SCUOLA ALTI STUDI LUCCA

Global and preference-based optimization using surrogate-based methods

Mengjia Zhu

Supervisor: Prof. Alberto Bemporad

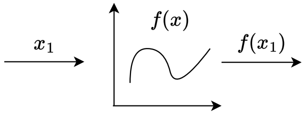
IMT School for Advanced Studies Lucca

PhD Thesis Defense, May 30, 2024

Introduction

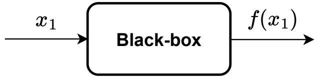
- Optimization problem:

$$\text{find } \boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x})$$



- Many decision making problems (hyperparameter tuning in machine learning, DOE in engineering design,...) require identify a global optimum w/o an **explicit** analytic expression $f(\boldsymbol{x})$
- Such problems are often referred to as black-box opt. prob., which can be solved by
  - Evolutionary algorithms
  - Direct search methods
  - Surrogate-based opt. methods

    Useful when experiment/simulation is **expensive** to evaluate,
    (want to reduce # of required evaluations/observations to locate satisfactory candidate)



Surrogate-based optimization methods - General framework

- *Objective function*: the underlying latent function, whose explicit analytic expression is unknown
- *Real experiment/simulation model*: given an input decision vector, we can observe output of obj. fun.
- *Surrogate model*: approximate the behavior of the obj. fun.
- *Acquisition function*: trade-off b/t exploitation and exploration to suggest the next point to observe

```
Initial sampling stage:
    initial dataset $\mathcal{D}$                 e.g., random sampling, LHS, and
                                                  observe obj. fun. values at sampled points
    update surrogate                              fit with the initial dataset
Repeat
    $x \leftarrow$ POLICY($\mathcal{D}$)          minimize the acq. fun.
    $y \leftarrow$ OBSERVE($x$)                   observe/sample at the chosen location, obj. fun.
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x, y)\}$    update dataset
    update the surrogate and acq. fun.            with the updated dataset
Until termination condition reached               e.g., budget exhausted
Return $\mathcal{D}$, $x^*$
```

Overview

- **C-GLISp**: Pref-based optimization with unknown constraints
  - Numerical problems
  - Case study: automated-driving for lane-keeping and obstacle-avoidance
- **PWAS**: Global and pref-based optimization with mixed variables using piecewise affine surrogates
  - Numerical problems
  - Case study: experimental planning

C-GLISp: Pref-based optimization with unknown constraints

Motivation

- Objective function may not always be *quantifiable*, *e.g.*,
  - Qualitative (subjective) feedback, not explicitly defined, difficult to evaluate
  - Multi-objective function, whose relative weights are hard to determine a prior to form a single metric
- Challenge of *unknown/non- or hard-to-quantifiable* constraints
  - In many real-world problems, not all constraints are known or can be explicitly defined at the beginning
  - Unknown constraints can hinder the implementation of certain processes at some decision variables, while traditional methods cannot directly incorporate this information

Motivation

- Objective function may not always be *quantifiable*, *e.g.*,
  - Qualitative (subjective) feedback, not explicitly defined, difficult to evaluate
  - Multi-objective function, whose relative weights are hard to determine a prior to form a single metric
- Challenge of *unknown/non- or hard-to-quantifiable* constraints
  - In many real-world problems, not all constraints are known or can be explicitly defined at the beginning
  - Unknown constraints can hinder the implementation of certain processes at some decision variables, while traditional methods cannot directly incorporate this information

To address these challenges, we propose the use of a preference-based optimization method.

Problem formulation

Target problem:

$$\begin{aligned} &\text{find } \boldsymbol{x}^\star \text{ such that } \pi(\boldsymbol{x}^\star, \boldsymbol{x}) \leq 0,\ \forall \boldsymbol{x} \in \Omega_S \cap \Omega_G \\ &\text{implying } f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x}) \end{aligned} \tag{1}$$

- $\pi(\boldsymbol{x}_1, \boldsymbol{x}_2)$: preference function
- $f(\boldsymbol{x})$: unknown latent fun. ***assumed to exist***, can be estimated with the expressed prefs
- Ω_G: unknown feasibility set
- Ω_S: unknown satisfactory set
- $\mathcal{D}$: domain (known constraints/bounds)

Unknown: not explicitly defined, *i.e.*, no known analytic expression

Main assumptions

- Obj. fun. f cannot be directly quantified, but *indirectly observed* in two ways:
  - *Preference function*: $\pi(\boldsymbol{x}_1, \boldsymbol{x}_2)$

$$\pi(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{cases} -1 & \text{if } \boldsymbol{x}_1 \text{ “better” than } \boldsymbol{x}_2 \quad [f(\boldsymbol{x}_1) < f(\boldsymbol{x}_2)] \\ 0 & \text{if } \boldsymbol{x}_1 \text{ “as good as” } \boldsymbol{x}_2 \quad [f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2)] \\ 1 & \text{if } \boldsymbol{x}_2 \text{ “better” than } \boldsymbol{x}_1 \quad [f(\boldsymbol{x}_1) > f(\boldsymbol{x}_2)]. \end{cases}$$

  - *Satisfaction function*: the set Ω_S contains all the vectors $\boldsymbol{x}$ leading to a performance that the decision-maker judges satisfactory, which, like Ω_G cannot be *explicitly* expressed

$$S(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \notin \Omega_S \\ 1 & \text{if } \boldsymbol{x} \in \Omega_S, \end{cases} \tag{2}$$

Main assumptions

- Obj. fun. f cannot be directly quantified, but *indirectly observed* in two ways:
  - *Preference function*: $\pi(\boldsymbol{x}_1, \boldsymbol{x}_2)$

$$\pi(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{cases} -1 & \text{if } \boldsymbol{x}_1 \text{ "better" than } \boldsymbol{x}_2 \quad [f(\boldsymbol{x}_1) < f(\boldsymbol{x}_2)] \\ 0 & \text{if } \boldsymbol{x}_1 \text{ "as good as" } \boldsymbol{x}_2 \quad [f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2)] \\ 1 & \text{if } \boldsymbol{x}_2 \text{ "better" than } \boldsymbol{x}_1 \quad [f(\boldsymbol{x}_1) > f(\boldsymbol{x}_2)]. \end{cases}$$

  - *Satisfaction function*: the set Ω_S contains all the vectors $\boldsymbol{x}$ leading to a performance that the decision-maker judges satisfactory, which, like Ω_G cannot be *explicitly* expressed

$$S(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \notin \Omega_S \\ 1 & \text{if } \boldsymbol{x} \in \Omega_S, \end{cases} \tag{2}$$

- The set Ω_G cannot be *explicitly* expressed, but given a vector $\boldsymbol{x} \in \mathcal{D}$, a decision-maker can assess the value of the *feasibility function* $G : \mathcal{D} \to \{0, 1\}$ defined as

$$G(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \notin \Omega_G \\ 1 & \text{if } \boldsymbol{x} \in \Omega_G \end{cases} \tag{3}$$

Why include feasibility and satisfaction function

- if using pairwise comparison information *only*: infeasible/unsatisfactory sample only indirectly reveals itself as such by losing pairwise comparisons against feasible/satisfactory ones

- If using additional feasibility and satisfaction functions: exploit the information on whether $\boldsymbol{x} \in \Omega_G$ and/or $\boldsymbol{x} \in \Omega_S$ to facilitate the optimization process

  - Explore the infeasible and/or unsatisfactory region will be explicitly penalized and therefore reduce the number of samples $\boldsymbol{x}_i \notin \Omega_G$ and/or $\boldsymbol{x}_i \notin \Omega_S$

Proposed solution strategy - Overview of C-GLISp

- **Step 1**: Learning unknown *feasibility/satisfaction* constraints functions
- **Step 2**: Learning unknown *preference* function
- **Step 3**: Minimizing the *acquisition function* to identify the next sample to observe
- Iterate steps 1-3 until some terminal conditions are met

Proposed solution strategy - Step 1

Step 1: Learning unknown feasibility/satisfaction constraints functions with Inverse Distance Weighting (IDW) interpolation function (A. Bemporad, 2020)

- Decision-maker: provide for samples $i = 1\ldots, N$
  - *feasibility vector* $G_F = [G_1 \; \ldots \; G_N]' \in \{0,1\}^N$ with

$$G_i = G(\boldsymbol{x}_i) \tag{4}$$

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

$$S_i = S(\boldsymbol{x}_i) \tag{5}$$

Proposed solution strategy - Step 1

- Surrogate fitting:
  - Predict the probability of satisfying the feasibility constraint $\boldsymbol{x} \in \Omega_G$ with $\hat{G} : \mathcal{D} \to \mathbb{R}$

$$\hat{G}(\boldsymbol{x}) = \sum_{i=1}^{N} \nu_i(\boldsymbol{x})\, G_i, \tag{6}$$

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

$$\nu_i(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \\ 0 & \text{if } \boldsymbol{x} = \boldsymbol{x}_j, j \neq i \\ \frac{w_i(\boldsymbol{x})}{\sum_{i=1}^{N} w_i(\boldsymbol{x})} & \text{otherwise} \end{cases} \tag{7}$$

Here $w_i : \mathcal{D} \setminus \{\boldsymbol{x}_i\} \to \mathbb{R}$ is the following IDW function (VR. Joseph & L. Kang, 2011): $w_i(x) = \frac{e^{-d^2(x,x_i)}}{d^2(x,x_i)}$ where $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ denotes the squared Euclidean distance: $d(\boldsymbol{x}, \boldsymbol{x}_i) = \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2$

  - Predict the probability of satisfying the satisfaction constraint $\boldsymbol{x} \in \Omega_S$ with $\hat{S} : \mathcal{D} \to \mathbb{R}$, defined similarly as $\hat{G}$

Proposed solution strategy - Step 2

Step 2: Learning the preference function

- <u>Decision-maker</u>: provide
  - *preference vector* $B = [b_1 \ \ \dots \ \ b_M]^T \in \{-1, 0, 1\}^M$ with

$$b_h = \pi(\boldsymbol{x}_{i(h)}, \boldsymbol{x}_{j(h)}) \tag{8}$$

for $\boldsymbol{x}_i, \ \boldsymbol{x}_j \in \mathcal{D}$ such that $\boldsymbol{x}_i \neq \boldsymbol{x}_j, \forall i \neq j, \ i, j = 1, \dots, N$

M: number of expressed preferences, $1 \leq M \leq \binom{N}{2}$

$h \in \{1, \dots, M\}$: index enumerating the preferences

$i(h), j(h) \in \{1, \dots, N\}, \ i(h) \neq j(h)$

Proposed solution strategy - Step 2

(A. Bemporad & D. Piga, 2021)

- Surrogate model:
  - Parameterize the surrogate function $\hat{f}: \mathcal{D} \to \mathbb{R}$ as a linear combination of radial basis functions (RBFs)

$$\hat{f}(\boldsymbol{x}) = \sum_{k=1}^{N} \beta_k \phi(\epsilon d(\boldsymbol{x}, \boldsymbol{x}_i)), \quad (9)$$

$\phi : \mathbb{R} \to \mathbb{R}$ is an RBF

$\beta = [\beta_1 \ \ldots \ \beta_N]^T$: unknown coefficients to be determined (to be discussed in the following slides)

$\epsilon > 0$: scalar param. defining the shape of the RBF (periodically updated via K- fold cross-validation)

Proposed solution strategy - Step 2

(A. Bemporad & D. Piga, 2021)

- Integrate expressed preferences with the surrogate:
  - Use the expressed preferences b_h to shape $\hat{f}$ by imposing the following constraints:

$$\begin{aligned} \hat{f}(\boldsymbol{x}_{i(h)}) &\le \hat{f}(\boldsymbol{x}_{j(h)}) - \sigma && \text{if } b_h = -1 \\ \hat{f}(\boldsymbol{x}_{i(h)}) &\ge \hat{f}(\boldsymbol{x}_{j(h)}) + \sigma && \text{if } b_h = 1 \\ |\hat{f}(\boldsymbol{x}_{i(h)}) - \hat{f}(\boldsymbol{x}_{j(h)})| &\le \sigma && \text{if } b_h = 0 \end{aligned} \tag{10}$$

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

  - ***Goal*** : fit $\hat{f}$ so that it satisfies the implicit rankings of the underlying latent function

Proposed solution strategy - Step 2

- Surrogate fitting: (A. Bemporad & D. Piga, 2021)
  - Coefficients $\boldsymbol{\beta}$ describing $\hat{f}$ is obtained by solving the following convex QP problem

$$
\begin{aligned}
\min_{\beta,\varepsilon_h} \quad & \sum_{h=1}^{M} c_h\varepsilon_h + \frac{\lambda}{2}\sum_{k=1}^{N}\beta_k^2 \\
\text{s.t.} \quad & \sum_{k=1}^{N}\beta_k(\phi(\epsilon d(\boldsymbol{x}_{i(h)},\boldsymbol{x}_k)) - \phi(\epsilon d(\boldsymbol{x}_{j(h)},\boldsymbol{x}_k))) \le -\sigma+\varepsilon_h, \quad \forall h: \ b_h=-1 \\
& \sum_{k=1}^{N}\beta_k(\phi(\epsilon d(\boldsymbol{x}_{i(h)},\boldsymbol{x}_k)) - \phi(\epsilon d(\boldsymbol{x}_{j(h)},\boldsymbol{x}_k))) \ge \sigma-\varepsilon_h, \quad \forall h: \ b_h=1 \\
& \left|\sum_{k=1}^{N}\beta_k(\phi(\epsilon d(\boldsymbol{x}_{i(h)},\boldsymbol{x}_k)) - \phi(\epsilon d(\boldsymbol{x}_{j(h)},\boldsymbol{x}_k)))\right| \le \sigma+\varepsilon_h, \quad \forall h: \ b_h=0 \\
& h=1,\dots,M
\end{aligned}
\tag{11}
$$

c_h: positive weights; ε_h: positive slack variables to relax the constraints imposed by (10)

Proposed solution strategy - Step 2

- Surrogate fitting: (A. Bemporad & D. Piga, 2021)
  - Coefficients $\boldsymbol{\beta}$ describing $\hat{f}$ is obtained by solving the following convex QP problem

$$
\begin{aligned}
\min_{\beta,\varepsilon_h} \quad & \sum_{h=1}^{M} c_h \varepsilon_h + \frac{\lambda}{2} \sum_{k=1}^{N} \beta_k^2 \\
\text{s.t.} \quad & \sum_{k=1}^{N} \beta_k(\phi(\epsilon d(\boldsymbol{x}_{i(h)}, \boldsymbol{x}_k)) - \phi(\epsilon d(\boldsymbol{x}_{j(h)}, \boldsymbol{x}_k))) \le -\sigma + \varepsilon_h, \quad \forall h: \; b_h = -1 \\
& \sum_{k=1}^{N} \beta_k(\phi(\epsilon d(\boldsymbol{x}_{i(h)}, \boldsymbol{x}_k)) - \phi(\epsilon d(\boldsymbol{x}_{j(h)}, \boldsymbol{x}_k))) \ge \sigma - \varepsilon_h, \quad \forall h: \; b_h = 1 \\
& \left| \sum_{k=1}^{N} \beta_k(\phi(\epsilon d(\boldsymbol{x}_{i(h)}, \boldsymbol{x}_k)) - \phi(\epsilon d(\boldsymbol{x}_{j(h)}, \boldsymbol{x}_k))) \right| \le \sigma + \varepsilon_h, \quad \forall h: \; b_h = 0 \\
& h = 1, \dots, M
\end{aligned}
\tag{11}
$$

c_h: positive weights; ε_h: positive slack variables to relax the constraints imposed by (10)

  - The expressed preferences: soft constraints (maximize constraint satisfaction while still allowing inconsistent expressed rankings and/or imperfect surrogate model to some extent)

Proposed solution strategy - Step 3

Step 3: Construction and minimization of the acquisition function

- Acquisition function: minimized to identify the next sample $\boldsymbol{x}_{N+1}$ to observe

$$\boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x} \in \mathcal{D}} a(\boldsymbol{x}) \tag{12}$$

Proposed solution strategy - Step 3

Step 3: Construction and minimization of the acquisition function

- Acquisition function: minimized to identify the next sample $\boldsymbol{x}_{N+1}$ to observe

$$\boldsymbol{x}_{N+1} = \arg \min_{\boldsymbol{x} \in \mathcal{D}} a(\boldsymbol{x}) \tag{12}$$

- How to construct an acquisition function based on the information gathered?
  - Minimizing $\hat{f}$ greedily to generate the next sample $\boldsymbol{x}_{N+1}$ may lead the solver to converge to a point that is not the global optimum
  - **Ideal**: balance *exploitation* of $\hat{f}$, and *safe exploration* within unexplored regions

Proposed solution strategy - Step 3

- Acquisition function: $a : \mathcal{D} \to \mathbb{R}$ is defined as

$$a(\boldsymbol{x}) = \underbrace{\frac{\hat{f}(\boldsymbol{x})}{\Delta \hat{F}}}_{\text{surrogate of pref. fun.}} - \underbrace{\delta_E z_N(\boldsymbol{x})}_{\text{pure exploration}} + \underbrace{\delta_G(1 - \hat{G}(\boldsymbol{x}))}_{\text{penalize infeasible samples}} + \underbrace{\delta_S(1 - \hat{S}(\boldsymbol{x}))}_{\text{penalize unsatisfactory samples}} \tag{13}$$

  - Surrogate of pref. fun.: *exploitation* of the leaned latent fun.
    $\Delta \hat{F} = \max_i\{\hat{f}(\boldsymbol{x}_i)\} - \min_i\{\hat{f}(\boldsymbol{x}_i)\}$: range of $\hat{f}$ on the samples in $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
  - Pure exploration: search regions with limited/no samples to reduce the uncertainty associated with $\hat{f}$
  - Penalization terms: *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 $\boldsymbol{x} \notin \Omega_S$

Proposed solution strategy - Step 3

Pure exploration term $z_N(\boldsymbol{x})$:

- Modified the IDW exploration function in (A. Bemporad & D. Piga, 2021) with the following guidelines:
  - Encourage 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
  - Empirically observed to better escape from local minima

$$z_N(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \\ \left(1 - \frac{N}{N_{max}}\right) \tan^{-1}\left(\frac{\sum_{i=1}^{N} r_i(\boldsymbol{x}_N^*)}{\sum_{i=1}^{N} r_i(\boldsymbol{x})}\right) + \frac{N}{N_{max}} \tan^{-1}\left(\frac{1}{\sum_{i=1}^{N} r_i(\boldsymbol{x})}\right) & \text{otherwise} \end{cases} \tag{14}$$

where $r_i(x) = \frac{1}{d^2(\boldsymbol{x}, \boldsymbol{x}_i)}$; x_N^*: best decision variable found up to iteration N

N_{max}: max. allowed number of exp., common stopping criterion

arc tangent fun.: prevent new sampled pt from getting excessively far away from the existing ones

Proposed solution strategy - Step 3

Penalization terms :

- Use ample stand. dev. to update, after each iteration, the weights δ_G and δ_S as follows:

$$\begin{aligned} \delta_G &= (1-\hat{\sigma}_G)\delta_{G,\text{default}} \\ \delta_S &= (1-\hat{\sigma}_S)\delta_{S,\text{default}} \end{aligned} \tag{15}$$

where

$$\begin{aligned} \hat{\sigma}_G &= \min\left\{1,\ \sqrt{\frac{\sum_{i=1}^{N}(\hat{G}(\boldsymbol{x_i}) - G(\boldsymbol{x_i}))^2}{N-1}}\right\} \\ \hat{\sigma}_S &= \min\left\{1,\ \sqrt{\frac{\sum_{i=1}^{N}(\hat{S}(\boldsymbol{x_i}) - S(\boldsymbol{x_i}))^2}{N-1}}\right\} \end{aligned} \tag{16}$$

$\delta_{G,\text{default}}$ and $\delta_{S,\text{default}}$ are default values set by the user
$\delta_{G,\text{default}} > \delta_{S,\text{default}}$ (infeasibility is penalized more than unsatisfactory behavior)

Proposed solution strategy - Step 3

- Acquisition function: minimized to identify the next sample $\boldsymbol{x}_{N+1}$ to observe

$$\boldsymbol{x}_{N+1} = \arg\min_{\boldsymbol{x}\in\mathcal{D}} a(\boldsymbol{x})$$

$$a(\boldsymbol{x}) = \frac{\hat{f}(\boldsymbol{x})}{\Delta\hat{F}} - \delta_E z_N(\boldsymbol{x}) + \delta_G(1 - \hat{G}(\boldsymbol{x})) + \delta_S(1 - \hat{S}(\boldsymbol{x}))$$

- Different opt. methods can be used to solve the minimization problem
  - *e.g.*, Particle Swarm Optimization. (PSO), evolutionary algorithms, ...

Numerical problem

"camelsixhumps-hard and soft constrained" (CHSC): include both feasibility and satisfaction constraints

- Objective function:

$$f(x,y) = (4 - 2.1x^2 + x^{4/3})x^2 + xy + (4y^2 - 4)y^2$$

- Feasibility constraints:

$$g_2(x,y) = x^2 + (y + 0.04)^2 < 0.8$$

- Satisfaction constraints:

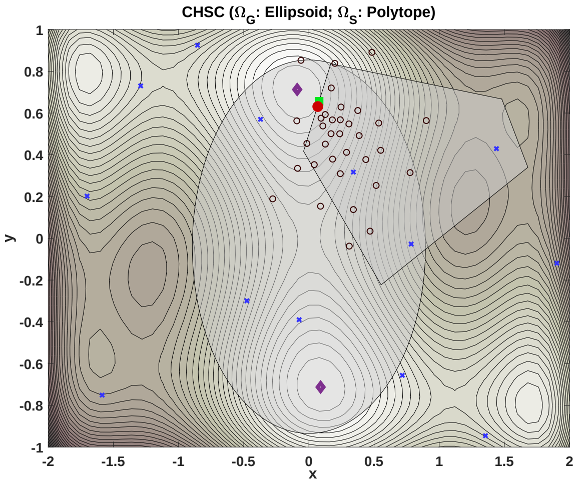
$$g_1(x,y) : \begin{bmatrix} 1.6295 & 1 \\ 0.5 & 3.875 \\ -4.3023 & -4 \\ -2 & 1 \\ 0.5 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} < \begin{bmatrix} 3.0786 \\ 3.324 \\ -1.4909 \\ 0.5 \\ 0.5 \end{bmatrix}$$

- Search domain $\mathcal{D}$: $[-2,2] \times [-1,1]$

- The two unconstrained optima are both feasible but NOT satisfactory

Numerical problem

“camelsixhumps-hard and soft constrained” (CHSC): include both feasibility and satisfaction constraints



- Blue $\times$: pts generated from initial sampling phase
- Black $\circ$: pts generated from active learning phase
- Purple $\Diamond$: global unconstrained optimizer
- Red $\bullet$: constrained optimizer found after N_{max} iter
- Green $\square$: global constrained optimizer

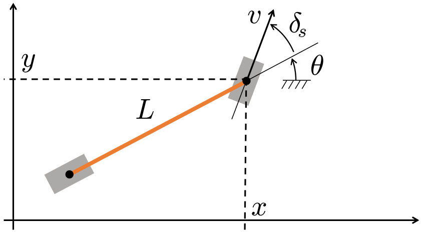
Observations

- As N increases, the pts generated by C-GLISp approach the constrained optimizer
- Most of the pts generated during the active learning lay in the feasibility and satisfaction regions

Case study - Overview

- **General description**: automated driving with lane-keeping (LK) and obstacle-avoidance (OA)
- **System description**: simplified 2-DOF kinematic bicycle model with the front wheel as the reference point

$$\begin{aligned}\dot{x}_f &= v\cos(\theta+\delta_s)\\ \dot{y}_f &= v\sin(\theta+\delta_s)\\ \dot{\theta} &= \frac{v\sin(\delta_s)}{L}\end{aligned}$$

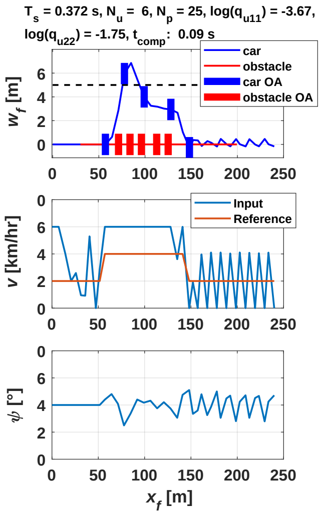


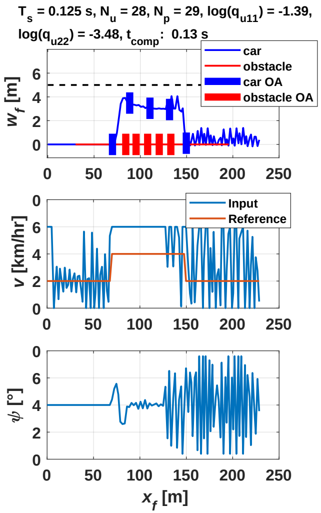
- **Control strategy**: linear time-varying (LTV) MPC, with constant sampling time
- **Goal**: Tune the MPC parameters (sampling time, prediction horizon, control horizon, and weight matrix of manipulated variables) using expressed preferences and feasibility and satisfaction indications by the calibrator

Case study - Guideline of the calibrator

1. Whether it is feasible
2. Whether it is satisfactory
3. Whether the vehicle guarantees passengers' comfort during the OA period, for example, by not changing velocities or moving the lateral position too aggressively
4. whether the deviations of the velocity from the ref. val. is minor in both LK and OA periods
5. whether aggressive variations of steering angles are avoided
6. If a conflict combination among criteria mentioned above appears, criterion (1) has the highest priority, and if the conflict is among criteria (2)–(5), the control policy that leads to qualitatively safer driving practice based on the calibrator's experience is preferred

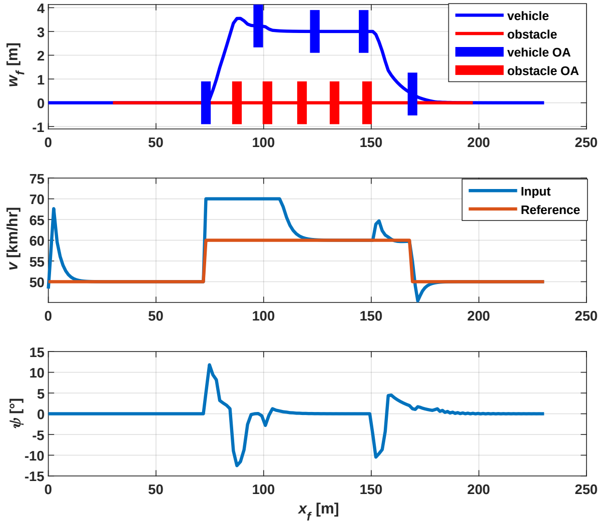
Case study - Example query window





- At each iter., the calibrator is asked to express
  1) pref. between the newly observed exp. and the current best exp.
  2) feasibility assess. for the new exp.
  3) satisfaction assess. for the new exp.
- Left panels: preferred, feasible, unsatisfactory
- Right panels: infeasible ($t_{\text{comp}} > T_s$), unsatisfactory
- Note: infeasible exp. is NOT implementable (simulated exp. is used in the case study)

Case study - Results



- After 50 simulated closed-loop exp. and 49 pairwise comp.
- Best MPC design parameters
  T_s = 0.085 s
  ϵ_c = 0.100
  N_p = 23
  $\log(q_{u11})$ = -0.323
  $\log(q_{u22})$ = -3.71
  t_{comp} = 0.0789 s
  (worst-case computational time)
- C-GLISp can find satisfactory results within a small number of iter.

PWAS: Global and pref-based optimization with mixed variables using piecewise affine surrogates

Motivation

- The variables $\boldsymbol{x}$ can be defined over a **mixed-variable** domain
  - Continuous, integer (ordinal), categorical (non-ordinal)
- Many engineering prob. (*e.g* exp. design) often include **constraints** of mixed-integer nature
- Existing surrogate-based methods often address constraints by penalization which can still suggest infeasible samples to observe, especially when mixed-integer linear constraints are present

Motivation

- The variables $\boldsymbol{x}$ can be defined over a **mixed-variable** domain
  - Continuous, integer (ordinal), categorical (non-ordinal)
- Many engineering prob. (*e.g* exp. design) often include **constraints** of mixed-integer nature
- Existing surrogate-based methods often address constraints by penalization which can still suggest infeasible samples to observe, especially when mixed-integer linear constraints are present

Goal: Solve medium-sized mixed-variable nonlinear opt. problems s.t mixed-integer linear equality and/or inequality constraints

Problem formulation

Optimization problems with **mixed variables** and **linear constraints**

$$
\begin{aligned}
\text{find } \boldsymbol{X}^* \in \quad & \arg\min_{\boldsymbol{X}\in\Omega} f(\boldsymbol{X}) && \text{(17a)}\\
\text{s.t.} \quad & A_{\text{eq}}\boldsymbol{x} + B_{\text{eq}}\boldsymbol{y} + C_{\text{eq}}\boldsymbol{z} = \boldsymbol{b_{\text{eq}}} && \text{(17b)}\\
& A_{\text{ineq}}\boldsymbol{x} + B_{\text{ineq}}\boldsymbol{y} + C_{\text{ineq}}\boldsymbol{z} \le \boldsymbol{b_{\text{ineq}}} && \text{(17c)}
\end{aligned}
$$

- $\boldsymbol{X} = [\boldsymbol{x};\ \boldsymbol{y};\ \boldsymbol{z}]$
  - continuous $\boldsymbol{x}$, integer $\boldsymbol{y}$, one-hot encoded categorical $\boldsymbol{z}$
  - $\boldsymbol{x} \in \mathbb{R}^{n_c}$, $\boldsymbol{y} \in \mathbb{Z}^{n_{\text{int}}}$, $\boldsymbol{\ell_x} \le \boldsymbol{x} \le \boldsymbol{u_x}$, $\boldsymbol{\ell_y} \le \boldsymbol{y} \le \boldsymbol{u_y}$
  - $Z = [Z^1, \dots, Z^{n_d}]$, with n_i classes in each categorical variable Z^i, $i = 1, \dots, n_d$.
    - Z^i is one-hot binary encoded into the subvector $[z_{1+d^{i-1}} \ \dots \ z_{d^i}]^\mathsf{T} \in \{0,1\}^{n_i}$ $\forall i = 1, \dots, n_d$, where $d^0 = 0,\ d^i = \sum_{j=1}^{i} n_j$,
    - $\boldsymbol{z} \in \{0,1\}^{d^{n_d}}$: complete vector of binary variables after the encoding, with $\boldsymbol{z} \in \Omega_z = \{\boldsymbol{z} \in \{0,1\}^{d^{n_d}} : \sum_{j=1}^{n_i} z_{j+d^{i-1}} = 1,\ \forall i = 1, \dots, n_d\}$.
- Domain $\Omega = [\boldsymbol{\ell_x}, \boldsymbol{u_x}] \times ([\boldsymbol{\ell_y}, \boldsymbol{u_y}] \cap \mathbb{Z}) \times \Omega_z$

Proposed solution strategy

PWAS: Global optimization with mixed variables using Piecewise Affine Surrogates

- Why PWA:
  - Allow discontinuities introduced by sharp transitions induced by taking values in different classes of the categorical variables
  - PWA surrogates have a direct mixed-integer linear reformulation and, therefore, can be minimized by efficient MILP solvers (*e.g.*, Gurobi and GLPK)

Overview of PWAS

- Change of variables: scaling and encoding
- Surrogate fitting: Piecewise affine function
- Exploration function: distance-based and frequency-based
- Acquisition function

PWAS - change of variables

- **Continuous variables**: scale
- **Categorical variables**: one-hot encoded
- **Integer variables**: treat as categorical or continuous
  - Depend on the number of possible combinations

PWAS - updated problem formulation

New opt. vector $\bar{\boldsymbol{X}} = [\bar{\boldsymbol{x}};\ \bar{\boldsymbol{y}};\ \boldsymbol{z}] \in \bar{\Omega}$, Problem (17) translates to

- **Scenario one** (treat integer variables as **categorical**)

$$\begin{aligned} \text{find } \bar{\boldsymbol{X}}^* \in\ & \arg\min_{\bar{\boldsymbol{X}}\in\bar{\Omega}} f(S(\bar{\boldsymbol{X}})) \\ \text{s.t. } & \bar{A}_{\text{eq}}\bar{\boldsymbol{x}} + \bar{B}_{\text{eq}}\bar{\boldsymbol{y}} + C_{\text{eq}}\boldsymbol{z} = \bar{\boldsymbol{b}}_{\mathbf{eq}} \\ & \bar{A}_{\text{ineq}}\bar{\boldsymbol{x}} + \bar{B}_{\text{ineq}}\bar{\boldsymbol{y}} + C_{\text{ineq}}\boldsymbol{z} \le \bar{\boldsymbol{b}}_{\mathbf{ineq}}. \end{aligned} \tag{18a}$$

- **Scenario two** (treat integer variables as **continuous**)

$$\begin{aligned} \text{find } \begin{bmatrix} \bar{\boldsymbol{X}}^* \\ \boldsymbol{y}^* \end{bmatrix} \in\ & \arg\min_{\bar{X}\in\bar{\Omega},\, y\in[\ell_y, u_y]\cap\mathbb{Z}} f(S(\bar{\boldsymbol{X}})) \\ \text{s.t. } & \bar{A}_{\text{eq}}\bar{\boldsymbol{x}} + B_{\text{eq}}\boldsymbol{y} + C_{\text{eq}}\boldsymbol{z} = \bar{\boldsymbol{b}}_{\mathbf{eq}} \\ & \bar{A}_{\text{ineq}}\bar{\boldsymbol{x}} + B_{\text{ineq}}\boldsymbol{y} + C_{\text{ineq}}\boldsymbol{z} \le \bar{\boldsymbol{b}}_{\mathbf{ineq}} \end{aligned} \tag{18b}$$

Note:

- Denote $D \subseteq \bar{\Omega}$ the set of admissible vectors $\bar{\boldsymbol{X}}$ satisfying the constraints
- Evaluating the obj. fun. requires the original values in $\boldsymbol{X}$, denote $S : \bar{\Omega} \mapsto \Omega$ the inverse scaling/encoding mapping of $\bar{\boldsymbol{X}}$, *i.e.*, $\boldsymbol{X} = S(\bar{\boldsymbol{X}})$

PWAS - surrogate fitting

Consider N samples $\bar{\boldsymbol{X}}_{\boldsymbol{1}}, \ldots, \bar{\boldsymbol{X}}_{\boldsymbol{N}} \in \mathbb{R}^n$ and their corresponding function evaluations $f(\bar{\boldsymbol{X}}_{\boldsymbol{1}}), \ldots, f(\bar{\boldsymbol{X}}_{\boldsymbol{N}}) \in \mathbb{R}$.

Goal: Define the PWA surrogate function $\hat{f}$ over a polyhedral partition of $\bar{\Omega}$ into K regions.

PWA separation function $\phi : \mathbb{R}^n \mapsto \mathbb{R}$

$$\phi(\bar{\boldsymbol{X}}) = \boldsymbol{\omega}_{j(\bar{\boldsymbol{X}})}^\top \bar{\boldsymbol{X}} + \gamma_{j(\bar{\boldsymbol{X}})} \tag{19a}$$

where $\boldsymbol{\omega}_j \in \mathbb{R}^n$ and $\gamma_j \in \mathbb{R}$, $j = 1, \ldots, K$, need to be determined, and

$$j(\bar{\boldsymbol{X}}) = \arg \max_{j=1,\ldots,K} \{\boldsymbol{\omega}_j^\top \bar{\boldsymbol{X}} + \gamma_j\}, \tag{19b}$$

PWA surrogate function $\hat{f}$ as

$$\hat{f}(\bar{\boldsymbol{X}}) = \boldsymbol{a}_{j(\bar{\boldsymbol{X}})}^\top \bar{\boldsymbol{X}} + b_{j(\bar{\boldsymbol{X}})} \tag{19c}$$

where $\boldsymbol{a}_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$, $j = 1, \ldots, K$, also need to be determined.

Note: $\hat{f}$ is possibly non-convex and discontinuous.

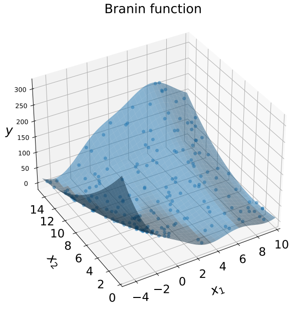
PWAS - surrogate fitting

Piecewise affine function with PARC

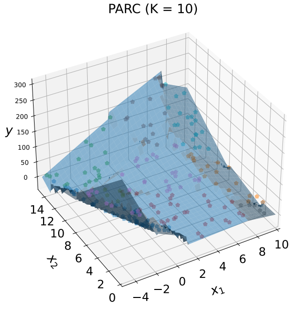
(Bemporad, 2023)

Example: Branin function (800 training samples, 10 PWA partitions):

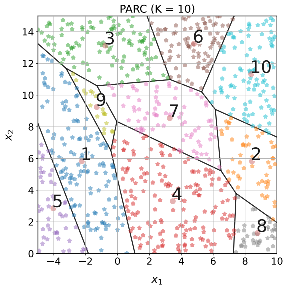
$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s$$
$$a = 1,\ b = \frac{5.1}{4\pi^2},\ c = \frac{5}{\pi},\ r = 6,\ s = 10,\ t = \frac{1}{8\pi}$$
$$-5 \le x_1 \le 10,\ 0 \le x_1 \le 15.$$



(a)



(b)



Remark:
Our purpose is to obtain a highly accurate approx. of the obj. fun. **around the global optimal solution** and not necessarily over the entire domain of $\bar{X}$, which usually requires **much fewer** samples

PWAS - max-box exploration method

Goal:

- Define a fun. $E_{ct} : \mathbb{R}^{n_{ct}} \mapsto \mathbb{R}$ mapping a generic **numeric vector** $\bar{\boldsymbol{x}} \in \mathbb{R}^{n_{ct}}$ into a nonnegative value

PWAS - max-box exploration method

Goal:

- Define a fun. $E_{ct} : \mathbb{R}^{n_{ct}} \mapsto \mathbb{R}$ mapping a generic **numeric vector** $\bar{\boldsymbol{x}} \in \mathbb{R}^{n_{ct}}$ into a nonnegative value

Desired features:

- Zero at given samples $\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_N$
- Grows away from the given samples
- Can be solved by MILP solvers

PWAS - max-box exploration method

Goal:

- Define a fun. $E_{ct} : \mathbb{R}^{n_{ct}} \mapsto \mathbb{R}$ mapping a generic **numeric vector** $\bar{\boldsymbol{x}} \in \mathbb{R}^{n_{ct}}$ into a nonnegative value

Desired features:

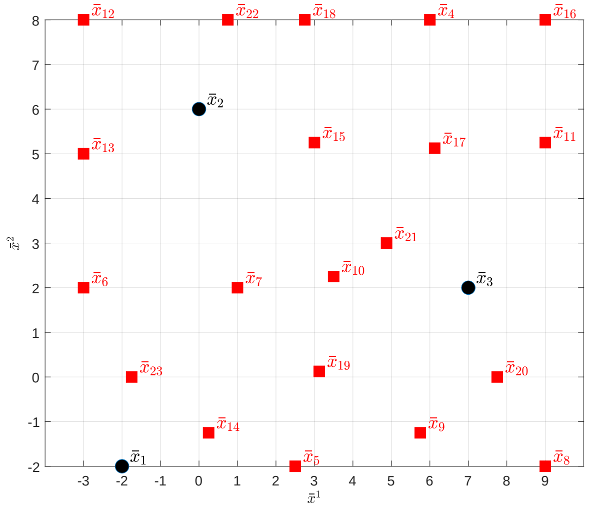
- Zero at given samples $\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_N$
- Grows away from the given samples
- Can be solved by MILP solvers

Max-box exploration method:

- $B_i(\beta_{ct}) = \{\bar{\boldsymbol{x}} : \ \|\bar{\boldsymbol{x}} - \bar{\boldsymbol{x}}_i\|_\infty \leq \beta_{ct}\}$
- $E_{ct}(\bar{\boldsymbol{x}}) = \min\{\beta_{ct} \geq 0 : \ \bar{\boldsymbol{x}} \in B_i(\beta_{ct})$ for some $i = 1, \dots, N\}$
- Maximizing $E_{ct}(\bar{\boldsymbol{x}})$ is equivalent to finding the largest value β_{ct} and a vector $\bar{\boldsymbol{x}}^*$ outside the interior of all boxes $B_i(\beta_{ct})$

PWAS - max-box exploration method

Example: $\bar{\boldsymbol{x}} \in \mathbb{R}^2$, $D = [-3, 9] \times [-2, 8]$, 3 existing samples $\bar{x}_1, \bar{x}_2, \bar{x}_3$, run for 20 sequential iterations



PWAS - Hamming distance method

Goal:

- Define a fun. $E_{dt} : \{0,1\}^d \mapsto \mathbb{R}$ mapping a **binary vector** $\boldsymbol{z} \in \{0,1\}^d$ into a nonnegative value

PWAS - Hamming distance method

Goal:

- Define a fun. $E_{dt} : \{0, 1\}^d \mapsto \mathbb{R}$ mapping a **binary vector** $\boldsymbol{z} \in \{0, 1\}^d$ into a nonnegative value

Desired features:

- Account frequency of occurrence of a particular combination of binary variables
- Select the ones occur less frequently
- Can be solved by MILP solvers

PWAS - Hamming distance method

Goal:

- Define a fun. $E_{dt} : \{0,1\}^d \mapsto \mathbb{R}$ mapping a **binary vector** $\boldsymbol{z} \in \{0,1\}^d$ into a nonnegative value

Desired features:

- Account frequency of occurrence of a particular combination of binary variables
- Select the ones occur less frequently
- Can be solved by MILP solvers

Hamming distance exploration method:

- Given two binary vectors $\boldsymbol{z} = [z^1, \dots, z^d]^\mathsf{T} \in \{0,1\}^d$ and $\boldsymbol{z}_i = [z_i^1 \;\; \dots \;\; z_i^d]^\mathsf{T} \in \{0,1\}^d$
- Hamming distance: $d_H(\boldsymbol{z}, \boldsymbol{z}_i) = \sum_{m=1}^{d} |z^m - z_i^m|$
- Linear encoding: $d_H(\boldsymbol{z}, \boldsymbol{z}_i) = \sum_{m: z_i^m = 0} z^m + \sum_{m: z_i^m = 1} (1 - z^m)$
- $E_{dt}(\boldsymbol{z})$ quantifies the average number of different binary components between $\boldsymbol{z}$ and the given N vectors $\boldsymbol{z}_1, \dots, \boldsymbol{z}_N$: $E_{dt}(\boldsymbol{z}) = \frac{1}{dN} \sum_{i=1}^{N} d_H(\boldsymbol{z}, \boldsymbol{z}_i)$
- Maximizing $E_{dt}(\boldsymbol{z})$ to find $\boldsymbol{z}^*$

PWAS - Hamming distance method

Example: three categorical variables $Z = [Z^1, Z^2, Z^3]$

- $Z^1 \in \{A, B\}$, $Z^2 \in \{A, B, C, D, E\}$, and $Z^3 \in \{A, B, C\}$
- Three initial samples: $Z_1 = [A, E, C]$, $Z_2 = [B, B, B]$, $Z_3 = [A, D, C]$
- Binary encode and get the corresponding vectors $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, $\boldsymbol{z}_3 \in \{0, 1\}^{10}$
- Solve $\boldsymbol{z}^* \in \arg\max_{\boldsymbol{z} \in D} \; E_{dt}(\boldsymbol{z})$ to identify $\boldsymbol{z}_4 = \boldsymbol{z}^*$ and its decoded form Z_4.

Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Z^1	B	A	B	A	B	A	B	A	B	A
Z^2	A	C	D	A	E	B	C	D	A	E
Z^3	A	B	A	C	A	B	C	A	B	C
Iteration	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Z^1	B	A	B	A	B	A	B	A	B	A
Z^2	B	C	D	A	E	B	C	D	A	E
Z^3	A	B	C	A	B	C	A	B	C	A

PWAS - Acquisition function

$$\text{find } \bar{\boldsymbol{X}}_{N+1} \in \arg\min_{\bar{\boldsymbol{X}} \in D} \frac{\hat{f}(\bar{\boldsymbol{X}})}{\Delta F} - \delta_1 E_{ct}(\bar{\boldsymbol{x}}) - \delta_2 E_{ct}(\bar{\boldsymbol{y}}) - \delta_3 E_{dt}(\boldsymbol{z}) \tag{20a}$$

$$\text{find } \begin{bmatrix} \bar{\boldsymbol{X}}_{N+1} \\ \boldsymbol{y}_{N+1} \end{bmatrix} \in \arg\min_{\bar{\boldsymbol{X}} \in D, y \in [\ell_y, u_y] \cap \mathbb{Z}} \frac{\hat{f}(\bar{\boldsymbol{X}})}{\Delta F} - \delta_1 E_{ct}(\bar{\boldsymbol{x}}) - \delta_2 E_{dt}(\bar{\boldsymbol{y}}) - \delta_3 E_{dt}(\boldsymbol{z}) \tag{20b}$$

where

$$\Delta F = \max\left\{ \max_{i=1,\ldots,N} f(\boldsymbol{X}_i) - \min_{i=1,\ldots,N} f(\boldsymbol{X}_i), \epsilon_{\Delta F} \right\}$$

- $\epsilon_{\Delta F} > 0$: a threshold to prevent division by zero
- ΔF: scaling factor, eases the selection of the exploration parameters δ_1, δ_2, and δ_3

Numercial problem - linearly constrained

Problem description: $n_c = 3$, $n_{\text{int}} = 4$, and $n_d = 2$ with $n_1 = 3$ and $n_2 = 2$

$$f(X) = \begin{cases} |f_1| & n_{d2} = 0 \\ f_1 & n_{d2} = 1 \end{cases}$$

$$\text{s.t} \quad A_{\text{ineq}}x + B_{\text{ineq}}y \le b_{\text{ineq}}$$

$$\text{where} \quad f_1(x, y) = \begin{cases} f_{\text{Horst6}}(x) + f_{\text{hs044}}(y) & n_{d1} = 0 \\ 0.5 f_{\text{Horst6}}(x) + f_{\text{hs044}}(y) & n_{d1} = 1 \\ f_{\text{Horst6}}(x) + 2 f_{\text{hs044}}(y) & n_{d1} = 2 \end{cases}$$

$$f_{\text{Horst6}}(x) = x^T Q x + p x$$

$$Q = \begin{bmatrix} 0.992934 & -0.640117 & 0.337286 \\ -0.640117 & -0.814622 & 0.960807 \\ 0.337286 & 0.960807 & 0.500874 \end{bmatrix}$$

$$p = \begin{bmatrix} -0.992372 & -0.046466 & 0.891766 \end{bmatrix}$$

$$f_{\text{hs044}}(y) = x_0 - x_1 - x_2 - x_0 x_2 + x_0 x_3 + x_1 x_2 - x_1 x_3$$

$$A_{\text{ineq}} = \begin{bmatrix} 0.488509 & 0.063565 & 0.945686 \\ -0.578592 & -0.324014 & -0.501754 \\ -0.719203 & 0.099562 & 0.445225 \\ -0.346896 & 0.637939 & -0.257623 \\ -0.202821 & 0.647361 & 0.920135 \\ -0.983091 & -0.886420 & -0.802444 \\ -0.305441 & -0.180123 & -0.515399 \end{bmatrix}, \quad B_{\text{ineq}} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$b_{\text{ineq}} = [2.86506,\ -1.49161,\ 0.51959,\ 1.58409,\ 2.19804,\ -1.30185,\ -0.73829,\ 8,\ 12,\ 12,\ 8,\ 8,\ 5]^{\mathsf{T}}$$

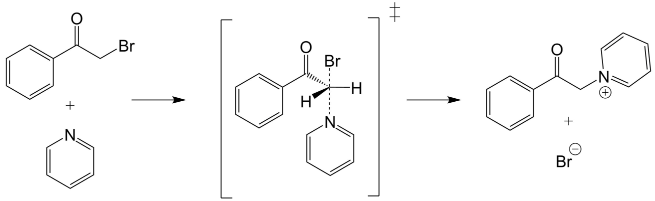
PWAS implementation

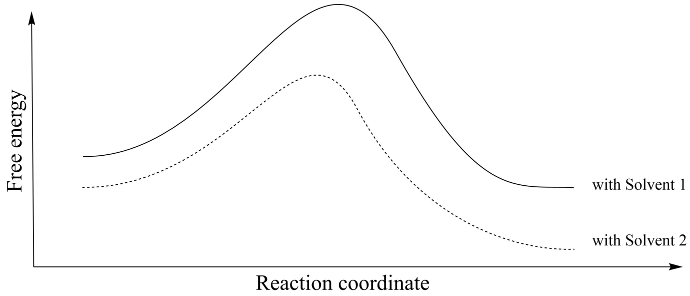
- Formulated with the PuLP library
- Solved by Gurobi's MILP solver to find the next query point

Global optimum

- Analytical: -62.579
- PWAS: -62.579 $\pm$ 3.5275e-08 (over 20 random repetitions)

Case study - Menschutkin reaction





Goal: Select a solvent to maximize the reaction rate k

A good solvent can help lower the liquid phase activation Gibbs

Problem description

Optimization variables - 54 in total

(Gui et al, 2023)

- 46 integer variables, indicating the number of each functional group presented in the designed solvent
- 1 categorical variable (m), auxiliary variable, which equals to 1 for an acyclic molecule, 0 for a monocyclic molecule and -1 for a bicyclic molecule
- 7 binary variables for structure-related constraints
- integer variables are upper bounded to reduce the complexity of the designed solvent
- one-hot binary encode the categorical and binary variables

Inequality constraints - 116 in total

- structure-property related
- chemical feasibility and complexity related

Equality constraints - 5 in total

- structure-property related
- chemical feasibility and complexity related

Results

PWAS

Rank	Chemical formula	ln k	
		QM	pred
1	CH_3NHCHO	-5.92	-5.92
2	$OHCH_2NO_2$	-6.46	-6.49
3	$CH_2OHCH_2NO_2$	-6.72	-6.69
4	$(CH_3)_2SO$	-6.82	-6.82
5	$(CH_2)_2OHCH_2NO_2$	-6.93	-6.93
6	$CH_3CHOHCH_2NO_2$	-6.96	-6.97
7	$CH_2{=}COHCH_2NO_2$	-6.98	-6.97
8	$CH{=}CHOHCH_2NO_2$	-7.00	-6.98
9	$(CH_2)_3OHCH_2NO_2$	-7.10	-7.10
10	$CHCH_2{=}CHOHCH_2NO_2$	-7.11	-7.11

QM-DoE-CAMD

Rank	Chemical formula	ln k	
		QM	pred
1	$CH_2OHCH_2NO_2$	-6.72	-5.50
2	$(CH_3)_2SO$	-6.82	-5.59
3	$CH_2OHCH_2NO_2$	-6.72	-6.28
4	$CH_2{=}COHCH_2NO_2$	-6.98	-6.66
5	$(CH_2)_2OHCH_2NO_2$	-6.93	-6.74
6	$CH_3CHOHCH_2NO_2$	-6.96	-6.87
7	$(CH_3)_2COHCH_2NO_2$	-7.23	-6.91
8	$CH{=}CHOHCH_2NO_2$	-7.00	-6.92
9	$CH_2CH_2{=}COHCH_2NO_2$	-7.15	-6.97
10	CH_3NHCHO	-5.92	-7.00

- The top 10 solvents obtained using PWAS are consistent with the top 10 solvents ranked based on QM calculation within the whole feasible domain
- The pred. val. based on PWAS are closer to the QM calculated values compared to the ones calculated from QM-DoE-CAMD (MLR) method

Conclusion and discussions

Conclusion and discussions

- Discussed two methods
  - C-GLISp: address problems with non-quantifiable obj. fun. and unknown constraints
  - PWAS: address problems with mixed-variables with linear equality/inequality constraints
- Both methods can achieve satisfactory performance within small number of experiments for their target problems
- Future research directions
  - Preference-based opt. methods: transition from conventional "human-in-the-loop" opt. to "AI/LLM-in-the-loop" opt.
  - Mixed-variable opt. methods: investigate the adoption of acquisition strategies in PWAS to other BO methods, especially the ones with tree-based kernels

Thank you!

- Supervisor: Prof. Alberto Bemporad
- Thesis review
  - Prof. Fabio Schoen, University of Florence
  - Prof. Benoît Chachuat, Imperial College London
  - Dr. Mario Eduardo Villanueva, IMT Lucca
- Collaborators:
  - Pref-based MPC calibration, C-GLISp: Prof. Dario Piga
  - DENSO corner-case detection: Dr. Hasan Esen, Dr. Maximilian Kneissl, and Dr. Adam Molin, Dr. Dejan Ničković, and Dr. Edgar A. Aguilar
  - Experimental planning with PWAS: Dr. Ehecatl Antonio del Río Chanona, Dr. Ye Seol Lee, Prof. Kim Jelfs, Dr. Austin Mroz and Dr. Lingfeng Gui
  - Distributed GLIS: Prof. Dario Piga, Dr. Loris Cannelli and Dr. Francesco Farina

Complementary slides

C-GLISp

Pref-based optimization for multi-objective function

Comment from Prof. Fabio Schoen:

> *Did you consider multi-objective Bayesian optimization, which might be used to build a part of the Pareto front, leaving the request for preference a posteriori?*

We did not consider multi-objective BO explicitly, but there are several works in literature that explored this area. For example (R. Ozaki, 2024), (M. Abdolshah, 2019) , (A. Ahmadianshalchi, 2024). In general, similar to GLISp/C-GLISp, preferences from the decision-maker are used as constraints to structure the utility function for acquisition.

In the original GLISp paper (A. Bemporad & D. Piga, 2020), GLISp is used to learn a preferred Pareto optimal solution for MOO

Preliminary: pref-based optimization

- **Goal**: leverage the knowledge of preferences provided by the user to guide the optimization process
- **Key features**:
  - Often interactive and involve an iterative process
  - Rely on **preference elicitation techniques**: direct assessment, pairwise comparisons, interactive visualization
  - **Preference modeling**: learning utility functions, learning preference relations, function approximation

Focus on pref-based opt. methods that learn a surrogate model (***utility function***) respecting the prefs expressed by the decision-maker (***constraints***) based on ***pairwise comparisons***

Preliminary: active pref-based optimization

Assumptions: (A. Bemporad and D. Piga, 2021)

- Simulation/Real experiment: expensive to evaluate
- Optimization vector $\boldsymbol{x}$: bounded, s.t known constraints
- Closed-loop performance objective function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ is not available,(**latent function**)
- We can only express a **preference** between two choices, implying the relation between their fun. eval. of the latent fun.:

$$\pi(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{cases} -1 & \text{if } \boldsymbol{x}_1 \text{ ``better'' than } \boldsymbol{x}_2 & [f(\boldsymbol{x}_1) < f(\boldsymbol{x}_2)] \\ 0 & \text{if } \boldsymbol{x}_1 \text{ ``as good as'' } \boldsymbol{x}_2 & [f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2)] \\ 1 & \text{if } \boldsymbol{x}_2 \text{ ``better'' than } \boldsymbol{x}_1 & [f(\boldsymbol{x}_1) > f(\boldsymbol{x}_2)]. \end{cases}$$

The following **transitivity** properties hold for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$:

$$\pi(\boldsymbol{x}_1, \boldsymbol{x}_1) = 0, \quad \pi(\boldsymbol{x}_1, \boldsymbol{x}_2) = -\pi(\boldsymbol{x}_2, \boldsymbol{x}_1)$$
$$\pi(\boldsymbol{x}_1, \boldsymbol{x}_2) = \pi(\boldsymbol{x}_2, \boldsymbol{x}_3) = -1 \Rightarrow \pi(\boldsymbol{x}_1, \boldsymbol{x}_3) = -1$$

Goal: Find a global optimum $\boldsymbol{x}^*$ (= "not worse" than any other $\boldsymbol{x}$)

Find $\boldsymbol{x}^\star$ such that $\pi(\boldsymbol{x}^\star, \boldsymbol{x}) \leq 0, \ \forall \boldsymbol{x} \in \mathcal{D}$, implying $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$

Preference modeling

Discussed extensively in (C. Wirth, et al., 2017)

- Learning utility functions: evaluate individual alternatives
  - Estimate a utility score for each option or item, which reflects the user's preference.
  - Assume that user preferences can be represented by a numerical value, where higher values indicate stronger preferences
- Learning preference relations: compare pairs of competing alternatives
  - Model the relative preference between pairs of items/options instead of absolute utility values
- Function approximation
  - Use machine learning algorithms to approximate the function that maps features to preference scores
  - A generalization of learning utility functions
  - Features are not necessarily defined, *e.g.*, deep learning can automatically learn representations from raw data (e.g., images, text) and can integrate various data types into a single model

Feasibility and satisfaction function in C-GLISp

Comment from Prof. Fabio Schoen:

> *Why this concept (satisfaction function) should not lead to a constraint ... Why these two concepts are treated separately?*

You are correct, the satisfaction function leads to constraints. Here, it is mainly used to facilitate the problem-solving process by providing additional information to guide the search. Since the relative importance is different (penalized different), they are treated separately to make most use of each experiment.

When the decision-maker judges for the satisfaction function, unknown constraints related to implementation details are ignored, *i.e.*, solely judged based on the final performance. This is relevant for simulation case studies.

Consider two scenarios:

- Simulation case study: 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.
- Physical experiments: Ω_S is necessarily a subset of Ω_G, as no performance would be available for evaluation when the parameters are infeasible

Feasibility and satisfaction function in C-GLISp - cont.

It is also useful to provide implicit rankings to help refine the solution, for example:
current best: feasible, satisfactory
sample one: feasible, unsatisfactory
sample two: feasible, satisfactory
When comparing the current best with samples one and two, assume the current best is still preferred to both samples one and two. If we don't have information on the satisfaction function, we don't know the implicit ranking b/t samples one and two unless we make another comparison among them explicitly.

Nevertheless, each new sample will provide one more data point to train the surrogate of the satisfaction function to promote sampling within the regions that can lead to a satisfactory performance with a higher probability.

Proposed solution strategy - Step 1

Why use IDW interpolation functions?

- Other binary classification methods
  - Logistic regression or random forests: accuracy with a small number of training data is limited
  - Support vector machines (SVMs): empirical tests revealed that IDW outperform SVM in this case
- The functions $\hat{G}$ and $\hat{S}$ generated by IDW interpolation are always between 0 and 1 by construction (Lemma 1-P2 (A. Bemporad, 2020))
  - Can be interpreted as probabilities of being feasible/satisfactory

Proof for $\hat{G} \in [0,1]$

Adopted from the proof for Lemma 1-P2 of (A. Bemporad, 2020)

For all $\boldsymbol{x} \in \mathbb{R}^{n_x}$, the value $v_i(\boldsymbol{x}) \in [0,1], \forall i = 1, \dots, N$, and $\sum_{i=1}^{N} v_i(\boldsymbol{x}) = 1$, so that

$$\min_j \{G_j\} = \sum_{i=1}^{N} v_i(\boldsymbol{x}) \min_j \{G_j\} \leq \sum_{i=1}^{N} v_i(\boldsymbol{x}) G_i = \hat{G} \leq \sum_{i=1}^{N} v_i(\boldsymbol{x}) \sum_{i=1}^{N} v_i(\boldsymbol{x}) \max_j \{G_j\} = \max_j \{G_j\}$$

since $G \in \{0,1\}$, $\min_j \{G_j\} = 0$ and $\max_j \{G_j\} = 1$: $\hat{G} \in [0,1]$

Common RBFs

RBF: measure similarity based on distance d

- *Gaussian*: $\phi(\epsilon d) = e^{-(\epsilon d)^2}$
  - Most common
- *Multiquadric RBF Kernel*: $\sqrt{(1+\epsilon d)^2}$
  - Less sensitive to the scale of the data compared to the Gaussian RBF
- *Inverse multiquadric* : $\phi(\epsilon d) = \frac{1}{\sqrt{1+(\epsilon d)^2}}$
  - Inverse of the multiquadric kernel, can handle data with large variations in feature values
- *Inverse quadratic*: $\phi(\epsilon d) = \frac{1}{1+(\epsilon d)^2}$
  - Similar to the inverse multiquadric, but without the square root, it can provide a steeper decay
  - Default of C-GLISp
- *Thin plate spline*: $\phi(\epsilon d) = (\epsilon d)^2 \log(\epsilon d)$
  - Useful for interpolation problems, derived from the theory of thin plate splines

Hyper-parameter selection for C-GLISp

- Case-dependent, but in general, it mainly depends on $N_{\max}$ (resource limitation)
- In general, set $\delta_{G,\text{default}} = \delta_E$, with $\delta_{S,\text{default}} = 0.5\ \delta_{G,\text{default}}$
- Solver default: $\delta_{G,\text{default}} = \delta_E = 1$, $\delta_{S,\text{default}} = 0.5$, with $N_{\max} = 50$
- When $N_{\max}$ increases, suggest to increase δ_E to promote exploration,
  - *e.g.*, if double $N_{\max}$, double δ_E
  - Harder problem (prob. with multiple local optimizers), if larger $N_{\max}$ is allowed, increase δ_E
- Another option: adaptively update δ_E similar to $\delta_{G,\text{default}}$ and $\delta_{S,\text{default}}$, based on constraint satisfaction (*i.e.*, the value of the slack variable ε_h in (11))

General procedures of surrogate-based opt. methods

1. Define the *objective function* to optimize (*e.g.*, the performance of a simulation or a real-world problem). It needs to provide the output based on an input decision vector, but it does not necessarily require an explicit analytic expression.

2. Select a *surrogate model* to approximate the behavior of the objective function (*e.g.*, Gaussian processes, radial basis functions, *etc.*.). One may select the model based on the prior knowledge available and the characteristics of the problems (deterministic/stochastic).

3. Select an *exploration model* (*e.g.*, space-filling methods, *i.e.*, disperse points to promote a broad coverage across the domain, probability of improvement, *etc.*.). An appropriate exploration model encourages exploration in unvisited feasible regions, aiming to decrease uncertainties in the surrogate model and avoid getting stuck in local optima.

4. Generate an initial set of samples using some *initial sampling strategies* (*e.g.*, Latin hypercube sampling). For sample efficiency, it is important to ensure diversity and coverage across a wide range of input space.

5. Evaluate the initial samples.

General procedures of surrogate-based opt. methods - cont.

6. Build the surrogate model by training it with the initial samples and their corresponding function evaluations.

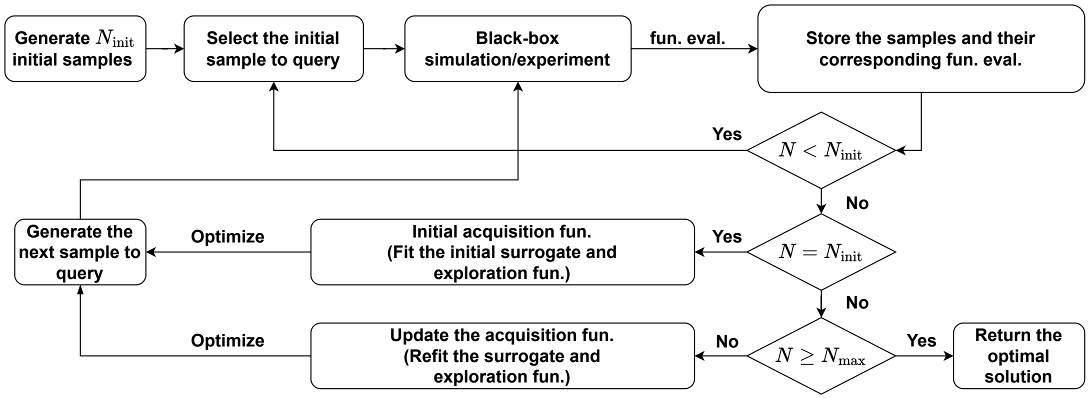
7. Select the next sample to test by optimizing the *acquisition function*, which trades off between the exploitation of the objective function predictions based on the surrogate model and the exploration of the input space (prediction uncertainty/improvement/diversity) based on the exploration model.

8. Evaluate the objective function with the newly queried sample from step 7.

9. Update the surrogate model by incorporating the new sample and its function evaluation. The predictability of the surrogate model is improved by iteratively updating the surrogate model throughout the optimization procedure,

10. Repeat steps 7 to 9 until a stopping criterion is met (*e.g.*, a maximum number of function evaluations, a converge threshold, *etc.*).

Surrogate-based optimization methods - General framework



Generate N_{init} initial samples
Select the initial sample to query
Black-box simulation/experiment
fun. eval.
Store the samples and their corresponding fun. eval.
Yes
$N < N_{\text{init}}$
No
Yes
$N = N_{\text{init}}$
No
Generate the next sample to query
Optimize
Initial acquisition fun. (Fit the initial surrogate and exploration fun.)
Optimize
Update the acquisition fun. (Refit the surrogate and exploration fun.)
No
$N \geq N_{\max}$
Yes
Return the optimal solution

Automated driving case study details

MPC controller

- Let us describe the model in general nonlinear multi-input multi-output form

$$\begin{aligned}\dot{x} &= f(x, u)\\ y &= g(x, u),\end{aligned}$$

- **Linear time-varying (LTV)** MPC strategy, with constant sampling time T_s:

$$\begin{aligned}\tilde{x}_{k+1} &= A_k\tilde{x}_k + B_k\tilde{u}_k\\ \tilde{y}_k &= C_k\tilde{x}_k + D_k\tilde{u}_k,\end{aligned}$$

- At each sample t, compute the MPC action $u_{t|t}$ by solving a **quadratic program (QP)**

$$\min_{\{u_{t+k|t}\}_{k=0}^{N_u-1},\varepsilon} \sum_{k=0}^{N_p-1} \left\|y_{t+k|t} - y_{t+k}^{\mathrm{ref}}\right\|_{Q_y}^2 + \sum_{k=0}^{N_p-1} \left\|u_{t+k|t} - u_{t+k}^{\mathrm{ref}}\right\|_{Q_u}^2 + \sum_{k=0}^{N_p-1} \left\|\Delta u_{t+k|t}\right\|_{Q_{\Delta u}}^2$$

Automated driving case study details - cont.

Discrete-time state-space model for the case study:

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

$$\tilde{y}_k = \tilde{s}_k,$$

- The subscript $_k$ denotes the value at time step k
- Nominal trajectory: $\bar{s}_k = [\bar{x}_{f_k}\ \bar{w}_{f_k}\ \bar{\theta}_k]'$, $\bar{u}_k = [\bar{v}_k\ \bar{\psi}_k]'$, and $\bar{y}_k = \bar{s}_k$
- $\widetilde{\text{Var}} = \text{Var} - \overline{\text{Var}}$ denotes the deviation from the nominal value

Automated driving case study details - cont.

Real-time iteration (RTI) scheme (S. Gros, et al., 2020), (M. Diehl, et al., 2005):

- RTI exploits the fact that successive optimal control problems in NMPC are closely related, allowing for iterative improvement of solutions ”on-the-fly”
- The RTI approach linearizes the system dynamics at the current state and control prediction, rather than the reference trajectory, and uses a numerical integration scheme for simulation
- This method achieves fast convergence by employing Newton-type optimization techniques, ensuring the NMPC solution updates in real time with system dynamics
- RTI-based NMPC can be viewed as a special case of LTV MPC, where linearization occurs online

PWAS

One-hot encoding

Categorical variable example:

Class	One-hot encoding
A	001
B	010
C	100

Integer variable example:

Integer	One-hot encoding
3	0001
4	0010
5	0100
6	1000

for integer in [3,6]

PWAS - change of variables

Continuous variables:

- Rescale every continuous variable x_i into a new variable $\bar{x}_i \in [-1, 1]$ such that

$$x_i = \frac{u_x^i - \ell_x^i}{2}\bar{x}_i + \frac{u_x^i + \ell_x^i}{2}, \ \forall i = 1, \dots, n_c.$$

- Update the corresponding constraint matrices and right-hand-side vectors

- Possibly further tightened the intervals $[-1, 1]$ by taking the updated inequality constraints (if exist) into account

PWAS - change of variables

Integer variables (assume only a finite number $N_{\max}$ of queries can be made):

- Treat as **categorical** ($\prod_{i=1}^{n_{\text{int}}} n_i^{\text{int}} < N_{\max}$)
  - One-hot encode integer variables into further $d^{n_{\text{int}}}$ binary variables $\bar{y}_j \in \{0,1\}$, $j = 1, \dots, d^{n_{\text{int}}}$, where $d^{n_{\text{int}}} = \sum_{i=1}^{n_{\text{int}}} n_i^{\text{int}}$
  - Update the relevant constraint matrices and right-hand-side vectors
- Treat as **continuous** ($\prod_{i=1}^{n_{\text{int}}} n_i^{\text{int}} \geq N_{\max}$)
  - Similar to continuous variables, rescale integer variable y_i into a new variable $\bar{y}_i \in [-1, 1]$ such that

$$y_i = \frac{u_y^i - \ell_y^i}{2}\bar{y}_i + \frac{u_y^i + \ell_y^i}{2}.$$

  - Update the corresponding constraint matrices and right-hand-side vectors
  - Possibly further tightened the intervals $[-1, 1]$ by taking the updated inequality constraints (if exist) into account

Note: $n_i^{\text{int}} = \lfloor u_y^i \rfloor - \lceil \ell_y^i \rceil + 1$ is the cardinality of the set $[l_y^i, u_y^i] \cap \mathbb{Z}$, , *i.e.*, the number of integer values that variable y_i can take.

Choice of K and initial sample size

Comment from Prof. Fabio Schoen:

Comment on how the choice of K and the initial sample size influence the overall performance in realistic test cases

- For K, since we used *PARC*, where K is adaptively updated. It is less critical to assign a very precision initial K. It is often a good practice to assign $K = n_X + 1$. Also, K needs to be smaller than N_{init}
- N_{init} depends on the number of opt. var. and max. number of samples allowed