C-GLISp: Preference-Based Global Optimization under Unknown Constraints with Applications to Controller Calibration

Mengjia Zhu, *Member, IEEE*, Dario Piga, *Member, IEEE*, and Alberto Bemporad, *Fellow, IEEE*

***Abstract*— Preference-based global optimization algorithms minimize an unknown objective function only based on whether the function is better, worse, or similar for given pairs of candidate optimization vectors. Such optimization problems arise in many real-life examples, such as finding the optimal calibration of the parameters of a control law. The calibrator can judge whether a particular combination of parameters leads to a better, worse, or similar closed-loop performance. Often, the search for the optimal parameters is also subject to unknown constraints. For example, the vector of calibration parameters must not lead to closed-loop instability. This paper extends an active preference learning algorithm introduced recently by the authors to handle unknown constraints. The proposed method, called C-GLISp, looks for an optimizer of the problem only based on *preferences* expressed on pairs of candidate vectors, and on whether a given vector is reported *feasible* and/or *satisfactory*. C-GLISp learns a surrogate of the underlying objective function based on the expressed preferences, and a surrogate of the probability that a sample is feasible and/or satisfactory based on whether each of the tested vectors was judged as such. The surrogate functions are used iteratively to propose a new candidate vector to test and judge. Numerical benchmarks and a semi-automated control calibration task demonstrate the effectiveness of C-GLISp, showing that it can reach near-optimal solutions within a small number of iterations.**

***Index Terms*— Active preference learning, Global Optimization with Unknown Constraints, Model predictive control**

I. Introduction

Active learning algorithms for black-box global optimization problems have been studied since the sixties under different names [1–4]. These algorithms solve the problem by optimizing a surrogate of the objective function, which is estimated by exploring the space of the optimization variables. In particular, nowadays *Bayesian Optimization* (BO) [5] is widely used to solve problems in which the cost function can only be quantified after running an experiment, such as in experimental controller calibration [6] and in automated machine learning [7]. The main idea of such methods is to fit a surrogate function to the available observations and iteratively suggest the next query point by optimizing an acquisition function. The latter trades off between exploiting the surrogate function to improve the objective and scouting unexplored areas of the search domain. An alternative surrogate method to BO based on estimating the underlying objective by radial basis functions and inverse distance weighting for exploration, called GLIS, was recently proposed in [8].

Successful applications of global optimization algorithms based on active learning for the calibration of Model Predictive Control (MPC), PID, and state-feedback control laws were presented in [9–12]. In this context, the tuning parameters of the controller are the optimization variables, and a quantitative characterization of the resulting closed-loop performance after running a simulation or experiment is the objective to optimize. These algorithms were also used for model selection [13, 14], for controller tuning in robotic manipulation and trajectory tracking [15–17], in optimizing gait parameters in robotic bipedal locomotion [18], and for "safe" optimization of position controller parameters of quadrotors [19].

A limitation of black-box optimizers like BO and GLIS is that they require quantifying an objective function after running an experiment. However, many real-world controller calibration problems involve multiple objectives to optimize, such as settling time, overshoots, actuation effort, computational burden, and other performance-related metrics. The relative weights of such objectives can be hard to assign, and sometimes even impossible to quantify, as they are the result of a qualitative judgment. On the other hand, a skilled calibrator can often assess closed-loop performance of certain tuning combinations in terms of "this test was better than the other one," *i.e.*, by pairwise comparisons. Thus, when quantifying an objective function is difficult or impossible, one can instead use these expressed *preferences* to learn an underlying surrogate function to be optimized, which leads to the area of preference-based learning algorithms.

Preference-based Bayesian optimization (PBO) has been proposed in [20–23]. Preference-based *reinforcement learning* (RL) has also drawn much attention in recent years [24]. The reader is referred to the survey paper [25] for a comprehensive review. Note that sample efficiency, which is related to the credit assignment task in RL, is a major challenge in many preference-based RL methods [25]. A more sample-efficient active preference learning method, called GLISp (an extension of GLIS), was proposed in [26]. GLISp learns a surrogate

This paper was partially supported by the Italian Ministry of University and Research under the PRIN'17 project "Data-driven learning of constrained control systems" , contract no. 2017J89ARP.

M. Zhu and A. Bemporad are with IMT School for Advanced Studies Lucca, 55100 Lucca, Italy (e-mail: mengjia.zhu@imtlucca.it; alberto.bemporad@imtlucca.it).

D. Piga is with IDSIA Dalle Molle Institute for Artificial Intelligence, SUPSI-USI, 6962 Lugano, Switzerland (e-mail: dario.piga@supsi.ch).

of the underlying preference relations by solving a Quadratic Programming (QP) problem, whose constraints reflect the expressed *preference* on whether a specific candidate is better than the other. Then, the algorithm iteratively proposes a new candidate for testing to the decision-maker for comparison. This experiment-driven preference-based approach was tested in [27] for semi-automated MPC calibration (automatic selection of control parameters, manual assessment of performance by comparisons), demonstrating its effectiveness in terms of the number of experiments needed to reach near-optimal closed-loop performance.

Real-life control design problems often involve constraints that are unknown beforehand, or for which it is not possible to find an explicit analytic expression. This is a challenge since safe exploration can be essential in many control applications, and infeasible experiments can be dangerous and costly. Several methods have been proposed in the literature to handle unknown constraints and encourage safe exploration. In [28], Sui *et al.* presented a stagewise safe BO with Gaussian processes, which they later extended to allow multiple safety constraints independent of the objective function [29]. A general formulation for constrained BO and a modified version of the expected improvement acquisition function was illustrated in [30], which handles noisy constraint observations and considers cases in which the objective and constraint functions are decoupled. In [31], a *sequential model-based optimization method* was proposed. The unknown feasible region boundaries are first reconstructed from data through *support vector machines*. Then, a global optimization step is performed via BO. Differently from the aforementioned methods, GLISp accounts for unknown constraints *implicitly* in the preferences expressed by the decision maker by making the samples that are infeasible lose the pairwise comparisons against the feasible ones.

This paper extends GLISp to handle unknown constraints in the active learning phase *explicitly*, therefore encouraging safe exploration. Besides expressing preferences, the decision-maker is asked to label an experiment as feasible and if its outcome is overall satisfactory (yes/no). Based on such labels, a surrogate of the probability of constraint feasibility and experiment's satisfaction is learned via an *Inverse Distance Weighting* (IDW) interpolant function [8]. The surrogates are properly integrated within the acquisition function to find the next point to test. We show the efficiency and effectiveness of the proposed method, called C-GLISp, in three numerical benchmarks, and on an extension of the case study originally proposed in [27] on semi-automated MPC calibration for autonomous driving. MATLAB and Python implementations of C-GLISp are also provided and available at `http://cse.lab.imtlucca.it/~bemporad/glis`.

The rest of the paper is organized as follows. The problem of preference-based optimization with unknown constraints is formulated in Section II. The proposed active-learning algorithm and details for its practical implementation are discussed in Section III. Numerical benchmarks showing the properties and the effectiveness of the proposed method are reported in Section IV, while the case study on semi-automated MPC calibration for autonomous driving is presented in Section V. Conclusions and directions for future research are drawn in Section VI.

II. Problem Formulation

Let $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ be the space of decision vectors x. We are interested in minimizing an (unknown) objective function $f : \mathcal{D} \to \mathbb{R}$ subject to the constraint that x belongs to an (unknown) feasibility set $\Omega_G \subseteq \mathcal{D}$.

We assume that we cannot represent the set Ω_G, but rather that, given a vector $x \in \mathcal{D}$, a decision-maker can assess the value of the *feasibility function* $G : \mathcal{D} \to \{0, 1\}$ defined as

$$G(x) = \begin{cases} 0 & \text{if } x \notin \Omega_G \\ 1 & \text{if } x \in \Omega_G. \end{cases} \tag{1}$$

In other words, a value x outside Ω_G will be considered as "unacceptable" ($G(x) = 0$) by the decision-maker. For example, an unacceptable x can be a set of controller parameters leading to an unstable closed-loop behavior or to a control law that is too expensive to compute in real-time.

Furthermore, we assume that the objective function f cannot be directly quantified, but rather that can be indirectly observed in two ways:

1. For a given sample $x \in \mathcal{D}$, the decision-maker is requested to say whether or not x leads to certain "satisfactory performance". Formally, we can define a *satisfaction set* $\Omega_S \subseteq \mathcal{D}$ and a *satisfaction function* $S : \mathcal{D} \to \{0, 1\}$ as
$$S(x) = \begin{cases} 0 & \text{if } x \notin \Omega_S \\ 1 & \text{if } x \in \Omega_S, \end{cases} \tag{2}$$
where the set Ω_S contains all the vectors x leading to a performance that the decision-maker judges satisfactory. An analytic expression of Ω_S is therefore not available, only the value $S(x)$ is provided by the decision-maker for any given $x \in \mathcal{D}$. Note that Ω_S may not be a subset of Ω_G, for example when the preference-based optimization process is carried out in simulation: a sample may lead to satisfactory performance but would not be implementable due to hardware limitations. On the other hand, in cases of assessments based on physical experiments, Ω_S is necessarily a subset of Ω_G, as no performance would be available for evaluation when the parameters are infeasible.
2. For any pair $x_1, x_2 \in \mathcal{D}$, the decision-maker is requested to provide the output of the *preference function*: $\pi : \mathcal{D} \times \mathcal{D} \to \{-1, 0, 1\}$
$$\pi(x_1, x_2) = \begin{cases} -1 & \text{if } x_1 \text{ "better" than } x_2 \\ 0 & \text{if } x_1 \text{ "as good as" } x_2 \\ 1 & \text{if } x_2 \text{ "better" than } x_1. \end{cases} \tag{3}$$
where the preference in (3) is implicitly defined according to the underlying hidden function f to be minimized, namely
$$\pi(x_1, x_2) = \begin{cases} -1 & \text{if } f(x_1) < f(x_2) \\ 0 & \text{if } f(x_1) = f(x_2) \\ 1 & \text{if } f(x_1) > f(x_2). \end{cases} \tag{4}$$

The rationale behind the above problem formulation is that often one encounters practical decision problems in which a

function f is impossible to quantify, but anyway it is possible for a human operator to express a qualitative evaluation (*e.g.*, "good" or "bad") and a preference between the outcome of two experiments.

Formally, we want to find the optimal solution $x^\star \in \Omega_S \cap \Omega_G$ such that $x^\star$ is "better" (or "no worse") than any other x according to the preference function π:

$$\text{find } x^\star \text{ such that } \pi(x^\star, x) \leq 0, \ \forall x \in \Omega_S \cap \Omega_G. \tag{5}$$

We propose to solve problem (5) iteratively as follows: (i) suggest a sequence of decision vectors $x_1, \ldots, x_N \in \mathcal{D}$ to test, (ii) ask to evaluate the feasibility function $G(x_i)$ and the satisfaction function $S(x_i)$ for $i = 1, \ldots, N$, and (iii) ask to evaluate the preference function $\pi(x_i, x_j)$ for M given pairs (i, j), $i, j = 1, \ldots, N$, $i \neq j$, where M is the number of expressed preferences, $1 \leq M \leq \binom{N}{2}$. The goal is to propose candidate vectors x_N approaching the optimal solution $x^\star$ as N grows.

III. Proposed Method

The proposed preference-based optimization method to solve problem (5) is based on an extension of the GLISp algorithm originally introduced in [26]. We refer to the new algorithm as C-GLISp, whose aim is to handle unknown constraints expressed in terms of an approximation of the feasibility function G in (1) and of the satisfaction function S in (2).

Similarly to GLISp, C-GLISp involves two main phases: an initial random sampling and an active learning phase. In both phases, C-GLISp trains and updates three surrogate functions approximating, respectively, the feasibility function G, the satisfaction function S, and the underlying function f. During the active learning phase, the next point for evaluation is selected by optimizing an acquisition function which trades off *exploitation* (optimization only based on the surrogates describing the observed preferences and constraints) and *exploration* (searching unexplored areas of the domain $\mathcal{D}$). The goal of C-GLISp is to approach an optimal solution $x^\star$ as in (5) within a small number N of experiments.

A. Learning Unknown Constraint Functions

We discuss how to train surrogates of the functions G and S that approximate, respectively, the feasibility constraint $x \in \Omega_G$ and the satisfaction constraint $x \in \Omega_S$. The idea is to ask the decision-maker to assess, once an experiment is performed, whether the constraints $x \in \Omega_G$ and $x \in \Omega_S$ are satisfied or not, and train surrogate functions of G and S based on the outcome of $N \geq 2$ of such queries. These queries are performed on a set of samples $\{x_1, \ldots, x_N\}$ iteratively proposed by C-GLISp.

Compared to unconstrained preference-based optimization like GLISp, in which an infeasible/unsatisfactory sample only indirectly reveals itself as such by losing pairwise comparisons against feasible/satisfactory ones, C-GLISp exploits the information on whether $x \in \Omega_G$ and/or $x \in \Omega_S$ to facilitate the optimization process, in particular, to avoid exploring the infeasible and/or unsatisfactory region and therefore reduce the number of samples $x_i \notin \Omega_G$ and/or $x_i \notin \Omega_S$.

The surrogate functions for G and S are constructed as follows. The decision-maker observes the outcome of the performed experiments, and he/she provides a *feasibility vector* $G_F = [G_1 \ \ldots \ G_N]' \in \{0, 1\}^N$ with

$$G_i = G(x_i), \tag{6}$$

and a *satisfaction vector* $S_F = [S_1 \ \ldots \ S_N]' \in \{0, 1\}^N$ with

$$S_i = S(x_i), \tag{7}$$

by assessing whether each experiment is feasible and satisfactory. Then, surrogates $\hat{G}$ of G and $\hat{S}$ of S are constructed from the observations G_F and S_F, respectively, as detailed below.

A surrogate function $\hat{G} : \mathcal{D} \to \mathbb{R}$ predicting the probability of satisfying the feasibility constraint $x \in \Omega_G$ is defined as

$$\hat{G}(x) = \sum_{i=1}^{N} \nu_i(x) G_i, \tag{8}$$

where $\nu_i(x) : \mathcal{D} \to \mathbb{R}$ for $i = 1 \ldots, N$ is defined as

$$\nu_i(x) = \begin{cases} 1 & \text{if } x = x_i \\ 0 & \text{if } x = x_j, j \neq i \\ \frac{w_i(x)}{\sum_{i=1}^{N} w_i(x)} & \text{otherwise.} \end{cases} \tag{9}$$

Here $w_i : \mathcal{D} \setminus \{x_i\} \to \mathbb{R}$ is the following IDW function [32]

$$w_i(x) = \frac{e^{-d^2(x, x_i)}}{d^2(x, x_i)}, \tag{10}$$

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

$$d(x, x_i) = \|x - x_i\|_2^2. \tag{11}$$

The surrogate function $\hat{S} : \mathcal{D} \to \mathbb{R}$ is defined similarly. The approach presented in this paper aims at solving problems where experiments are expensive to run, so that data efficiency is essential. IDW interpolation functions are selected in this case because of their high accuracy. Other binary classification methods (*e.g.* logistic regression or random forests) would be less suitable in this context since their accuracy with a small number of training data is limited. Support vector machines (SVMs) [33] can be a potential substitute since they work well with small and medium-size training sets. However, our numerical tests have shown that IDW interpolation functions outperform SVM. In addition, the functions $\hat{G}$ and $\hat{S}$ generated by IDW interpolation are always between 0 and 1 by construction (see [8, Lemma 1-P2]), and can be interpreted as probabilities of being feasible/satisfactory.

B. Learning The Preference Function

Radial basis functions (RBFs) [34, 35] are flexible and have been adopted to solve global optimization problems in [8, 34–37] with success. Therefore, as in [26], we parameterize the surrogate function $\hat{f} : \mathcal{D} \to \mathbb{R}$ as a linear combination of RBFs [34, 35]

$$\hat{f}(x) = \sum_{k=1}^{N} \beta_k \phi(\epsilon d(x, x_i)), \tag{12}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is an RBF, $\epsilon > 0$ is a scalar hyper-parameter defining the shape of the RBF, and $\beta = [\beta_1 \ \dots \ \beta_N]^T$ are the unknown coefficients to be determined. Some RBFs commonly used are $\phi(\epsilon d) = \frac{1}{1+(\epsilon d)^2}$ (*inverse quadratic*), $\phi(\epsilon d) = e^{-(\epsilon d)^2}$ (*Gaussian*), and $\phi(\epsilon d) = (\epsilon d)^2 \log(\epsilon d)$ (*thin plate spline*).

Besides the *feasibility vector* G_F and the satisfaction vector S_F, the *preference vector* $B = [b_1 \ \dots \ b_M]^T \in \{-1, 0, 1\}^M$ is also assumed to be provided by the decision-maker, with

$$b_h = \pi(x_{i(h)}, x_{j(h)}), \tag{13}$$

for $x_i, \ x_j \in \mathcal{D}$ such that $x_i \neq x_j$, $\forall i \neq j$, $i, j = 1, \dots, N$. Here, M is the number of expressed preferences, $1 \leq M \leq \binom{N}{2}$, $h \in \{1, \dots, M\}$ is the index enumerating the preferences, and $i(h), j(h) \in \{1, \dots, N\}$, $i(h) \neq j(h)$.

The preferences b_h expressed by the decision-maker are used to shape the surrogate objective function $\hat{f}$ by imposing the following constraints:

$$\begin{array}{ll} \hat{f}(x_{i(h)}) \leq \hat{f}(x_{j(h)}) - \sigma & \text{if } b_h = -1 \\ \hat{f}(x_{i(h)}) \geq \hat{f}(x_{j(h)}) + \sigma & \text{if } b_h = 1 \\ |\hat{f}(x_{i(h)}) - \hat{f}(x_{j(h)})| \leq \sigma & \text{if } b_h = 0 \end{array} \tag{14}$$

for $h = 1, \dots, M$, where $\sigma > 0$ is a given scalar that avoids the trivial solution $\hat{f}(x) \equiv 0$.

Similarly to SVMs [33], the vector β of coefficients describing the surrogate $\hat{f}$ is obtained by solving the following convex QP problem

$$\begin{aligned} \min_{\beta, \varepsilon} \quad & \sum_{h=1}^{M} c_h \varepsilon_h + \frac{\lambda}{2} \sum_{k=1}^{N} \beta_k^2 \\ \text{s.t.} \quad & \sum_{k=1}^{N} \beta_k (\phi(\epsilon d(x_{i(h)}, x_k)) - \phi(\epsilon d(x_{j(h)}, x_k))) \\ & \qquad \leq -\sigma + \varepsilon_h, \qquad \forall h: \ b_h = -1 \\ & \sum_{k=1}^{N} \beta_k (\phi(\epsilon d(x_{i(h)}, x_k)) - \phi(\epsilon d(x_{j(h)}, x_k))) \\ & \qquad \geq \sigma - \varepsilon_h, \qquad \forall h: \ b_h = 1 \\ & \left| \sum_{k=1}^{N} \beta_k (\phi(\epsilon d(x_{i(h)}, x_k)) - \phi(\epsilon d(x_{j(h)}, x_k))) \right| \\ & \qquad \leq \sigma + \varepsilon_h, \qquad \forall h: \ b_h = 0 \\ & h = 1, \dots, M \end{aligned} \tag{15}$$

that captures the preference relationships in (14) and the parametrization of $\hat{f}$ in (12). In (15), c_h are positive weights and ε_h are positive slack variables used to relax the constraints imposed by (14). The violation of the imposed constraints could be caused by an inappropriate selection of the RBF, leading to poor flexibility in the parametric description of the surrogate function $\hat{f}$, and by inconsistent assessments provided by the decision-maker. The scalar $\lambda > 0$ in the cost function (15) is a regularization parameter. With $\lambda > 0$, problem (15) is a QP problem that admits a unique solution.

In the constrained preference-based optimization algorithm detailed in the following section, we will execute K-fold cross-validation periodically (*i.e.*, when i is in the predefined self-calibration index set $\mathcal{I}_{\text{sc}} \subseteq \{1, \dots, N_{\max} - 1\}$) to automatically tune the hyper-parameter ϵ defining the shape of the RBF in (12) during the active learning phase, as recommended in [26].

C. Acquisition Function

Minimizing $\hat{f}$ greedily to generate the next sample x_{N+1} may lead the solver to converge to a point that is not the global optimum of (5). Hence, when selecting the next point x_{N+1}, besides *exploiting* the surrogate $\hat{f}$, some *exploration* should be considered to search regions with limited/no samples to reduce the uncertainty associated with $\hat{f}$. Also, the feasibility and satisfactory regions are unknown and are only *implicitly* included in the surrogate function $\hat{f}$. Therefore, we also include terms to *explicitly* avoid the *exploration* in the regions with low probabilities of being feasible and satisfactory by penalizing the (estimated) infeasibility $x \notin \Omega_G$ and unsatisfactory performance $x \notin \Omega_S$.

The exploration term used in GLISp is the following IDW function $z : \mathcal{D} \to \mathbb{R}$

$$z(x) = \begin{cases} 0 & \text{if } x \in \{x_1, \dots, x_N\} \\ \tan^{-1}\left(\frac{1}{\sum_{i=1}^{N} r_i(x)}\right) & \text{otherwise,} \end{cases} \tag{16}$$

where $r_i(x) = \frac{1}{d^2(x, x_i)}$. Note that $z(x) = 0$ for all the decision variables x already sampled and tested, and $z(x) > 0$ in $\mathcal{D} \setminus \{x_1, \dots, x_N\}$. The *arc tangent* function is used to prevent the new sampled point from getting excessively far away from the existing ones.

Unlike GLISp, here we modify (16) into

$$\begin{aligned} z_N(x) = & \left(1 - \frac{N}{N_{max}}\right) \tan^{-1}\left(\frac{\sum_{i=1}^{N} r_i(x_N^*)}{\sum_{i=1}^{N} r_i(x)}\right) \\ & + \frac{N}{N_{max}} \tan^{-1}\left(\frac{1}{\sum_{i=1}^{N} r_i(x)}\right) \end{aligned} \tag{17}$$

for $x \notin \{x_1, \dots, x_N\}$ and $z_N(x) = 0$ otherwise. In (17), x_N^* is the best decision variable found up to iteration N. In (17), N_{max} is the maximum allowed number of experiments. The rationale behind (17) is that it encourages the exploration of regions of $\mathcal{D}$ further away from the current best solution in the early iterations and reduce its effects as the number N of experiments increases. The exploration function in (17) was empirically observed to better escape from local minima than that in (16).

The *acquisition function* $a : \mathcal{D} \to \mathbb{R}$ is defined as

$$\begin{aligned} a(x) = & \frac{\hat{f}(x)}{\Delta \hat{F}} - \delta_E z_N(x) \\ & + \delta_G(1 - \hat{G}(x)) + \delta_S(1 - \hat{S}(x)), \end{aligned} \tag{18}$$

where $\delta_E \geq 0$ is the exploration parameter, and $\delta_G, \delta_S \geq 0$ weight the probability of a sample x to be infeasible and/or unsatisfactory, respectively. The term $\Delta \hat{F} = \max_i\{\hat{f}(x_i)\} - \min_i\{\hat{f}(x_i)\}$ is the range of the surrogate function $\hat{f}$ on the samples in $\{x_1, \dots, x_N\}$. It is used as a scaling factor in (18) to make each term in (18) comparable, which eases the selection of the hyper-parameters δ_E, δ_G, and δ_S.

The exploration parameter δ_E encourages sampling unexplored regions of the domain $\mathcal{D}$. Setting $\delta_E = 0$ makes C-GLISp rely heavily on the accuracy of the surrogate functions

$\hat{f}$, $\hat{G}$, and $\hat{S}$, which may easily lead to missing the global optimum. On the other hand, setting $\delta_E \gg 1$ leads C-GLISp to explore the entire domain $\mathcal{D}$ regardless of the decision-maker's preferences and feasibility/satisfaction assessments.

Functions $\hat{G}$ and $\hat{S}$ in (18) aim at discouraging exploration in regions where the experiment is predicted to be infeasible (*i.e.*, $x \notin \Omega_G$) and/or unsatisfactory (*i.e.*, $x \notin \Omega_S$). Therefore, a poor selection of the hyperparameters δ_G and δ_S and/or a poor predictive capability of $\hat{G}$ and $\hat{S}$ (*e.g.*, due to a limited number of samples) can prevent finding new vectors that are actually feasible and/or satisfactory. To alleviate this issue, we suggest to adaptively tune δ_G and δ_S based on the sampled standard deviation obtained from leave-one-out cross-validation [38] of $\hat{G}$ and $\hat{S}$, respectively. More specifically, each available sample in the set $\{x_1, \ldots, x_N\}$ is used once as a testing point and the remaining ones are used to train $\hat{G}$ and $\hat{S}$. The prediction $\hat{G}(x_i)$ and $\hat{S}(x_i)$ on the test sample x_i is compared with the corresponding labels $G(x_i)$ and $S(x_i)$ assigned by the decision-maker to compute the following sampled standard deviations of the error:

$$\hat{\sigma}_G = \min\left\{1,\ \sqrt{\frac{\sum_{i=1}^{N}(\hat{G}(x_i) - G(x_i))^2}{N-1}}\right\},$$
$$\hat{\sigma}_S = \min\left\{1,\ \sqrt{\frac{\sum_{i=1}^{N}(\hat{S}(x_i) - S(x_i))^2}{N-1}}\right\}. \tag{19}$$

The sampled standard deviations are then used to update, after each iteration, the weights δ_G and δ_S as follows:

$$\delta_G = (1 - \hat{\sigma}_G)\delta_{G,\text{default}}, \tag{20a}$$
$$\delta_S = (1 - \hat{\sigma}_S)\delta_{S,\text{default}}, \tag{20b}$$

where $\delta_{G,\text{default}}$ and $\delta_{S,\text{default}}$ are default values set by the user. Clearly, one should select $\delta_{G,\text{default}} > \delta_{S,\text{default}}$, so that infeasibility is penalized more than unsatisfactory behavior. The updated values of δ_G and δ_S are then used to construct the acquisition function $a(x)$ in (18).

The next sample x_{N+1} to test is obtained by minimizing $a(x)$, *i.e.*,

$$x_{N+1} = \arg\min_{x \in \mathcal{D}} a(x). \tag{21}$$

Different optimization methods can be used to solve problem (21) efficiently either via derivative-free [39], or derivative based algorithms.

C-GLISp updates the surrogates $\hat{f}$, $\hat{G}$, and $\hat{S}$, and the exploration function $z_N(x)$, by iteratively suggesting a new point x_{N+1} to test, and by receiving feedback from the decision-maker in terms of feasibility, overall satisfaction, and preferences between pairs of experiments. Algorithm 1 summarizes the proposed method.

Algorithm 1 C-GLISp: Preference learning algorithm with unknown constraint handling

Input: Lower and upper bounds (ℓ, u), known constraint set if available; number $N_{\text{init}} \geq 2$ of initial samples, number $N_{\max} \geq N_{\text{init}}$ of maximum function evaluations; $\delta_E \geq 0$, $\delta_{G,\text{default}} \geq 0$ and $\delta_{S,\text{default}} \geq 0$; $\sigma > 0$; and $\epsilon > 0$; self-calibration index set $\mathcal{I}_{\text{sc}} \subseteq \{1, \ldots, N_{\max} - 1\}$.

1. Generate N_{init} random samples $X = \{x_1, \ldots, x_{N_{\text{init}}}\}$ using Latin hypercube sampling method [40];
2. $N \leftarrow 1$, $i^\star \leftarrow 1$;
3. **While** $N < N_{\max}$ **do**

3.1. **if** $N = 1$ **then**

3.1.1. Observe feasibility G_N and satisfaction S_N;

3.2. **if** $N \geq N_{\text{init}}$ **then**

3.2.1. **if** $N \in \mathcal{I}_{\text{sc}}$ **then** recalibrate ϵ through K-fold cross-validation;

3.2.2. Solve (15) to obtain β to define the surrogate function $\hat{f}$ (12);

3.2.3. Update δ_G and δ_S as in (20);

3.2.4. Define acquisition function a as in (18);

3.2.5. Solve optimization problem (21) and get x_{N+1};

3.3. $i(N) \leftarrow i^\star$, $j(N) \leftarrow N + 1$;

3.4. Observe feasibility $G_{j(N)}$. satisfaction $S_{j(N)}$ and preference $b_N = \pi(x_{i(N)}, x_{j(N)})$;

3.5. **if** $b_N = 1$ **then set** $i^\star \leftarrow j(N)$;

3.6. $N \leftarrow N + 1$;

4. **End**.

Output: Computed best input $x^\star = x_{i^\star}$.

IV. Optimization Benchmarks

We test C-GLISp on three constrained global optimization benchmarks to illustrate its effectiveness in solving optimization problems with unknown constraints. Computations are performed on an Intel i7-8550U 1.8-GHz CPU laptop with 8GB of RAM. The Latin hypercube sampling method [40] (*lhsdesign* function of the *Statistics and Machine Learning Toolbox* of MATLAB [44]) is used in the initial sampling phase of C-GLISp. Particle Swarm Optimization (PSO) [45] is used to minimize the acquisition function as in (21).

C-GLISp is compared to the original GLISp and to PBO (with *expected improvement* as acquisition function) [21, Section 2.3]. For numerical benchmarks, C-GLISp, GLISp, and PBO assign the preferences on pairwise comparisons based on the combined assessments of the objective function value, feasibility, and performance satisfaction. For each test function, depending on the problem formulation, a maximum of three types of queries is obtained when using C-GLISp, which are the preference relation (B), the feasibility label (G_F), and the satisfaction label (S_F). In contrast, GLISp and PBO only rely on the preference relation B. The goal of the comparison between C-GLISp and GLISp is to check if accounting the feasibility and/or satisfactory information explicitly in the acquisition function can encourage safe exploration from the comparisons. It is worth noting that the exact evaluation of the objective function and the constraints (feasibility and satisfactory outcomes) for these numerical benchmarks are unknown to the algorithms and are only used to construct a synthetic decision-maker.

Table I lists the specifications of the benchmarks. The original feasibility set of the test function *Mishra's Bird function-constrained* (MBC) [41, 42] is modified so that the uncon-

TABLE I
NUMERICAL BENCHMARKS - PROBLEM SPECIFICATION

Test function	Objective function	Unknown constraints	Search domain $\mathcal{D}$
Mishra's Bird function-constrained (modified) [41, 42] (MBC)	$f(x,y) = \sin(y)e^{(1-\cos(x))^2} + \cos(x)e^{(1-\sin(y))^2} + (x-y)^2$	Feasibility constraints: $(x+9)^2 + (y+3)^2 < 9$	$[-10.0, -2]$; $[-6.5, 0.0]$
camelsixhumps-hard constrained [8, 43] (CHC)	$f(x,y) = (4 - 2.1x^2 + x^{4/3})x^2 + xy + (4y^2 - 4)y^2$	Feasibility constraints: $g_1 \cap g_2$ $g_1 : \begin{bmatrix} 1.6295 & 1 \\ -1 & 4.4553 \\ -4.3023 & -1 \\ -5.6905 & -12.1374 \\ 17.6198 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} < \begin{bmatrix} 3.0786 \\ 2.7417 \\ -1.4909 \\ 1 \\ 32.5198 \end{bmatrix}$ $g_2 : x^2 + (y + 0.1)^2 < 0.5$	$[-2, 2]$; $[-1, 1]$
camelsixhumps-hard and soft constrained [8, 43] (CHSC)	$f(x,y) = (4 - 2.1x^2 + x^{4/3})x^2 + xy + (4y^2 - 4)y^2$	Feasibility constraints: g_2 Satisfaction constraints: g_1 $g_1 : \begin{bmatrix} 1.6295 & 1 \\ 0.5 & 3.875 \\ -4.3023 & -4 \\ -2 & 1 \\ 0.5 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} < \begin{bmatrix} 3.0786 \\ 3.324 \\ -1.4909 \\ 0.5 \\ 0.5 \end{bmatrix}$ $g_2 : x^2 + (y + 0.04)^2 < 0.8$	$[-2, 2]$; $[-1, 1]$

TABLE II
NUMERICAL BENCHMARKS - SOLVER SPECIFICATION

Test function	Max number of fun. eval. N_{max}	Number of initial sampling N_{init}	Hyper-parameter values			RBF specifications (12)			Tolerance	Weights	Regularization
			δ_E	$\delta_{G,\text{default}}$	$\delta_{S,\text{default}}$	function	initial ϵ	recalibration steps	σ in (15)	c_h in (15)	λ in (15)
MBC	50	13	1.0	1.0	–	Inverse quadratic	1.0	$\{13, 22, 32, 41\}$	0.02	1.0	$1e$-6
CHC	100	25	2.0	2.0	–	Inverse quadratic	1.0	$\{25, 44, 63, 81\}$	0.01	1.0	$1e$-6
CHSC	50	13	1.0	1.0	0.5	Inverse quadratic	1.0	$\{13, 22, 32, 41\}$	0.02	1.0	$1e$-6

Same parameters (if relevant) are used in C-GLISp, GLISp, and PBO.

TABLE III
NUMERICAL BENCHMARKS - RESULTS

Test function	Constrained optimum[a]				Feasibility[b]		
	Optimum	C-GLISp	GLISp	PBO	C-GLISp	GLISp	PBO
MBC	-48.4	-47.95	-48.33	-40.24	100	100	91
CHC	-0.5844	-0.3582	-0.5224	0.2571	96	66	33
CHSC	-0.9050	-0.8526	-0.8861	-0.6315	96 (95)	82 (84)	74 (72)

a- The median of computed constrained optima that are feasible out of 100 runs (the distribution over 100 runs is reported in Table IV).
b- Number of runs whose computed optimizers are feasible out of 100 runs. Values in parentheses indicate the number of runs the optimizer is satisfactory.

strained global optimum in the search domain is no longer in the feasible area. The *camelsixhumps-hard constrained* (CHC) benchmark [8, 43] considers two feasibility constraints, and the unconstrained global optimum also differs from the constrained one. Lastly, the benchmark function *camelsixhumps-hard and soft constrained* (CHSC) [8, 43] has both feasibility and satisfaction constraints. The two unconstrained optima for this test function are both feasible but not satisfactory.

The values of the hyper-parameters in C-GLISp, GLISp, and PBO are provided in Table II. The number of initial samples (N_{init}) is selected as one fourth of the maximum number of function evaluations ($N_{max}/4$) rounded to the nearest integer. Three-fold cross-validation is used to update the hyper-parameter ϵ (12) at iterations N_{init}, $N_{init}+(N_{max}-N_{init})/4$, $N_{init}+(N_{max}-N_{init})/2$, and $N_{init}+3(N_{max}-N_{init})/4$, rounded to the closest integers, which define the self-calibration index set $\mathcal{I}_{\text{sc}}$. The tolerance σ in (15) is set to $1/N_{max}$. The default value $\delta_{G,\text{default}}$ in (20) is the same as δ_E, so that the feasibility term in (18) is comparable to the pure exploration term, while the default value $\delta_{S,\text{default}}$ is selected as $\delta_{G,\text{default}}/2$ to reduce its effects with respect to hard feasibility constraints. The parameters δ_G and δ_S are kept at their default values during the first N_{init} experiments, then updated each time a new point is added using equation (20). The remaining parameters of the solvers are set according to the defaults used or suggested in [26].

TABLE IV
DISTRIBUTION OVER 100 RUNS OF THE PERCENTAGE DIFFERENCE BETWEEN ACHIEVED AND GLOBAL OPTIMUM

Benchmark	Algorithm	Number of runs within each interval			
		Intervals of % Difference from Global Optimum			
		(0,5]	(5,10]	(10,15]	(15,100]
MBC	PBO	39	4	2	18
	GLISp	67	1	2	3
	C-GLISp	69	6	1	5
		(0,5]	(5,20]	(20,50]	(50,100]
CHC	PBO	0	0	4	7
	GLISp	28	7	5	1
	C-GLISp	0	20	40	22
		(0,5]	(5,10]	(10,15]	(15,100]
CHSC	PBO	13	10	4	27
	GLISp	56	8	5	9
	C-GLISp	43	22	13	16

Note: only the runs with feasible solutions within 100% difference from the global optimum are counted.

Table III reports the results obtained by running a Monte-Carlo simulation with 100 runs of C-GLISp, GLISp, and PBO to obtain statistically significant results. One of such runs of C-GLISp on all three numerical benchmarks is depicted in Fig. 1. Table IV displays the distribution over 100 runs of the percentage difference between the achieved feasible solutions and the true constrained optimum. Overall, the results from Table III and IV show that C-GLISp can find a feasible near-optimal solution more frequently than GLISp and PBO.

From the results on the benchmark function MBC, where

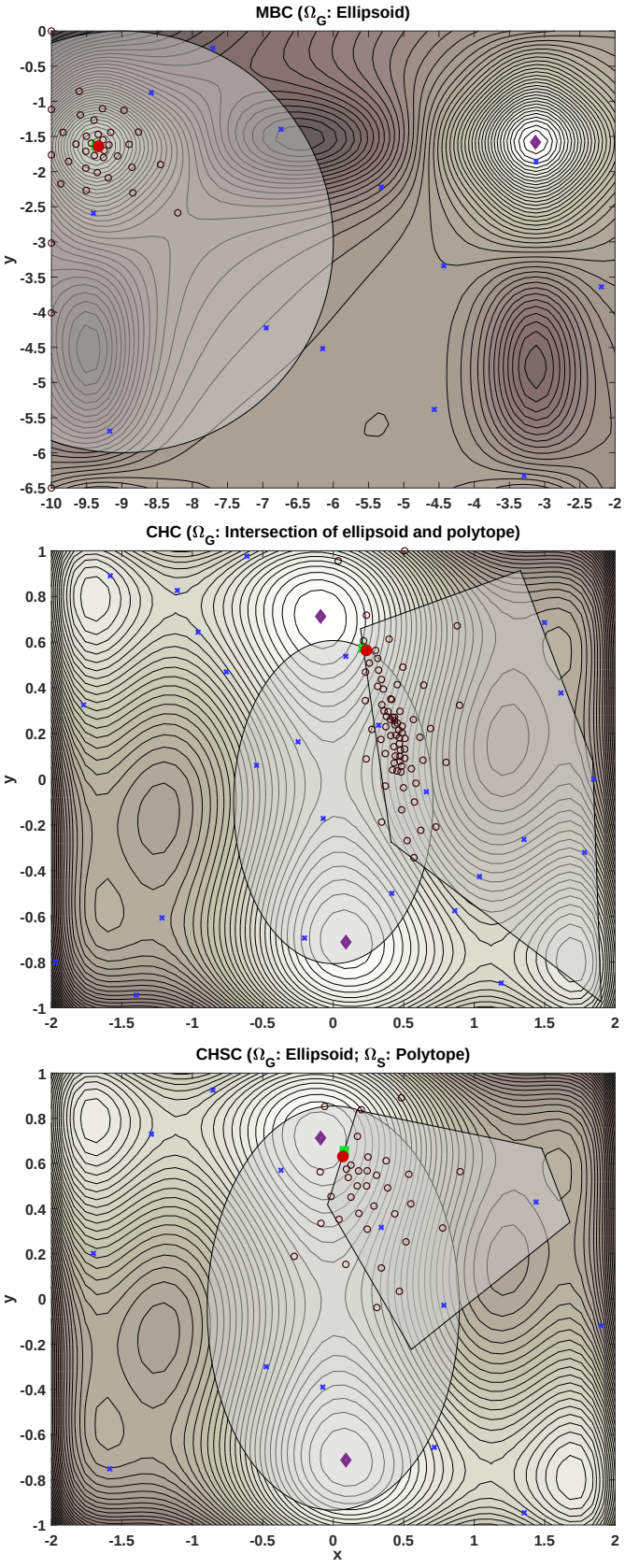

Fig. 1. Algorithm C-GLISp. Level sets of the functions used in the three benchmarks, along with feasibility and satisfaction sets. Blue $\times$: points generated from initial sampling phase; black $\circ$: points generated from active learning phase; purple $\diamondsuit$: global unconstrained optimizer; red $\bullet$: constrained optimizer found after N_{max} iterations; green $\square$: global constrained optimizer. As N increases, the points generated by C-GLISp approach the constrained optimizer, and most of the points generated during the active learning phase lay in the feasibility and satisfaction regions.

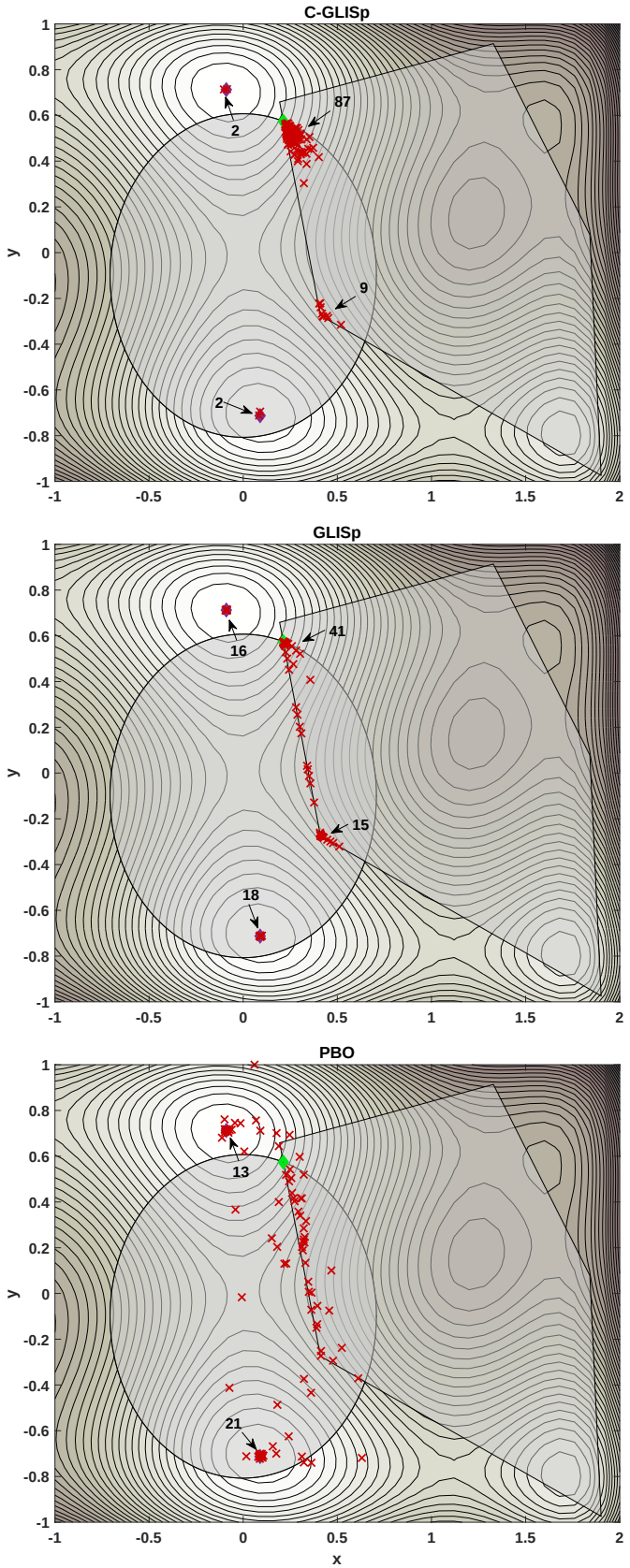

Fig. 2. Benchmark CHC. Optimizers computed by C-GLISp, GLISp and PBO in 100 runs. Red $\times$: optimizer computed at the end of each run; purple $\diamondsuit$: unconstrained optimizer; green $\diamondsuit$: global constrained optimizer. Numbers in black with arrows indicate the number of overlapping points.

the feasibility set Ω_G covers roughly one-third of the domain $\mathcal{D}$ (cf. Fig. 1), the performance of GLISp and C-GLISp are comparable. They always terminate the search with a feasible optimum (100 out of 100 runs), with 67% and 69% of them, respectively, located within 5% difference from the global solution (Table IV). On the other hand, PBO computes a feasible optimum in 91 runs, but with only around 39% within 5% difference (Table IV). When the constraint is more complex such as the one in CHC, the majority of the optima computed by C-GLISp (96 out of 100 runs) are feasible. In comparison, only 66 and 33 runs by GLISp and PBO, respectively, terminate with a feasible solution (Table III and Fig. 2). From Fig. 1, it is also observed that, for the test function CHC, after the initial sampling phase, most points generated in the active learning phase by C-GLISp are within the feasible region.

For the test function CHSC, C-GLISp often find a near-optimal solution that is both feasible and satisfactory. The performance of GLISp is slightly worse than C-GLISp in terms of the number of times a feasible and satisfactory solution is obtained. PBO can identify a feasible and satisfactory solution with a relatively high chance but still lower than both C-GLISp and GLISp. Also, its final outcome is worse, see Table III.

Table III also shows that, within the same number of iterations, the median of the computed feasible constrained optima from GLISp is always closer to the global constrained optimum than the one computed from C-GLISp. This is because of the trade-off between trying to get a more accurate solution (which is often achieved by sampling multiple points close to the current best solution up to iteration N) and exploring a larger area to reduce uncertainty (in problems with unknown constraints, C-GLISp also tries to identify possible feasible regions).

For our problem setting, we set a limit on the computational budget. Modification of the exploration term from (16) to (17) helps to better escape from local minima in the early iterations by encouraging the exploration of regions of $\mathcal{D}$ further away from the current best solution. This modification is significant for problems with small feasible regions (relative to the search domain) and/or complex unknown constraints (*e.g.*, numerical benchmark CHC). This is because the additional exploration introduced by the modification can help the solver identify the feasible region more quickly and start recommending feasible guesses faster, reducing the chance that the solver gets trapped into an infeasible local optimum.

From the number of feasible solutions computed shown in Table III and the computed optimizers displayed in Fig. 2, we observe that the situation of trapping into an infeasible local optimum occurs to GLISp more frequently than to C-GLISp. However, GLISp can achieve a solution closer to the constrained optimum than C-GLISp (Table IV) when GLISp successfully identifies the feasible region. This is because more computational budget is then used to get closer to the optimum than to explore other potentially feasible regions as in C-GLISp. For problems where testing an infeasible solution is expensive and/or dangerous, it is better to be conservative and have a less optimal but feasible solution. As a result, C-GLISp is preferred over GLISp under these problem settings.

Overall, the results on the numerical benchmarks show that both C-GLISp and GLISp can approach near-optimal solutions within a small number of function evaluations. In all of the three benchmarks, both C-GLISp and GLISp outperform PBO (Table III and IV). An explanation for the superior performance of C-GLISp in identifying feasible/satisfactory solutions is that it explicitly leverages feasibility/satisfaction information in the acquisition function, while GLISp and PBO handle unknown constraints only through preference queries.

V. MPC Calibration

To illustrate the application of C-GLISp to controller calibration, we consider the design of an MPC controller for lane-keeping (LK) and obstacle-avoidance (OA) in autonomous driving. MPC is employed to command vehicle velocity and steering angle to provide a smooth and safe drive. The same problem was considered in [27] and is extended in this paper to handle feasibility and satisfaction constraints.

The design of model predictive controllers requires tuning several knobs, such as the prediction and control horizons, the weight matrices in the cost function, numerical tolerances in the optimization solver, *etc.*, under hard constraints such as finding solutions within the sample interval of a real-time implementation. Thus, it is hard to well define in advance a single quantitative performance index that captures a usually multifaceted desired closed-loop behavior to tune the MPC parameters automatically.

We use C-GLISp to implement an iterative semi-automated calibration procedure (automatic selection of the MPC parameters and human-based qualitative assessment of closed-loop performance), thus avoiding the burden of defining a quantitative index to minimize. The calibrator only needs to express a *preference* on pairwise comparisons to indicate which of the two MPC tunings has better closed-loop performance. Different from the case study in [27], where we account for the information of feasibility and satisfaction conditions implicitly in the preference query (GLISp) as if they were, implicitly, underlying penalty functions, C-GLISp explicitly takes into account this information via direct queries as in (1) and (2).

A. System Description

We consider a simplified two-degree-of-freedom bicycle model with the front wheel as the reference point to describe the vehicle dynamics and simulate the experiment. The state variables $s = [x_f \ w_f \ \theta]'$ in the model are the longitudinal x_f and lateral w_f [m] positions of the front wheel, and the yaw angle θ [rad]. The manipulated variables $u = [v \ \psi]'$ are the commanded vehicle velocity v [m/s] and steering angle ψ [rad]. The standard kinematic equations

$$\begin{aligned} \dot{x}_f &= v\cos(\theta+\psi) \\ \dot{w}_f &= v\sin(\theta+\psi) \\ \dot{\theta} &= \frac{v\sin(\psi)}{L} \end{aligned} \tag{22}$$

are used to model the evolution of the vehicle, where L [m] is the vehicle length. Here, full state observation is assumed, *i.e.*, the control output $y = s$.

B. MPC Formulation

The formulation of the semi-automatic MPC calibration process is detailed in [27]. The dynamical model (22) is linearized around its nominal point $\bar{s}_k = [\bar{x}_{f_k} \ \bar{w}_{f_k} \ \bar{\theta}_k]'$, $\bar{u}_k = [\bar{v}_k \ \bar{\psi}_k]'$, and $\bar{y}_k = \bar{s}_k$ at each time step and discretized with sampling time T_s, resulting the following discrete-time state-space model:

$$\begin{aligned}\tilde{s}_{k+1} = &\begin{bmatrix} 1 & 0 & -\bar{v}_k \sin(\bar{\theta}_k+\bar{\psi}_k)T_s \\ 0 & 1 & \bar{v}_k \cos(\bar{\theta}_k+\bar{\psi}_k)T_s \\ 0 & 0 & 1 \end{bmatrix} \tilde{s}_k \\ &+ \begin{bmatrix} \cos(\bar{\theta}_k+\bar{\psi}_k)T_s & -\bar{v}_k \sin(\bar{\theta}_k+\bar{\psi}_k)T_s \\ \sin(\bar{\theta}_k+\bar{\psi}_k)T_s & \bar{v}_k \cos(\bar{\theta}_k+\bar{\psi}_k)T_s \\ \frac{\sin(\bar{\psi}_k)}{L}T_s & \frac{\bar{v}_k \cos(\bar{\psi}_k)}{L}T_s \end{bmatrix} \tilde{u}_k \\ \tilde{y}_k = &\ \tilde{s}_k,\end{aligned} \tag{23}$$

where subscript k denotes the value at time step k and $\widetilde{\text{Var}} = \text{Var} - \overline{\text{Var}}$. This prediction model is then used to design a linear MPC via a real-time iteration scheme [46, 47]. At each sampling time t, the following QP problem is solved to compute the MPC action to be applied:

$$\min_{\{u_{t+k|t}\}_{k=0}^{N_u-1}} \sum_{k=0}^{N_p-1} \left(\left\| y_{t+k|t} - y_{t+k}^{\text{ref}} \right\|_{Q_y}^2 + \left\| \Delta u_{t+k|t} \right\|_{Q_{\Delta u}}^2 \right) \tag{24}$$

subject to model equation (23) and the following input and output constraints

$$\begin{aligned} & y_{\min} \le y_{t+k|t} \le y_{\max}, \ \ k = 1, \dots, N_p \\ & u_{\min} \le u_{t+k|t} \le u_{\max}, \ \ k = 1, \dots, N_p \\ & \Delta u_{\min} \le \Delta u_{t+k|t} \le \Delta u_{\max}, \quad k = 1, \dots, N_p \\ & u_{t+N_u+j|t} = u_{t+N_u|t}, \quad j = 1, \dots, N_p - N_u, \end{aligned} \tag{25}$$

where $\|\cdot\|_M^2$ is the squared norm weighted by a matrix M; $\Delta u_{t+k|t} = u_{t+k|t} - u_{t+k-1|t}$; y_{ref} and u_{ref} are the reference values of control outputs and inputs during the experiment (which are unique to the LK and OA phases of the experiment); and N_p and N_u are the prediction and control horizons.

C. Control Objectives

The two main objectives involved in this control task are: (1) maintain the vehicle in the same lane with constant speed if no obstacles (other vehicles) are present; and (2) pass other moving vehicles if they are within a safety distance. The qualitative descriptions of these objectives, such as the ambiguity of transferring "optimal obstacle avoidance" into a mathematical formula, make it challenging to define a proper quantitative performance index for closed-loop performance. On the other hand, it is easier for a calibrator to compare the outcome of two different driving tests and then express a preference.

We describe the test scenario as follows. Note, for ease of assessment, the unit of v and ψ described in the following text as well as in the figures are represented in km/hr and degree (°), respectively. The controlled vehicle is initially at position (x_f, w_f) = (0, 0) m with θ = 0°. Another vehicle (obstacle) is at position (30, 0) m and moving horizontally at a constant speed of 40 km/hr. The shape of both vehicles is assumed to be rectangular, with a length of 4.5 m and a width of 1.8 m. During nominal LK conditions, the vehicle being controlled moves horizontally at 50 km/hr, with w_f = 0 m and $\dot{w}_f$ = 0 m/s . Once the obstacle is within a safety distance, the vehicle being controlled should pass it while keeping a safe lateral distance between them. In this case, the horizontal and lateral safety distances are 10 m and 3 m, respectively. The vehicle controlled by MPC can vary its velocity in the range of [40, 70] km/hr during the LK period and [50, 70] km/hr during the OA period, and its reference velocities are set to 50 and 60 km/hr, respectively. For both LK and OA periods, θ can take values in the range of [-45, 45]°, with its rate of change between each time step limited to [-5, 5]°/s.

The following MPC design parameters are tuned: sampling time (T_s); prediction and control horizons (N_p, N_u); weight matrix $Q_{\Delta u}$ in (24). The sampling time is restricted in the interval [0.085, 0.5] s. The prediction horizon is an integer allowed to vary in the range [10,30] and the control horizon is taken as a fraction ϵ_c of N_p rounded up to the closest integer, with $\epsilon_c \in$ [0.1, 1]. The weight matrix is set to be diagonal $Q_{\Delta u} = \left[\begin{smallmatrix} q_{u11} & 0 \\ 0 & q_{u22} \end{smallmatrix}\right]$ and the values of $\log(q_{u11})$ and $\log(q_{u22})$ are restricted in the interval [-5, 3]. The rest of the MPC design parameter is fixed, with $Q_y = \left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right]$.

D. Calibration Process

The first author of the paper plays the role of the calibrator for this case study. The maximum number of function evaluations N_{max} is set to 50, with $N_{init} = 10$. The default hyperparameters δ_E, δ_G and δ_S in (18) are set to 1, 1, and 0.5, respectively. The parameters σ, c_h and λ in (15) are set to 0.02, 1, and 1e-6, respectively. The hyperparameter ϵ characterizing the RBF function (12) is initialized to 1.0, and recalibrated at iterations 10, 20, 30, and 40 via 3-fold cross-validation.

The closed-loop experiment of the vehicle control is simulated for 15 seconds. Fig. 3 shows the query window for one iteration of the calibration process. The MPC design parameters and the worst-case computational time (t_{comp}) required for solving the QP problem of MPC (24) are displayed at the top of the figure. At each iteration, the calibrator is asked to decide whether the newly proposed experiment is feasible and/or satisfactory, and to express a preference between the new experiment and the current best one. More specifically, the calibrator labels the control policy that leads to "unstable/unsafe" and "unimplementable" behavior as infeasible (*i.e.*, $G(x) = 0$). Examples of "unstable/unsafe" behaviors include but are not limited to: vehicle hits the obstacle; vehicle oscillates on the road, *etc.* "Unimplementable" cases are the ones whose computational time (t_{comp}) required for solving the QP problem of MPC (24) exceeds the sampling time T_s. As for being labeled as satisfactory, the following two criteria are used: (i) the lateral position of the vehicle does not exceed 5 m during the OA period (black dashed line on Fig. 3); and (ii) the vehicle does not oscillate during the LK period (in other words, the vehicle moves at a constant speed with w_f and θ close to 0 m and 0°). These feasibility and satisfactory criteria are assumed to be unknown to C-GLISp and are learned based on the expressed feasibility and satisfactory labels.

For the pairwise comparisons, the calibrator expresses her preferences according to the following guidelines: (i) whether

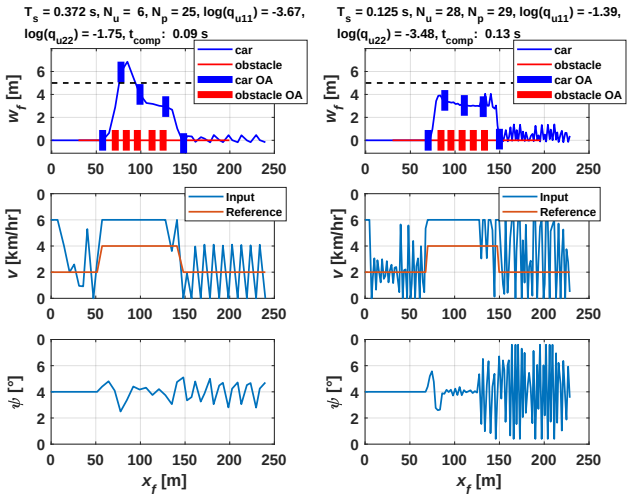

Fig. 3. Vehicle control query window. The top subplots show the vehicle and obstacle positions. The "vehicle OA" and "obstacle OA" bars show five relative positions of the vehicle and obstacle during the OA period. The dashed lines indicate the lateral distance that the car should avoid exceeding (5 m in this case). The middle subplots show the actual and reference velocity v at different longitudinal positions. The steering angle ψ over the longitudinal position is depicted in the bottom subplots. The results on the left panels are preferred and feasible, while the results on the right panels are infeasible. The results on both sets of panels fail to satisfy the satisfaction conditions.

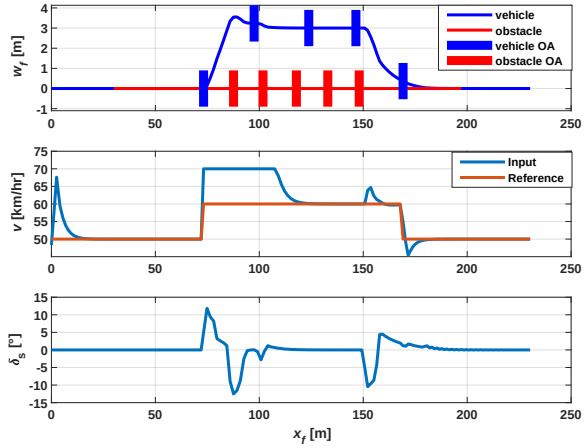

Fig. 4. Final vehicle control performance obtained by the designed MPC controller. The top subplot shows the vehicle and obstacle positions. The "vehicle OA" and "obstacle OA" bars show five relative positions of the vehicle and obstacle during the OA period. The middle subplot shows the actual and reference velocity v at different longitudinal positions. The bottom subplot shows the steering angle ψ over the longitudinal position.

it is feasible; (ii) whether it is satisfactory; (iii) whether the vehicle guarantees passengers' comfort during the OA period, for example, by not changing velocities or moving the lateral position too aggressively; (iv) whether the deviations of the vehicle velocity from the reference values is minor in both LK and OA periods; (v) whether aggressive variations of steering angles are avoided. If a conflict combination among criteria mentioned above appears, criterion (i) has the highest priority, and if the conflict is among criteria (ii)–(v), the control policy that leads to qualitatively safer driving practice based on the calibrator's experience is preferred. Note that conditions (iii)-(v) and the method of judging safe driving practice are mainly qualitative/subjective, and it is difficult to express them in terms of quantitative metrics.

For the example query window illustrated in Fig. 3, conflicting combinations of the assessing criteria are observed. Compared to the experiment shown in the left panels of the figure, the experiment shown in the right panels has more aggressive lateral movements during the OA period. Furthermore, the changes in velocity and steering angle are greater in both frequency and magnitude in both LK and OA periods. The experiment shown in the left panels is feasible since it is implementable ($t_{comp} < T_s$) and stable, while the experiment shown in the right panels of the figure is infeasible since t_{comp} exceeds T_s. Additionally, both experiments fail to satisfy the satisfaction conditions. Above all, the performance of the experiment shown in the left panels is preferred according to criterion (vi).

E. Results

C-GLISp terminates after 50 simulated closed-loop experiments and 49 pairwise comparisons. The best MPC design parameters T_s, ϵ_c, N_p, $\log(q_{u11})$ and $\log(q_{u22})$ are 0.085 s, 0.100, 23, -0.323 and -3.71, respectively, with a worst-case computation time $t_{comp} = 0.0789$ s. The closed-loop performance obtained via these MPC design parameters is depicted in Fig. 4. As shown in the figure, after only 50 simulated experiments, the proposed algorithm can tune the MPC parameters to achieve feasible and satisfactory performance, accomplishing the driving tasks with smooth and safe maneuvers.

VI. Conclusions

The algorithm C-GLISp introduced in this paper can handle preference-based global optimization with unknown objective functions and unknown constraints better than other existing black-box surrogate methods (PBO and GLISp), as illustrated through benchmark problems. The autonomous driving case study demonstrated the application of C-GLISp in semi-automated MPC calibration. Although convergence to global optimizers cannot be guaranteed, we observed that the C-GLISp can find satisfactory results within a small number of iterations and that it has a higher probability of proposing feasible samples during the exploration thanks to the introduction of additional information by the decision-maker that is used to synthesize corresponding surrogate functions.

Future research will be devoted to exploring different initial sampling methods, warm-starting procedures in the form of "transfer learning" from previous similar optimization runs, and preference-based collective learning to handle multiple decision-makers. We finally note that, while we have used C-GLISp for controller calibration, the algorithm can be used in many other applications in which a few tuning parameters must be decided based on preferences under constraints that cannot be easily quantified.

REFERENCES

[1] G. Matheron, "Principles of geostatistics," *Economic geology*, vol. 58, no. 8, pp. 1246–1266, 1963.

[2] H. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, 1964.

[3] J. Sacks, W. Welch, T. Mitchell, and H. Wynn, "Design and analysis of computer experiments," *Statistical Science*, pp. 409–423, 1989.

[4] D. Jones, M. Schonlau, and W. Matthias, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[5] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[6] G. Savaia, Y. Sohn, S. Formentin, G. Panzani, M. Corno, and S. M. Savaresi, "Experimental automatic calibration of a semi-active suspension controller via Bayesian optimization," *Control Engineering Practice*, vol. 112, 2021.

[7] G. Luo, "A review of automatic selection methods for machine learning algorithms and hyper-parameter values," *Network Modeling Analysis in Health Informatics and Bioinformatics*, vol. 5, no. 1, pp. 1–16, 2016.

[8] A. Bemporad, "Global optimization via inverse distance weighting and radial basis functions," *Computational Optimization and Applications*, vol. 77, pp. 571–595, 2020, code available at http://cse.lab.imtlucca.it/~bemporad/glis.

[9] M. Forgione, D. Piga, and A. Bemporad, "Efficient calibration of embedded MPC," in *Proc. of the 21st IFAC World Congress*, Berlin, Germany, 2020.

[10] A. Lucchini, S. Formentin, M. Corno, D. Piga, and S. M. Savaresi, "Torque vectoring for high-performance electric vehicles: an efficient MPC calibration," *IEEE Control Systems Letters*, 2020.

[11] M. Fiducioso, S. Curi, B. Schumacher, M. Gwerder, and A. Krause, "Safe contextual bayesian optimization for sustainable room temperature PID control tuning," in *Proc. of the 28th IJCAI*, Macao, China, 2019, pp. 5850–5856.

[12] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on Gaussian process global optimization," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 270–277.

[13] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven MPC design," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 577–582, 2019.

[14] S. Bansal, R. Calandra, T. Xiao, S. Levine, and C. J. Tomiin, "Goal-driven dynamics learning via Bayesian optimization," in *Proc. of the IEEE 56th Annual Conference on Decision and Control*, 2017, pp. 5168–5173.

[15] D. Drieß, P. Englert, and M. Toussaint, "Constrained Bayesian optimization of combined interaction force/task space controllers for manipulations," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 902–907.

[16] L. Roveda, M. Magni, M. Cantoni, D. Piga, and G. Bucca, "Human–robot collaboration in sensorless assembly task learning enhanced by uncertainties adaptation via Bayesian Optimization," *Robotics and Autonomous Systems*, vol. 136, p. 103711, 2021.

[17] L. Roveda, M. Forgione, and D. Piga, "Robot control parameters auto-tuning in trajectory tracking applications," *Control Engineering Practice*, vol. 101, 2020.

[18] R. Calandra, N. Gopalan, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian gait optimization for bipedal locomotion," in *International Conference on Learning and Intelligent Optimization*. Springer, 2014, pp. 274–290.

[19] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 491–496.

[20] W. Chu and Z. Ghahramani, "Extensions of Gaussian processes for ranking: semisupervised and active learning," *Learning to Rank*, vol. 29, 2005.

[21] E. Brochu, N. De Freitas, and A. Ghosh, "Active preference learning with discrete choice data." in *NIPS*, 2007, pp. 409–416.

[22] J. González, Z. Dai, A. Damianou, and N. D. Lawrence, "Preferential Bayesian optimization," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1282–1291.

[23] M. Abdolshah, A. Shilton, S. Rana, S. Gupta, and S. Venkatesh, "Multi-objective Bayesian optimisation with preferences over objectives," 2019. [Online]. Available: arXiv:1902.04228

[24] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems*, 2017, pp. 4299–4307.

[25] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz, "A survey of preference-based reinforcement learning methods," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 4945–4990, 2017.

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

[27] 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.

[28] Y. Sui, V. Zhuang, J. Burdick, and Y. Yue, "Stagewise safe Bayesian optimization with Gaussian processes," in *Proc. of the 35th ICML*. PMLR, 2018, pp. 4781–4789.

[29] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics," *Machine Learning*, pp. 1–35, 2021.

[30] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, Arlington, VA, USA, 2014, pp. 250–259.

[31] A. Candelieri, "Sequential model based optimization of partially defined functions under unknown constraints," *Journal of Global Optimization*, pp. 1–23, 2019.

[32] V. R. Joseph and L. Kang, "Regression-based inverse distance weighting with applications to computer experiments," *Technometrics*, vol. 53, no. 3, pp. 254–265, 2011.

[33] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

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

[35] 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.

[36] A. Costa and G. Nannicini, "Rbfopt: an open-source library for black-box optimization with costly function evaluations," *Mathematical Programming Computation*, vol. 10, no. 4, pp. 597–629, 2018.

[37] R. G. Regis and C. A. Shoemaker, "Constrained global optimization of expensive black box functions using radial basis functions," *Journal of Global optimization*, vol. 31, no. 1, pp. 153–171, 2005.

[38] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 36, no. 2, pp. 111–133, 1974.

[39] L. Rios and N. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013.

[40] M. McKay, R. Beckman, and W. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[41] S. K. Mishra, "Some new test functions for global optimization and performance of repulsive particle swarm method," *Available at SSRN 926132*, 2006.

[42] Phenix Integration. (2016) Bird problem (constrained). [Online]. Available: https://web.archive.org/web/20161229032528/http://www.phoenix-int.com/software/benchmark_report/bird_constrained.php

[43] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," 2013.

[44] The MathWorks, *Statistics and Machine Learning Toolbox*, Natick, Massachusetts, United State, 2019. [Online]. Available: https://www.mathworks.com/products/statistics.html

[45] A. Vaz and L. Vicente, "PSwarm: A hybrid solver for linearly constrained global derivative-free optimization," *Optimization Methods and Software*, vol. 24, pp. 669–685, 2009, http://www.norg.uminho.pt/aivaz/pswarm/.

[46] 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.

[47] 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.