHz, 256 MB L3 cache. *Open MPI* [28] and the *disropt library* [29] are used for the distributed operations and for distributed optimization, respectively. Python codes can be downloaded at `https://leon.idsia.ch/lib_download`.

### A. Benchmark optimization problems

We test D-GLISp on four optimization problems, denoted as `camelsixhumps`, `beale`, `brent`, and `ls`. Problems `camelsixumps`, `beale`, and `brent` are inspired from original optimization benchmarks (commonly used in non-convex optimization), and described in [30]. The global objective of these original benchmarks is a sum of functions. In our problem, we consider each summand as a local function for each agent. As an example, we show here how to reformulate the `camelsixhumps` problem to fit our distributed framework: number of agents $N = 3$; $x \in \mathbb{R}^2$; cost function of the original benchmark

$$f(x) = \underbrace{\left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2}_{f_1(x)} + \underbrace{x_1 x_2}_{f_2(x)} + \underbrace{\left(4x_2^2 - 4\right) x_2^2}_{f_3(x)}$$

under the constraints $x_1, x_2 \in [-5, 5]$.

Problem `ls` is a distributed least square problem, generated according to the procedure described below. A local function $f_{i,\texttt{ls}}(x) \triangleq \|A_i x - b_i\|_2^2$ is associated with each agent $i$, with $A_i \in \mathbb{R}^{100\times N}$, and $b_i \in \mathbb{R}^{100}$. The underlying linear model is generated as: $b_i = A_i x_i^\star$, where the elements of $x_{i,\texttt{ls}}^\star \in \mathbb{R}^N$ are i.i.d. and drawn from the uniform distribution $\mathcal{U}[-1,1]$. The entries of the matrices $A_i$ are also i.i.d. and drawn from the normal distribution. The values of $A_i$ and $b_i$ are normalized with a scaling factor equal to 100. The feasible set is $\mathcal{X} = [-1,1]^N$.

We run `camelsixhumps`, `beale`, and `brent` over a network composed of $N = 3$ agents, while the `ls` problem is solved using $N = 4$ agents. In all the experiments the agents communicate over a fixed undirected graph $\mathcal{G}$, generated using an Erdős–Rényi random model. The adjacency matrix $\mathcal{A}$ of the graph is obtained through a Metropolis-Hastings weight model [31]. Algorithm 1 has been applied on the four aforementioned benchmark problems, running GTAdam as an inner distributed solver in Steps 4 and 9.

In all the experiments, the agents start the optimization procedure with an initial local dataset $\mathcal{D}_i$ composed of $M_i = 2n$ feasible points, generated uniformly at random. Each agent constructs the vector of initial local preferences $b_i$ by comparing the first point in the local dataset $\mathcal{D}_i$ with the second, evaluating the winner of the preference and then comparing it with the third point. The winner is compared with the next point, and so on, until all points in $\mathcal{D}_i$ have been considered. In this way, $P_i = M_i - 1$ initial preferences are obtained. We save the winning point from these initial preferences to compare it with the point generated by the optimization process in Step 5, thereby obtaining a new preference. The winner of this new comparison is stored, and the procedure is repeated each time Step 5 is reached again, thus generating a new preference at each iteration and updating the local surrogates accordingly in Step 7.

For each outer iteration of D-GLISp, the inner solver GTAdam runs for 100 iterations with an initial stepsize equal to 0.01. The value of the exploration hyper-parameter $\delta$ (see (16)) is updated by the agents while D-GLISp is running, according to the following heuristic: $\delta = N(|\mathcal{D}_i| - 1)$. The number of outer iterations is set to $10N$, namely, 30 for `camelsixhumps`, `beale` and `brent`, and 40 for the `ls` benchmark.

Each problem is run for 10 Monte Carlo independent realizations and the performance of D-GLISp is shown in Figures 1 and 2. Each panel in Figure 1 illustrates the results for a single Monte Carlo realization on the considered benchmark problems. Specifically, each plot shows two sequences of points:

i) The red points represent the $\hat{x}^\star$ solutions generated by D-GLISp after each outer iteration, computed according to (15). These points have been computed only for the purpose of monitoring the performances of D-GLISp, but, in practice $\hat{x}^\star$ is computed only once, in Step 9, when the desired maximum number of iterations $T_{\max}$ is reached.
ii) The blue points correspond to a batch of $K = 500$ points uniformly sampled from the feasible domain.

Each point from sets (i) and (ii) is compared against all others in terms of preferences, with each agent responding to the comparisons according to its own $f_i(x)$. The plots in Figure 1 therefore show the total number of preferences obtained by each point when compared to the rest. This procedure serves as a measure to estimate the optimality condition defined in (5). The green line in the figures indicates the value corresponding to the maximum possible number of preferences a point can achieve relative to all other points involved in the procedure, that is: $N(K + T_{\max} - 1)$.

Figure 1 clearly shows that the points computed by the D-GLISp algorithm improve in terms of preferences after each outer iteration, eventually reaching values close to the maximum achievable. Figure 2 presents the same performance results from a different perspective. In each panel, for every single Monte Carlo run, both (i) the final point from the D-GLISp-generated sequence and (ii) the best point among 200 randomly generated samples are shown. It can be seen that the points generated by the D-GLISp algorithm consistently outperform the best of the random samples in every Monte Carlo run, despite D-GLISp requiring only $10N$ iterations.

### *B. Case study: MPC for autonomous driving*

We demonstrate the application of D-GLISp to distributed calibration of an MPC law for autonomous vehicles, focusing on lane-keeping and obstacle-avoidance tasks. A similar scenario was previously explored in [17], [8], [32], but this is the first time where the problem is solved by a decentralized preference-based algorithm.

The control scenario is sketched in Figure 3. A Subject Vehicle (SV) operates on a one-way, two-lane horizontal road. In contrast to the earlier studies [8], [32], our scenario includes two Obstacle Vehicles (OVs), each occupying the center of a lane and moving forward at a constant velocity. The OVs' initial speeds and positions vary across different test conditions. The SV is governed by an MPC controller, which ensures lane-following and dynamically avoids collisions by performing lane changes, acceleration, or deceleration when necessary. The goal is to collaboratively identify suitable MPC parameters while preserving the privacy of individual experimental settings. Each calibration agent may adopt distinct preferences over the optimization objectives and conduct independent experiments accordingly.

In the following sections, we describe the SV dynamics, the structure of the MPC controller, the agents' control objectives, and the obtained results.

*1) System description:* To model the vehicle kinematics and simulate the experimental scenario, we adopt a simplified bicycle model with two degrees of freedom. The model uses the front wheel as the reference point, and its dynamics is described by the following nonlinear equations:

$$\begin{aligned} \dot{x}_f &= v\cos(\theta+\psi), \\ \dot{w}_f &= v\sin(\theta+\psi), \\ \dot{\theta} &= \frac{v\sin(\psi)}{L}, \end{aligned} \tag{21}$$

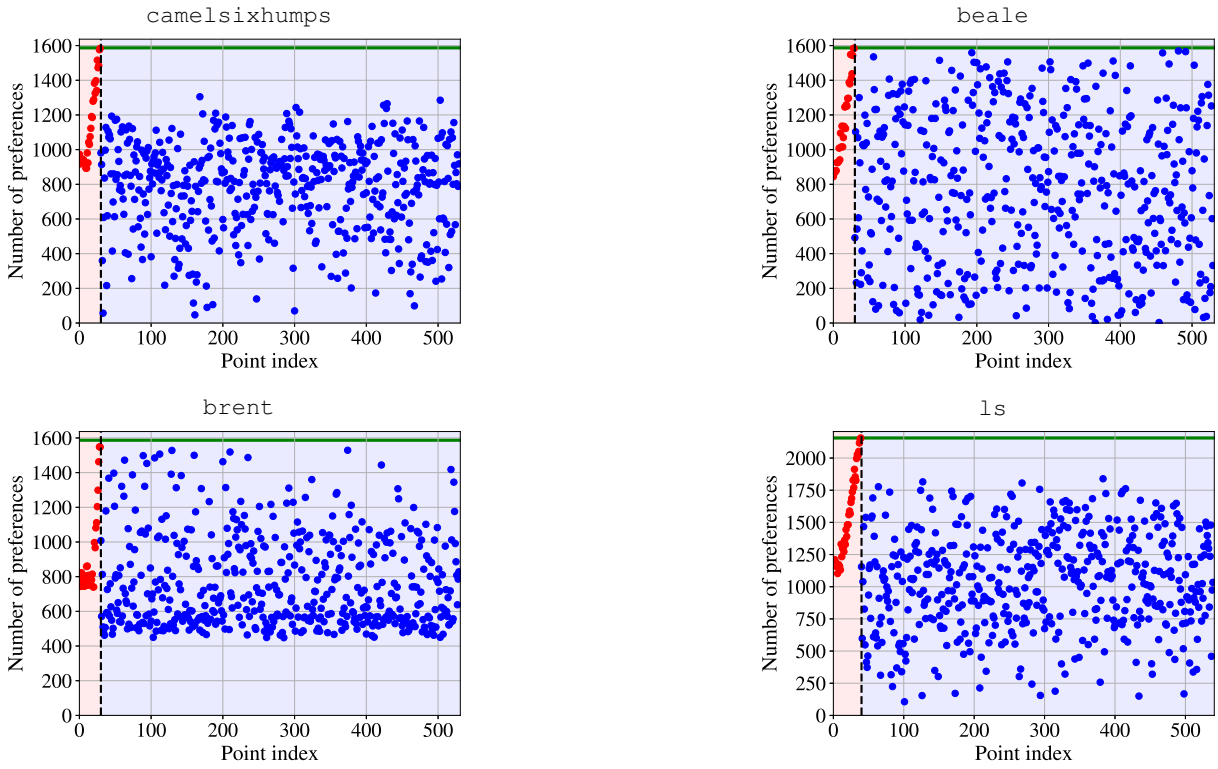


Fig. 1: Performance of D-GLISp on a single Monte Carlo realization across the 4 benchmarks. Maximum number of possible preferences (green line); number of preferences obtained from points generated by D-GLISp iteration by iteration (red dots in the shaded red area); number of preferences obtained from uniformly random points drawn from the feasible set (blue dots).

where $x_f$ and $w_f$ (m) denote the longitudinal and lateral positions of the SV's front wheel, respectively, and $\theta$ (rad) is the yaw angle. These variables define the state vector $s \triangleq [x_f \;\; w_f \;\; \theta]^{\mathrm{T}}$. The control inputs are grouped in the input vector $u \triangleq [v, \;\; \psi]^{\mathrm{T}}$, where $v$ (m/s) represents the longitudinal vehicle's speed and $\psi$ (rad) is the steering angle. The parameter $L$ (m) corresponds to the vehicle's wheelbase.

*2) MPC formulation:* We assume full state availability, and the control output is defined to coincide with the system state, i.e., $y = s$. The nonlinear dynamics in (21) are discretized and linearized around a nominal trajectory to obtain a time-varying linear approximation of the system, given by:

$$\begin{aligned}\tilde{s}_{k+1} = &\begin{bmatrix} 1 & 0 & -\bar{v}_k \sin(\bar{\theta}_k + \bar{\psi}_k) T_s \\ 0 & 1 & \bar{v}_k \cos(\bar{\theta}_k + \bar{\psi}_k) T_s \\ 0 & 0 & 1 \end{bmatrix} \tilde{s}_k \\ &+ \begin{bmatrix} \cos(\bar{\theta}_k + \bar{\psi}_k) T_s & -\bar{v}_k \sin(\bar{\theta}_k + \bar{\psi}_k) T_s \\ \sin(\bar{\theta}_k + \bar{\psi}_k) T_s & \bar{v}_k \cos(\bar{\theta}_k + \bar{\psi}_k) T_s \\ \frac{\sin(\bar{\psi}_k)}{L} T_s & \frac{\bar{v}_k \cos(\bar{\psi}_k)}{L} T_s \end{bmatrix} \tilde{u}_k, \\ \tilde{y}_k = &\; \tilde{s}_k, \end{aligned} \tag{22}$$

where $T_s$ is the sampling time, and subscript $k$ denotes the discrete-time index. The variables with tilde $(\tilde{\cdot})$ represent deviations from nominal trajectories, defined as $\tilde{x} \triangleq x - \bar{x}$, with $x \in \{s_{k+1}, s_k, v_k, \theta_k, \psi_k, u_k, y_k\}$. Variables with overbars $(\bar{\cdot})$ denote nominal reference values.

Based on the linearized model (22), a linear time-varying MPC strategy is formulated and solved using a real-time iteration scheme [33], [34]. At each time step $t$, the following quadratic programming problem is solved to compute the optimal control inputs:

$$\begin{aligned}\min_{\{u_{t+k|t}\}_{k=0}^{N_u-1}} &\sum_{k=0}^{N_p-1} \left\| y_{t+k|t} - y^{\mathrm{ref}}_{t+k} \right\|^2_{Q_y} \\ &+ \sum_{k=0}^{N_p-1} \left\| u_{t+k|t} - u^{\mathrm{ref}}_{t+k} \right\|^2_{Q_u} \\ &+ \sum_{k=0}^{N_p-1} \left\| \Delta u_{t+k|t} \right\|^2_{Q_{\Delta u}}, \end{aligned} \tag{23}$$

subject to the following constraints:

$$\begin{aligned} &y_{\min} \le y_{t+k|t} \le y_{\max}, \quad k = 1, \dots, N_p, \\ &u_{\min} \le u_{t+k|t} \le u_{\max}, \quad k = 1, \dots, N_p, \\ &\Delta u_{\min} \le \Delta u_{t+k|t} \le \Delta u_{\max}, \quad k = 1, \dots, N_p, \\ &u_{t+N_u+j|t} = u_{t+N_u|t}, \quad j = 1, \dots, N_p - N_u, \end{aligned} \tag{24}$$

where $\Delta u_{t+k|t} \triangleq u_{t+k|t} - u_{t+k-1|t}$ denotes the input increment, and $y^{\mathrm{ref}}$, $u^{\mathrm{ref}}$ are the reference trajectories for output and input, respectively. $Q_y$, $Q_u$, and $Q_{\Delta u}$ are positive definite weighting matrices, and $N_u$ and $N_p$ define the control

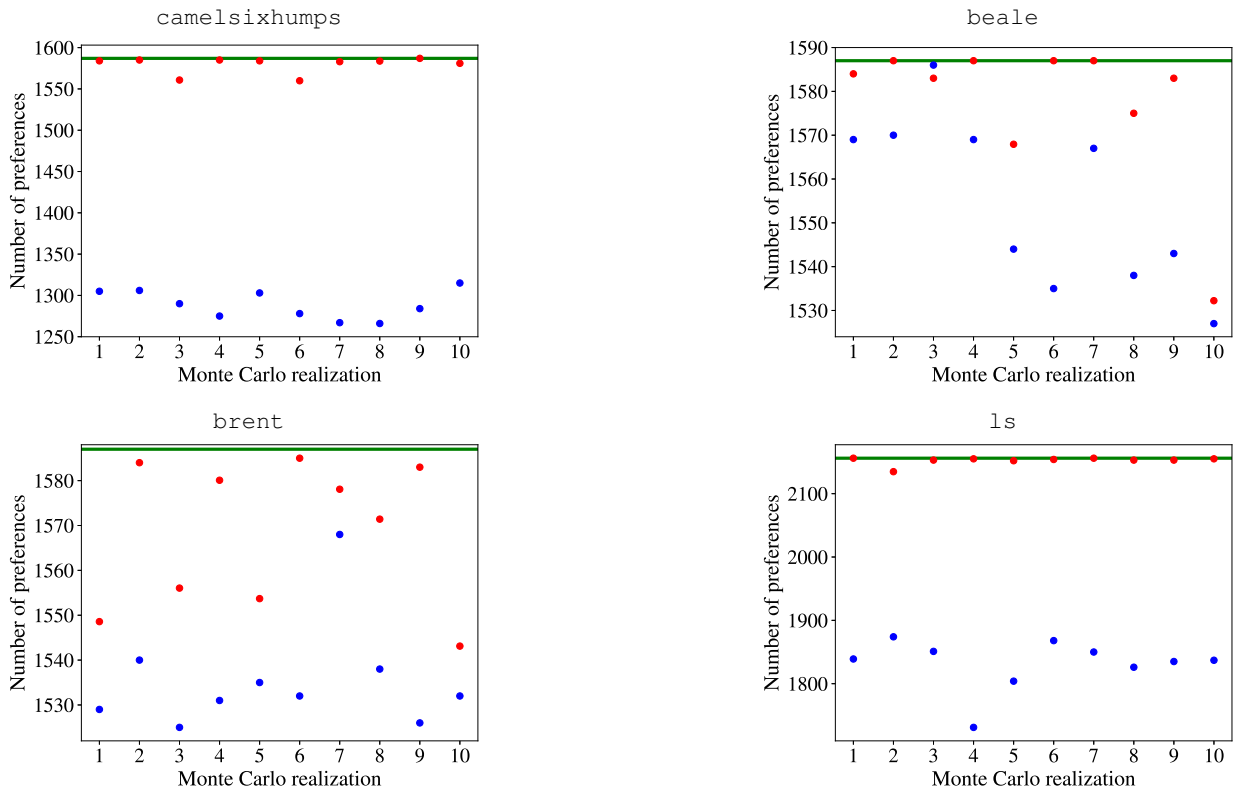


Fig. 2: Performance of D-GLISp across the 4 benchmark problems for 10 Monte Carlo runs. Maximum number of possible preferences (green line); number of preferences obtained by optimal point computed with D-GLISp (red dots); number of preferences obtained by the best out of 200 randomly generated point (blue dots).

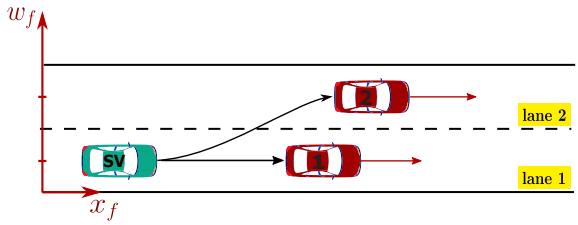


Fig. 3: Test scenario for MPC calibration. Subject (green car) and obstacle vehicles (red cars).

and prediction horizons, respectively.

*3) Test scenarios and control objectives:* The parameters selected for calibration include the control horizon $N_u$, the prediction horizon $N_p$, and the diagonal elements of the input increment weight matrix $Q_{\Delta u} \triangleq \begin{bmatrix} q_{u11} & 0 \\ 0 & q_{u22} \end{bmatrix}$. These parameters are tuned over the following ranges: $N_p \in [10,\ 30]$; $N_u = \epsilon_c N_p$, with $\epsilon_c \in [0.1,\ 1]$ and $N_u$ rounded to its closest integer; and $\log(q_{u11}),\ \log(q_{u22}) \in [-5,\ 3]$. All remaining MPC parameters are fixed and not subject to calibration. Specifically, the sampling time is set to $T_s = 0.085$ s, and the weighting matrices are

$$Q_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad Q_u = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The reference speed $v^{\text{ref}}$ is fixed at 50 km/h. During operations, the longitudinal velocity $v$ of SV is constrained to lie within the interval $[1,\ 90]$ km/h, with a maximum acceleration/deceleration rate $\dot{v} \in [-4,\ 4]$ m/s$^2$. The steering angle $\psi$ varies between $\pm 45°$, with a maximum rate $\dot{\psi} \in [-60,\ 60]°/\text{s}$, and its reference value is set to $\psi^{\text{ref}} = 0°$.

As for the control outputs, the lateral position is constrained to $w_f \in [-0.6,\ 3.6]$ m, ensuring the SV remains within the roadway. The lateral reference $w_f^{\text{ref}}$ is set to either 0 m or 3 m, corresponding to the center of lane 1 or lane 2, respectively, depending on the active lane. The yaw angle $\theta$ is constrained to $[-90°,\ 90°]$, with $\theta^{\text{ref}} = 0°$.

The MPC controller is designed to achieve three high-level objectives: (i) maintain a consistent vehicle velocity; (ii) minimize abrupt changes in steering; (iii) avoid collisions with OVs. In our case study, and only for simulation purposes, these objectives are emulated via the following mathematical expressions, respectively:

$$\begin{aligned} m_1 &= \frac{1}{N_{\text{total}}} \sum_{k=1}^{N_{\text{total}}} \left| \frac{v_k - v_k^{\text{ref}}}{v_k^{\text{ref}}} \right|, \\ m_2 &= \frac{1}{N_{\text{total}}} \sum_{k=1}^{N_{\text{total}}} \left| \frac{\psi_k - \psi_k^{\text{ref}}}{\psi_k^{\text{ref}} + 0.1} \right|, \\ m_3 &= 1000 \cdot \mathcal{I}_{\text{collision}}, \end{aligned} \tag{25}$$

where $N_{\text{total}} = 2 \left\lfloor \frac{t_{\text{exp}}}{2T_s} \right\rfloor$ is the number of discrete time steps,

$t_{\text{exp}}$ is the experiment duration, and $\mathcal{I}_{\text{collision}} \in \{0, 1\}$ is an indicator function that equals 1 if a collision between the SV and any OV occurs, and 0 otherwise.

The case study involves $N = 4$ independent calibrators (agents), each prioritizing the first two objectives differently. Specifically, for agent $i \in \{1, 2, 3, 4\}$, the corresponding objective functions $f_i$ are expressed as:

$$\begin{aligned} f_1 &= 0.5m_1 + 0.5m_2 + m_3, \\ f_2 &= 0.8m_1 + 0.2m_2 + m_3, \\ f_3 &= 0.3m_1 + 0.7m_2 + m_3, \\ f_4 &= 0.6m_1 + 0.4m_2 + m_3. \end{aligned} \tag{26}$$

Each agent runs simulations under different OV initial conditions, as reported in Table IV, where $x^0_{f,\text{OV}}$ denotes the initial longitudinal position and $v^0_{\text{OV}}$ the initial speed of each OV. The ultimate goal of calibration is to design an MPC controller that performs robustly across all test conditions and agents. Each simulation runs for $t_{\text{exp}} = 30$ s.

TABLE IV: Initial conditions of OVs for each agent: $x^0_{f,\text{OV}}$ is the initial longitudinal position; $v^0_{\text{OV}}$ is the initial speed.

| agent | $x^0_{f,\text{OV}}$ [m] | | $v^0_{\text{OV}}$ [km/h] | |
|---|---|---|---|---|
| | $\text{OV}_1$ | $\text{OV}_2$ | $\text{OV}_1$ | $\text{OV}_2$ |
| 1 | 10 | 33 | 38 | 40 |
| 2 | 15 | 17 | 30 | 48 |
| 3 | 20 | 60 | 40 | 42 |
| 4 | 9 | 20 | 60 | 45 |

*4) Calibration process:* The calibration of the MPC parameters $N_p$, $\epsilon_c$, $\log(q_{u11})$, and $\log(q_{u22})$ have been performed by running D-GLISp with the following configuration:

- the $i$-th agent (with $i = 1, 2, 3, 4$) expresses its preference based on its own local function $f_i$ defined in (26), without knowing the preferences of the other agents;
- the agents communicate over a fixed undirected graph $\mathcal{G}$, generated using an Erdős-Rényi random model $(4, 0.3)$. The adjacency matrix $\mathcal{A}$ of the graph is obtained through the Metropolis-Hastings weight model;
- GTAdam is used as an inner distributed solver in Steps 4 and 9. For each iteration of D-GLISp, the inner solver GTAdam runs for 100 iterations with an initial stepsize equal to 0.001;
- the $i$-th agent (with $i = 1, 2, 3, 4$) has an initial local dataset $\mathcal{D}_i$ composed of $M_i = 4$ feasible points generated uniformly at random;
- the value of the hyper-parameter $\delta$ is updated while D-GLISp is running, according to the same heuristic described in Section V-A;
- D-GLISp terminates after $T_{\max} = 100$ outer iterations.

*5) Results:* The optimization process involves four initial random experiments and 25 iterations of the D-GLISp algorithm per agent, resulting in a total of $T_{\max} = 100$ experiments across four agents. In Figure 4 a comparison among the $\hat{x}^\star$ points computed by D-GLISp at iterations 1, 31, 51, 81, and 100 is illustrated in terms of the trajectories of the manipulated variables: speed $v$ and steering angle $\psi$. As can be seen from the plots, iteration after iteration the algorithm is able to find increasingly better solutions, converging toward a behavior which achieves smooth and stable responses, indicating efficient and reliable closed-loop behavior.

## VI. Conclusions

We addressed the problem of distributed multi-agent black-box optimization based on preference-based feedback. To this end, we introduced the D-GLISp algorithm, which enables the agents to reach a consensus on a global optimum, where each agent seeks to optimize an unknown local objective using only pairwise preference information, without sharing its own preferences with the others. The method relies on a combination of local surrogate models and exploration mechanisms based on inverse distance weighting, while taking into account communication constraints and privacy preservation.

Although Monte Carlo methods have proven effective in the case studies considered for approximating the integrals involved in the computation of the global optimum, ongoing research is focused on exploring alternative approximation techniques. These methods will be evaluated in terms of both approximation accuracy and computational complexity when maximizing the acquisition function through distributed optimization. Future research activities also include a theoretical and practical analysis of alternative definitions of global optimality in the context of distributed preference-based learning.

## References


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

[2] C. Audet and W. Hare, *Derivative-free and blackbox optimization*. Springer, 2017.

[3] J. Gonzalez, Z. Dai, P. Hennig, and N. D. Lawrence, "Preferential Bayesian optimization," in *International Conference on Machine Learning*, 2017, pp. 1272–1281.

[4] Y. Koyama, I. Sato, and M. Goto, "Sequential gallery for interactive visual design optimization," *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 88:1–88:12, Jul. 2020.

[5] T. Okada, K. Tsuda, K. Terayama, and R. Tamura, "Black-box optimization for automated discovery, design, and optimization based on experiments and computations," *Accounts of Chemical Research*, vol. 54, no. 2, pp. 391–400, 2021.

[6] W. Xue, Q. Cai, Z. Xue, S. Sun, S. Liu, D. Zheng, P. Jiang, and B. An, "Prefrec: Preference-based recommender systems for reinforcing long-term user engagement," *arXiv preprint arXiv:2212.02779*, 2022.

[7] L. Roveda, B. Maggioni, E. Marescotti, A. A. Shahid, A. M. Zanchettin, A. Bemporad, and D. Piga, "Pairwise preferences-based optimization of a path-based velocity planner in robotic sealing tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, 2021.

[8] M. Zhu, A. Bemporad, and D. Piga, "Preference-based MPC calibration," in *European Control Conference*, 2021, also available on https://arxiv.org/pdf/2003.11294.pdf.

[9] A. Benavoli, D. Azzimonti, and D. Piga, "A unified framework for closed-form nonparametric regression, classification, preference and mixed problems with Skew Gaussian Processes," *Machine Learning*, vol. 110, no. 11, pp. 3095–3133, 2021.

[10] A. Ahmadianshalchi, S. Belakaria, and J. R. Doppa, "Preference-aware constrained multi-objective Bayesian optimization," in *Proceedings of the 7th Joint International Conference on Data Science & Management of Data*, 2024, pp. 182–191.

[11] S. Takeno, M. Nomura, and M. Karasuyama, "Towards practical preferential Bayesian optimization with skew gaussian processes," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

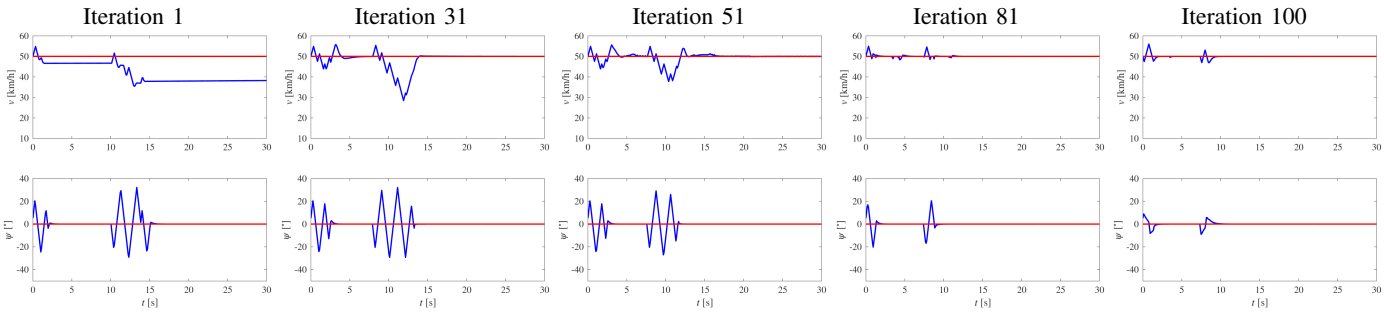


Fig. 4: Closed-loop performance of the Subject Vehicle controlled by the MPC with parameters computed by D-GLISp at iterations 1, 31, 51, 81, and 100. Reference trajectories (red); longitudinal velocity $v$ (top, blue line); steering angle $\psi$ (bottom, blue line) of the Subject Vehicle.


[12] A. Bemporad and D. Piga, "Global optimization based on active preference learning with radial basis functions," *Machine Learning*, vol. 110, pp. 417–448, 2021.

[13] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[14] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Communication-efficient federated learning," *Proceedings of the National Academy of Sciences*, vol. 118, no. 17, 2021.

[15] M. T. Young, J. D. Hinkle, R. Kannan, and A. Ramanathan, "Distributed Bayesian optimization of deep reinforcement learning algorithms," *Journal of Parallel and Distributed Computing*, vol. 139, pp. 43–52, 2020.

[16] Z. Li, Z. Dong, Z. Liang, and Z. Ding, "Surrogate-based distributed optimisation for expensive black-box functions," *Automatica*, vol. 125, 2021.

[17] L. Cannelli, M. Zhu, F. Farina, A. Bemporad, and D. Piga, "Multi-agent active learning for distributed black-box optimization," *IEEE Control Systems Letters*, vol. 7, pp. 1488–1493, 2023.

[18] M. Mohri, A. T. Suresh, and A. Tewari, "Federated learning with pairwise comparisons," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 10 015–10 025.

[19] H. Zhao, Q. Yao, Y. Zhang, and J. T. Kwok, "Decentralized dueling bandits: Efficient algorithms with convergence guarantees," in *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, vol. 84, 2018, pp. 1446–1455.

[20] R. Carli, G. Cavone, G. Notarstefano, and S. Zampieri, "A consensus-based approach for distributed ranking from pairwise comparisons," in *52nd IEEE Conference on Decision and Control*, 2013.

[21] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[22] H. Gutmann, "A radial basis function method for global optimization," *Journal of Global Optimization*, vol. 19, pp. 201–2227, 2001.

[23] D. McDonald, W. Grantham, W. Tabor, and M. Murphy, "Global and local optimization using radial basis function response surface models," *Applied Mathematical Modelling*, vol. 31, no. 10, pp. 2095–2110, 2007.

[24] A. Bemporad, "Global optimization via inverse distance weighting and radial basis functions," *Computational Optimization and Applications*, vol. 77, pp. 571–595, 2020.

[25] G. Carnevale, F. Farina, I. Notarnicola, and G. Notarstefano, "Distributed online optimization via gradient tracking with adaptive momentum," *arXiv preprint arXiv:2009.01745*, 2020.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[28] R. L. Graham, T. S. Woodall, and J. M. Squyres, "Open MPI: A flexible high performance MPI," *Parallel Processing and Applied Mathematics, Poznań, Poland*, pp. 228–239, 2006.

[29] F. Farina, A. Camisa, A. Testa, I. Notarnicola, and G. Notarstefano, "Disropt: a python framework for distributed optimization," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2666–2671, 2020.

[30] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[31] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," *Fourth International Symposium on Information Processing in Sensor Networks*, pp. 63–70, 2005.

[32] M. Zhu, D. Piga, and A. Bemporad, "C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration," *IEEE Trans. on Control Systems Technology*, vol. 30, no. 5, pp. 2176–2187, 2022.

[33] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[34] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.