Global and Preference-Based Optimization with Mixed Variables Using Piecewise Affine Surrogates

Mengjia Zhu[1] · Alberto Bemporad[1]

Abstract

Optimization problems involving mixed variables (i.e., variables of numerical and categorical nature) can be challenging to solve, especially in the presence of mixed-variable constraints. Moreover, when the objective function is the result of a complicated simulation or experiment, it may be expensive-to-evaluate. This paper proposes a novel surrogate-based global optimization algorithm to solve linearly constrained mixed-variable problems up to medium size (around 100 variables after encoding). The proposed approach is based on constructing a piecewise affine surrogate of the objective function over feasible samples. We assume the objective function is black-box and expensive-to-evaluate, while the linear constraints are quantifiable, unrelaxable, a priori known, and are cheap to evaluate. We introduce two types of exploration functions to efficiently search the feasible domain via mixed-integer linear programming solvers. We also provide a preference-based version of the algorithm designed for situations where only pairwise comparisons between samples can be acquired, while the underlying objective function to minimize remains unquantified. The two algorithms are evaluated on several unconstrained and constrained mixed-variable benchmark problems. The results show that, within a small number of required experiments/simulations, the proposed algorithms can often achieve better or comparable results than other existing methods.

 Springer

1 Introduction

A large variety of decision problems in several application domains, such as model selection in machine learning [37], engineering design [30], and protein design [59], require identifying a global optimum without an explicit closed-form expression correlating the optimization variables to form the objective function to optimize. Such a black-box input/output function can be expensive-to-evaluate, as they may represent the outcomes (i.e., function outputs) of costly experiments or computationally intensive simulations resulting from a given value of the decision variables (i.e., function inputs). Therefore, the number of function evaluations should be reduced as much as possible. Also, these functions often operate over mixed-variable domains, meaning the variables can be of different types, such as continuous, integer, or categorical, adding complexity to the optimization process. Additionally, physical problems frequently include constraints of a mixed-integer nature (e.g., constraints formed with logical conditions, which often involve continuous and binary variables). In some cases, evaluating infeasible instances of the optimization variables may be impossible—for instance, when the corresponding function evaluation requires running a simulation or an experiment that can not be conducted or poses safety risks. As a result, it is preferable to efficiently exploit the known admissible set of the problem to encourage feasible sampling.

Surrogate-based optimization techniques have been studied extensively to target black-box optimization problems with expensive-to-evaluate objective functions [3, 29]. For example, Bayesian Optimization (BO) [13, 42] has been widely used in hyperparameter tuning in machine learning [58] and adaptive experimental design [22]. One of the authors recently developed a deterministic algorithm, GLIS [8], which has been applied to controller tuning [18], demonstrating comparable performance to BO but with lower computational cost. Although most of the literature has focused only on real-valued optimization variables, a few approaches have been adopted to handle integer and categorical variables [2]. Here, we distinguish integer variables as those representing ordinal relationships and categorical variables as those representing non-ordinal relationships. Integer variables are most commonly considered as continuous variables during the solution process and rounded to the nearest integer during post-analysis (e.g., MISO [47], RBFopt [15]). On the other hand, categorical variables are often one-hot encoded and treated as continuous variables within the range [0, 1] when constructing the surrogate model during optimization. After the optimization step, these variables are rounded and decoded back to their original categorical form for testing (e.g., One-hot BO [21], MINOAN [30]). See also [21, 24, 40] for algorithms that have applied similar approaches to handle integer and categorical variables. Ploskas and Sahinidis [50] comprehensively analyzed and compared different algorithms and their associated software packages, targeting bound-constrained mixed-integer derivative-free optimization problems. In their review, the authors observed that MISO [47] demonstrates superior performance when dealing with large (51–500 variables) and binary problems. On the other hand, NOMAD [4, 5] emerged as the top performer for mixed-integer, discrete (non-binary), small, and medium-sized (up to 50 variables) problems.

Most of the surrogate-based methods assume all the inputs as continuous and ordinal [30, 47, 50]. On the other hand, different classes for the categorical variables often represent different choices rather than ordinal relations. Therefore, if one attempts to fit the latent function using a unified surrogate, in which categorical variables are one-hot encoded and treated as continuous vectors with entries in [0, 1], sharp transitions might be observed in the constructed surrogate, leading to poor fitting qualities. Alternatively, one can fit different surrogate models to each categorical class [20, 48, 54]. For example, EXP3BO [20] constructs a surrogate model with Gaussian Process (GP) for each chosen class of the categorical variable. However, as the number of categories and classes within each category increase, the size of the problem quickly blows up. To alleviate this issue, Ru et al. [51] propose an approach that makes efficient use of the information in the acquired data by combining the strengths of multi-armed bandits and BO with GP models. This method, named as CoCaBO, has been shown to effectively solve bound-constrained problems with multiple categorical variables and multiple possible choices. Certain algorithms can inherently handle categorical variables due to the nature of their models. For example, the tree-structured models used in TPE [11] and the random forests employed in SMAC [27] naturally process categorical data.

In addition to mixed variables, real-life optimization problems frequently contain constraints. In this case, if the integer and the one-hot encoded categorical variables are relaxed as continuous variables while optimizing (i.e., the integrality of the variables is neglected) the constraints may not be satisfied after post-analysis, especially when equality constraints are present [30]. In [30], to maintain the integrality of the variables, the authors use one-hot encoding to convert integer and one-hot encoded categorical variables to auxiliary variables. However, infeasibility with respect to constraints is still allowed during the solution process in [30]. In [49], piecewise-linear neural networks are employed as surrogate models to address constrained discrete black-box optimization problems, where mixed-integer linear programming (MILP) is used to optimize the acquisition function. However, the no-good constraints used in [49] to tackle discrete-variable-only problems cannot be trivially transferred to the mixed-variable domain; hence, this approach cannot be directly applied to domains with mixed variables.

1.1 Contribution

In this work, we aim to solve medium sized mixed-variable nonlinear optimization problems (up to 100 variables after encoding) subject to mixed-integer linear equality and/or inequality constraints (up to 100 constraints). Specifically, the optimization variables can be continuous, integer, and categorical, and the constraints are quantifiable unrelaxable a priori known (QUAK) based on the taxonomy in [33]. We propose an algorithm that uses piecewise affine (PWA) functions as the surrogate models. PWA functions can effectively handle the discontinuities introduced by sharp transitions among different classes of categorical variables. Furthermore, they can be directly reformulated into MILPs, allowing us to leverage efficient, off-the-shelf MILP solvers to optimize the acquisition function. To balance the exploitation and exploration during

acquisition, we incorporate two types of exploration functions—distance-based and frequency-based—in the acquisition function. Additionally, we propose incorporating the exploration function as part of the initial sampling strategy to obtain well-scattered initial samples, especially when a large number of linear equality and/or inequality constraints are present. This is crucial because the initial samples play an essential role in surrogate fitting, particularly when the function-evaluation budget is limited.

We name the proposed algorithm as **PWAS**, short for **P**iece**w**ise **A**ffine **S**urrogate-based optimization. We show the efficiency and effectiveness of PWAS by comparing its performance with other existing solvers on a set of benchmark problems. We also present an extension of PWAS to address problems where function evaluations are unavailable, such as those involving multiple objectives with unclear relative weights to form a single objective function or cases where only qualitative assessments are available. This approach assumes that a decision-maker can express *preferences* between two candidate solution vectors. Such preference information is used to shape the PWA surrogate through the proposed algorithm, named **PWASp** (short for **PWAS** based on **p**references). Python implementations of PWAS and PWASp are available on the GitHub repository (https://GitHub.com/mjzhu-p/PWAS).

The rest of the paper is organized as follows. The description of the target problem is formulated in Sect. 2. The proposed surrogate-based optimization algorithms are discussed in Sects. 3 and 4. Section 5 reports the numerical benchmarks demonstrating the effectiveness of the proposed method. Lastly, conclusions and directions for future research are discussed in Sect. 6.

2 Problem Formulation

We consider a decision problem with n_c real variables grouped in vector $x \in \mathbb{R}^{n_c}$, n_{int} integer variables grouped in vector $y \in \mathbb{Z}^{n_{\text{int}}}$, and n_d categorical variables grouped in list $Z = [Z^1, \ldots, Z^{n_d}]$. Each categorical variable Z^i can take values within its corresponding n_i classes, for $i = 1, \ldots, n_d$. Let us assume that each categorical variable Z^i is one-hot binary encoded into the subvector $[z_{1+d^{i-1}} \ldots z_{d^i}]^\mathsf{T} \in \{0, 1\}^{n_i}$ for each $i = 1, \ldots, n_d$, where $d^0 = 0$ and $d^i = \sum_{j=1}^{i} n_j$, with $z \in \{0, 1\}^{d^{n_d}}$ being the complete vector of binary variables after the encoding. Here, $z \in \Omega_z = \{z \in \{0, 1\}^{d^{n_d}} : \sum_{j=1}^{n_i} z_{j+d^{i-1}} = 1, \ \forall i = 1, \ldots, n_d\}$. Let $X = [x^\mathsf{T} \ y^\mathsf{T} \ z^\mathsf{T}]^\mathsf{T}$ denote the overall optimization vector. We assume that the vectors x and y of interest are bounded (i.e., $\ell_x \leq x \leq u_x$ and $\ell_y \leq y \leq u_y$) and denote the domain of X by $\Omega = [\ell_x, u_x] \times ([\ell_y, u_y] \cap \mathbb{Z}) \times \Omega_z$. We denote $f : \Omega \mapsto \mathbb{R}$ as the objective function to minimize, assuming it is noiseless and expensive-to-evaluate.

The black-box mixed-variable optimization problem we want to solve can be stated as follows, where we aim to

$$\text{find} \ \ X^* \in \ \arg\min_{X \in \Omega} f(X) \tag{1a}$$

$$\text{s.t.} \quad A_{\text{eq}}x + B_{\text{eq}}y + C_{\text{eq}}z = b_{\text{eq}} \tag{1b}$$

$$A_{\text{ineq}}x + B_{\text{ineq}}y + C_{\text{ineq}}z \leq b_{\text{ineq}}, \tag{1c}$$
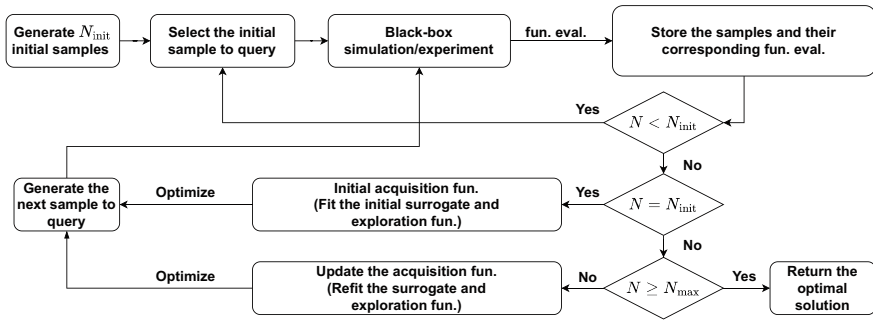
Fig. 1 General procedures for surrogate-based optimization methods

where A_{eq}, B_{eq}, C_{eq}, A_{ineq}, B_{ineq}, and C_{ineq} are matrices of suitable dimensions that, together with the right-hand-side column vectors b_{eq} and b_{ineq}, define possible linear equality and inequality constraints on x, y, and z. For example, if $x \in \mathbb{R}$ and $Z = [Z^1]$ with $Z^1 \in \{\mathsf{red}, \mathsf{blue}, \mathsf{yellow}\}$, the logical constraint $[Z^1 = \mathsf{red}] \rightarrow [x \le 0]$ can be modeled as $x \le u_x(1 - z_1)$. For modeling more general types of mixed linear/logical constraints (possibly involving the addition of auxiliary real and binary variables), the reader is referred to, e.g., [25, 56, 57]. Note that, different from function f, which is assumed to be black-box and expensive-to-evaluate, we assume the mixed-integer linear constraints on X in (1) are QUAK [33] and are cheap to evaluate.

3 Solution Method

We follow the general surrogate-based optimization procedure, schematically depicted in Fig. 1 (see, e.g., [8]), to solve the optimization problem (1). The approach consists of an initial (passive) sampling and an active learning stage, in which a surrogate model of the objective function f is repeatedly learned. During the initial phase, N_{init} feasible samples X_k are generated, and the corresponding values $f(X_k)$ are collected, for $k = 1, \ldots, N_{init}$. During active learning, the surrogate model of f is estimated from a finite number of function evaluations $f(X_k)$, for $k = 1, \ldots, N$, where N is increased sequentially between the initial value N_{init} and the maximum budget N_{max} of queries available. Each new sample X_{k+1} is determined by minimizing an *acquisition function*, which combines the surrogate with an *exploration function*, to reach an exploitation/exploration tradeoff. The exploration function is constructed based on the existing samples $\{X_k\}$, for $k = 1, \ldots, N$, to ensure that the feasible domain is sufficiently explored. The procedure aims to effectively reduce the objective function value of the posed problem within a small number of function evaluations.

In this paper, we propose fitting a *piecewise affine* (PWA) surrogate of the latent objective function f. PWA surrogate has two main benefits: (*i*) allow discontinuities introduced by sharp transitions induced by taking values in different classes of the categorical variables. In this case, instead of using one surrogate model for each categorical class as in [20, 48, 54], it is possible to adaptively update the number

of partitions allowed in the PWA function by analyzing the clusters of the queried samples. For example, one can initiate the surrogate fitting procedure by setting a maximum allowed number of partitions and then discard some partitions if the number of queried samples within these partitions is smaller than some fixed minimum values (cf. [9]); (*ii*) PWA surrogates have a direct mixed-integer linear reformulation and, therefore, can be minimized by efficient MILP solvers (e.g., Gurobi [23] and GLPK [38]). Also, we can explicitly reformulate and include linear equality and inequality constraints involving integer and one-hot encoded categorical variables in the standard MILP form to maintain their integrality during the solution process, enabling the possibility to make feasible queries during the acquisition step. Since the acquisition function includes both the *surrogate* and the *exploration function*, we will also define a suitable PWA *exploration function* that admits a MILP representation. The resulting approach, which we call PWAS, is summarized in Algorithm 1, whose steps will be described in detail in the next sections.

Algorithm 1 PWAS: Global Optimization Using Piecewise Affine Surrogates

---

Input: Lower and upper bounds ℓ_x, u_x, ℓ_y, u_y; linear constraint matrices $A_{\text{eq}}, B_{\text{eq}}, C_{\text{eq}}, A_{\text{ineq}}, B_{\text{ineq}}$ and C_{ineq}; linear constraint right-hand-side vectors b_{eq} and b_{ineq}; number n_d of categorical variables and n_i of possible categories, for $i = 1, \ldots, n_d$; initial number K of polyhedral partitions; number $N_{\text{init}} \geq 2$ of initial samples; number $N_{\text{max}} \geq N_{\text{init}}$ of maximum function evaluations; $\delta_1 \geq 0, \delta_2 \geq 0$ and $\delta_3 \geq 0$ if solve (13) in one step or $\delta \geq 0$ if solve (13) in multiple steps; solving strategy for (13): {"one-step" or "multi-steps"}.

---

1. Pre-process the optimization variables as described in Section 3.1;
2. $N \leftarrow 1, N_{\text{curr}}^* \leftarrow 1, f_{\text{curr}}^* \leftarrow +\infty$;
3. Generate N_{init} random scaled and encoded samples $\bar{X} = \{\bar{X}_1, \ldots, \bar{X}_{N_{\text{init}}}\}$ using one of the initial sampling methods reported in Section 3.5 based on the problem setup;
4. **While** $N \leq N_{\text{max}}$ **do**

   (a) Scale back and decode $\bar{X}_N$ to X_N, *i.e.*, $X_N = S(\bar{X}_N)$, and query $f_N = f(X_N)$;
   (b) **If** $f_N < f_{\text{curr}}^*$ **then update** $N_{\text{curr}}^* \leftarrow N, f_{\text{curr}}^* \leftarrow f_N$;
   (c) **If** $N \geq N_{\text{init}}$ **then**
      i. Update and fit the PWA separation function ϕ and PWA surrogate function $\hat{f}$ as described in Section 3.2;
      ii. Define the acquisition function a as in (13);
      iii. Solve the global optimization problem (13) and get $\bar{X}_{N+1}$ either in one-step or multi-steps;
   (d) $N \leftarrow N + 1$;

5. **End**.

Output: Best decision vector $X^* = X_{N_{\text{curr}}^*}$ found.

3.1 Change of Variables: Scaling and Encoding

Before attempting solving problem (1), we first rescale every continuous variable x_i into a new variable $\bar{x}_i \in [-1, 1]$ such that

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

Accordingly, the constraint matrices A_{eq} and A_{ineq} are rescaled to

$$\bar{A}_{\mathrm{eq}} = A_{\mathrm{eq}}\mathrm{diag}\left(\frac{u_x - \ell_x}{2}\right) \text{ and } \bar{A}_{\mathrm{ineq}} = A_{\mathrm{ineq}}\mathrm{diag}\left(\frac{u_x - \ell_x}{2}\right),$$

with the right-hand-side vectors updated as follows:

$$\bar{b}_{\mathrm{eq}} = b_{\mathrm{eq}} - A_{\mathrm{eq}}\left(\frac{u_x + \ell_x}{2}\right) \text{ and } \bar{b}_{\mathrm{ineq}} = b_{\mathrm{ineq}} - A_{\mathrm{ineq}}\left(\frac{u_x + \ell_x}{2}\right).$$

The intervals $[-1, 1]$ for the continuous variables are possibly further tightened by taking the updated inequality constraints (1c) (if they exist) into account (cf. [8]), i.e., for $i = 1, \ldots, n_c$, we set

$$\begin{aligned}
\bar{\ell}_x^i = \min_{\bar{x},y,z} \quad & e_i^\mathsf{T}[\bar{x}^\mathsf{T} \ y^\mathsf{T} \ z^\mathsf{T}]^\mathsf{T} \\
\text{s.t.} \quad & \bar{A}_{\mathrm{ineq}}\bar{x} + B_{\mathrm{ineq}}y + C_{\mathrm{ineq}}z \leq \bar{b}_{\mathrm{ineq}} \\
& \bar{x} \in [-1 \ 1]^{n_c}, \ y \in [\ell_y, u_y] \cap \mathbb{Z}, \ z \in \Omega_z,
\end{aligned}$$

and, similarly,

$$\begin{aligned}
\bar{u}_x^i = \max_{\bar{x},y,z} \quad & e_i^\mathsf{T}[\bar{x}^\mathsf{T} \ y^\mathsf{T} \ z^\mathsf{T}]^\mathsf{T} \\
\text{s.t.} \quad & \bar{A}_{\mathrm{ineq}}\bar{x} + B_{\mathrm{ineq}}y + C_{\mathrm{ineq}}z \leq \bar{b}_{\mathrm{ineq}} \\
& \bar{x} \in [-1 \ 1]^{n_c}, \ y \in [\ell_y, u_y] \cap \mathbb{Z}, \ z \in \Omega_z.
\end{aligned}$$

Here, e_i denotes the ith column of the identity matrix of the same dimension as vector X. We denote the resulting domain of the scaled continuous variables by $\Omega_x = [\bar{\ell}_x^1, \bar{u}_x^1] \times \ldots \times [\bar{\ell}_x^{n_c}, \bar{u}_x^{n_c}]$.

Let us assume that only a finite number $N_{\max}$ of queries can be made, which depends on the nature of function f (i.e., how expensive it is to evaluate) and the time available to solve the optimization problem. Moreover, we treat integer variables y differently depending on the relation between $N_{\max}$ and the number $\prod_{i=1}^{n_{\mathrm{int}}} n_i^{\mathrm{int}}$ of possible combinations of integer variables, where $n_i^{\mathrm{int}} = \lfloor u_y^i \rfloor - \lceil \ell_y^i \rceil + 1$ 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. To be described in Sects. 3.1.1 and 3.1.2 below, we will treat the vector y of integer variables in two distinct ways to make the exploration of the search space possibly more efficient. Specifically, y is treated as categorical when solving (1) in case $\prod_{i=1}^{n_{\mathrm{int}}} n_i^{\mathrm{int}} < N_{\max}$, i.e., when it may be possible to exhaustively list out all the potential combinations of the integer variables within $N_{\max}$ queries if no continuous or categorical variables are present; vice versa, we will maintain the optimization variables y_i integer. We note that this is a general heuristic we applied, which was empirically observed to be more efficient when handling integer variables. This heuristic is motivated by the acquisition strategies, which we will elaborate more in Sect. 3.4.

3.1.1 Treating Integer Variables as Categorical

The first scenario occurs when the number of possible combinations of integer variables $\prod_{i=1}^{n_{\text{int}}} n_i^{\text{int}} < N_{\text{max}}$. In this case, we treat all integer variables y_i as categorical (similarly to vector z) and one-hot encode them into further $d^{n_{\text{int}}}$ binary variables $\bar{y}_j \in \{0, 1\}$, for $j = 1, \ldots, d^{n_{\text{int}}}$, where $d^{n_{\text{int}}} = \sum_{i=1}^{n_{\text{int}}} n_i^{\text{int}}$. We also define $\Omega_y = \{\bar{y} \in \{0, 1\}^{d^{n_{\text{int}}}} : \sum_{j=1}^{n_i^{\text{int}}} \bar{y}_{j+d_y^{i-1}} = 1, \ \forall i = 1, \ldots, n_{\text{int}}\}$, where $d_y^0 = 0$ and $d_y^i = \sum_{j=1}^{i} n_j^{\text{int}}$ for $i = 1, \ldots, n_{\text{int}}$, and set $\bar{y} \in \Omega_y$. The constraint matrix B_{eq} (B_{ineq}) is modified accordingly into a new matrix $\bar{B}_{\text{eq}}$ ($\bar{B}_{\text{ineq}}$) by replacing each scalar entry B_{eq}^{ij} (B_{ineq}^{ij}) with the row vector obtained by multiplying the entry by the vector of integers that variable y_j can take, i.e.,

$$B_{\text{eq}}^{ij} \leftarrow B_{\text{eq}}^{ij} \left[\lceil \ell_y^j \rceil \ \ldots \ \lfloor u_y^j \rfloor \right] \in \mathbb{R}^{1 \times n_j^{\text{int}}}, \quad \forall j = 1, \ldots, n_{\text{int}},$$
$$B_{\text{ineq}}^{ij} \leftarrow B_{\text{ineq}}^{ij} \left[\lceil \ell_y^j \rceil \ \ldots \ \lfloor u_y^j \rfloor \right] \in \mathbb{R}^{1 \times n_j^{\text{int}}}, \forall j = 1, \ldots, n_{\text{int}}.$$

The new optimization vector becomes $\bar{X} = [\bar{x}^{\mathsf{T}} \ \bar{y}^{\mathsf{T}} \ z^{\mathsf{T}}]^{\mathsf{T}} \in \bar{\Omega}$, where $\bar{\Omega} = \Omega_x \times \Omega_y \times \Omega_z$, consisting of $n = n_c + d^{n_{\text{int}}} + d^{n_d}$ variables. As evaluating the objective function in (1) requires the original values in X, we denote by $S : \bar{\Omega} \mapsto \Omega$ the inverse scaling/encoding mapping of $\bar{X}$, i.e., $X = S(\bar{X})$. According to such a change of variables, problem (1) is now translated to

$$\begin{aligned}
\text{find } \bar{X}^* &\in \arg \min_{\bar{X} \in \bar{\Omega}} f(S(\bar{X})) \\
\text{s.t. } &\bar{A}_{\text{eq}} \bar{x} + \bar{B}_{\text{eq}} \bar{y} + C_{\text{eq}} z = \bar{b}_{\text{eq}} \\
&\bar{A}_{\text{ineq}} \bar{x} + \bar{B}_{\text{ineq}} \bar{y} + C_{\text{ineq}} z \leq \bar{b}_{\text{ineq}}.
\end{aligned} \tag{2a}$$

In the sequel, $D \subseteq \bar{\Omega}$ will denote the set of admissible vectors $\bar{X}$ satisfying the constraints in (2a).

3.1.2 Scaling Integer Variables

In the second scenario where $\prod_{i=1}^{n_{\text{int}}} n_i^{\text{int}} \geq N_{\text{max}}$, the integer variables are rescaled and treated as numeric variables $\bar{y}_i \in [-1, 1]$, $i = 1, \ldots, n_{\text{int}}$. In this case, we also keep the original n_{int} integer variables $y_i \in \mathbb{Z}$ in the model for the sole purpose of enforcing integrality constraints, as we link them with $\bar{y}_i$ by the scaling factors

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

Similar to the continuous variables, we can also further shrink the bounds on $\bar{y}_i$ by considering the updated inequality constraints (1c) (if present)

$$\bar{\ell}_y^i = \min_{\bar{x},\bar{y},z} e_{n_c+i}^\mathsf{T}[\bar{x}^\mathsf{T}\ \bar{y}^\mathsf{T}\ z^\mathsf{T}]^\mathsf{T}$$

$$\text{s.t.} \quad \bar{A}_{\text{ineq}}\bar{x} + B_{\text{ineq}}y + C_{\text{ineq}}z \le \bar{b}_{\text{ineq}}$$
$$\bar{x} \in \Omega_x,\ \bar{y} \in [-1\ 1]^{n_{\text{int}}},\ z \in \Omega_z,$$

$$\bar{u}_y^i = \max_{\bar{x},\bar{y},z} e_{n_c+i}^\mathsf{T}[\bar{x}^\mathsf{T}\ \bar{y}^\mathsf{T}\ z^\mathsf{T}]^\mathsf{T}$$

$$\text{s.t.} \quad \bar{A}_{\text{ineq}}\bar{x} + B_{\text{ineq}}y + C_{\text{ineq}}z \le \bar{b}_{\text{ineq}}$$
$$\bar{x} \in \Omega_x,\ y \in [-1\ 1]^{n_{\text{int}}},\ z \in \Omega_z.$$

We denote the domain of $\bar{y}$ after tightening as $\Omega_y = [\bar{\ell}_y^1, \bar{u}_y^1] \times \ldots \times [\bar{\ell}_y^{n_{\text{int}}}, \bar{u}_y^{n_{\text{int}}}]$. Accordingly, problem (1) is translated to

$$\text{find} \begin{bmatrix} \bar{X}^* \\ y^* \end{bmatrix} \in \arg \min_{\bar{X} \in \bar{\Omega},\, y \in [\ell_y, u_y] \cap \mathbb{Z}} f(S(\bar{X}))$$
$$\text{s.t. } \bar{A}_{\text{eq}}\bar{x} + B_{\text{eq}}y + C_{\text{eq}}z = \bar{b}_{\text{eq}}$$
$$\bar{A}_{\text{ineq}}\bar{x} + B_{\text{ineq}}y + C_{\text{ineq}}z \le \bar{b}_{\text{ineq}}, \tag{2b}$$

where now $\bar{X} = [\bar{x}^\mathsf{T}\ \bar{y}^\mathsf{T}\ z^\mathsf{T}]^\mathsf{T} \in \bar{\Omega}$ and consists of $n = n_c + n_{\text{int}} + d^{n_d}$ variables, with $\bar{\Omega} = \Omega_x \times \Omega_y \times \Omega_z$. And $S : \bar{\Omega} \mapsto \Omega$ is the new inverse scaling mapping. We will denote by $D \subseteq \bar{\Omega}$ the set of admissible vectors $\bar{X}$ such that the constraints in (2b) are satisfied for some vector $y \in [\ell_y, u_y] \cap \mathbb{Z}$.

3.2 Piecewise Affine Surrogate Function

When fitting a surrogate of the objective function, we treat the modified vector $\bar{X}$ as a vector in $\mathbb{R}^n$. We describe next how to construct a PWA surrogate function $\hat{f} : \mathbb{R}^n \mapsto \mathbb{R}$ such that $\hat{f}(\bar{X})$ approximates $f(S(\bar{X}))$.

Consider N samples $\bar{X}_1, \ldots, \bar{X}_N \in \mathbb{R}^n$ and their corresponding function evaluations $f(S(\bar{X}_1)), \ldots, f(S(\bar{X}_N)) \in \mathbb{R}$. We want to define the PWA surrogate function $\hat{f}$ over a polyhedral partition of $\bar{\Omega}$ into K regions. To this end, we consider the following convex *PWA separation function* $\phi : \mathbb{R}^n \mapsto \mathbb{R}$

$$\phi(\bar{X}) = \omega_{j(\bar{X})}^\mathsf{T}\bar{X} + \gamma_{j(\bar{X})}, \tag{3a}$$

where $\omega_j \in \mathbb{R}^n$ and $\gamma_j \in \mathbb{R}$, for $j = 1, \ldots, K$ are the parameters that need to be determined, with

$$j(\bar{X}) = \arg \max_{j=1,\ldots,K} \{\omega_j^\mathsf{T}\bar{X} + \gamma_j\}. \tag{3b}$$

We define the *PWA surrogate function* $\hat{f}$ as

$$\hat{f}(\bar{X}) = a_{j(\bar{X})}^\mathsf{T}\bar{X} + b_{j(\bar{X})}, \tag{3c}$$

where $a_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$, for $j = 1, \ldots, K$, are the surrogate parameters that need to be determined. Note that $\hat{f}$ is possibly non-convex and discontinuous.

We use the PARC algorithm recently proposed in [9] by one of the authors to fit the PWA separation and surrogate functions to obtain the required coefficients ω_j, γ_j, a_j, and b_j, for $j = 1, \ldots, K$. We stress that while the closed-form expression of $f \circ S$ as a function of $\bar{X}$ is generally unavailable and very expensive-to-evaluate for each given $\bar{X}$, evaluating its surrogate $\hat{f}$ is very cheap and, as we will show in Sect. 3.2.1, admits a simple mixed-integer linear encoding with K binary variables.

The number of partitions, K, is a hyper-parameter of the proposed global optimization algorithm that must be selected by trading off between having a more flexible surrogate function (large K) and reducing computational demands (small K). The PARC algorithm includes an adaptive mechanism to update K during surrogate fitting, specifically, if a partition contains fewer samples than a predefined minimum, it is discarded. The samples from these discarded partitions are then either reassigned to a neighboring partition or treated as outliers and excluded from surrogate fitting, depending on their function evaluations [9]. This adaptability makes the PARC algorithm more robust and flexible, allowing it to reduce computational complexity without significantly affecting prediction accuracy. For a detailed analysis of the PARC algorithm, we refer readers to [9]. For illustration purposes, we present two surrogate-fitting examples: one in a continuous domain and another in a mixed continuous and categorical domain. While these demonstrations use 2D problems for simplicity and better visualization, the PARC algorithm is capable of handling high-dimensional problems.

For the continuous function, we consider the Branin function [17]:

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

A PWA surrogate of f in (4) is fitted by the PARC algorithm utilizing 800 randomly generated training samples (x_{1k}, x_{2k}). The initial partition K was set to 10. Figure 2 show the final polyhedral partition induced by (3a). And Fig. 3 depicts the Branin function as represented both analytically and through the PWA surrogate fitted by PARC. We observe that the surrogate fitted by PARC captures the general shape of the nonlinear Branin function.

For the mixed continuous and categorical domain, we consider the following synthetic function:

$$f(x_1, x_2) = \begin{cases} x_1^2 + 2x_1 + 1 & x_2 = 0 \\ x_1 + 100 & x_2 = 1 \\ (1 - x_1)^3 & x_2 = 2 \end{cases} \tag{5}$$
$$-5 \le x_1 \le 5, \ x_2 \in \{0, 1, 2\},$$

where with different categorical values of x_2, function evaluations ($f(x_1, x_2)$) can vary significantly. We use 960 randomly generated training samples with 10 initial

Fig. 2 The polyhedral partition induced by (3a) with $K = 10$ initial partitions. Note that the final partition is also 10 (no partition is discarded). The dots in the figure are the training data for the PARC algorithm. Different colors indicate samples at different partitions. Brownish dots next to the partition numbers are the centroid of the training data within each partition
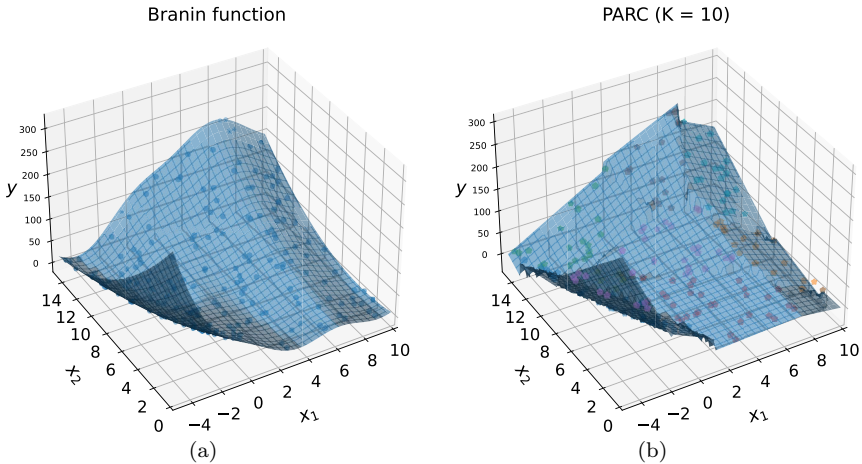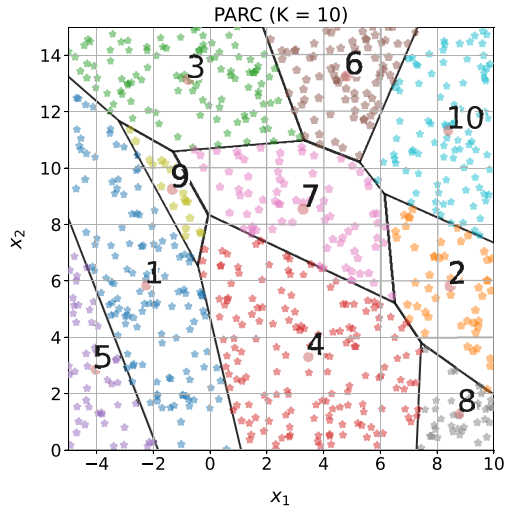

Fig. 3 **a** Branin function-analytical, and **b** Branin function fitted by PARC with $K = 10$. The dots in the figure are the test data (200 samples) for the PARC algorithm. Different colors in **b** indicate samples at different partitions

partitions (K) to fit the PWA surrogates. Figure 4 shows the final polyhedral partition induced by (3a). In this case, we observe that 2 initial partitions were discarded during surrogate fitting, resulting in 8 remaining partitions. In Fig. 5, we show the Function (5) fitted analytically, and fitted by PARC with $K = 8$ partitions, where we observe that PARC can make good predictions at different values of the categorical variable (x_2), despite their distinct characteristics.

We also remark that the goal of the current study is to obtain a highly accurate approximation of the objective function *around the global optimal solution* and not necessarily over the entire domain of $\bar{X}$, which usually requires much fewer samples. It is because, as the algorithm adaptively queries points to test from the domain, the

Fig. 4 The polyhedral partition induced by (3a) with $K = 10$ initial partitions. Note that the final partition is 8 (2 partitions are discarded). The dots in the figure are the training data for the PARC algorithm. Different colors indicate samples at different partitions. Brownish dots next to the partition numbers are the centroid of the training data within each partition
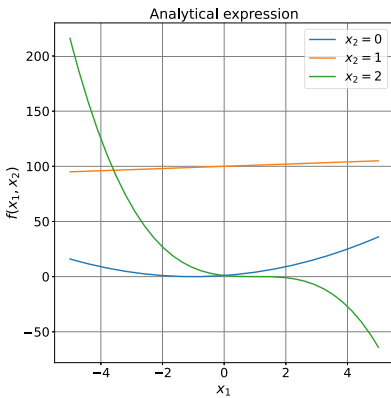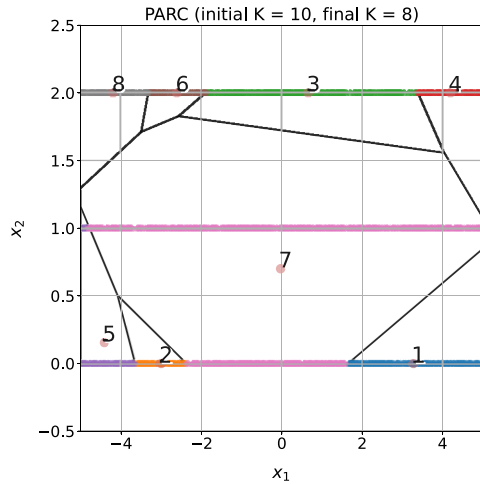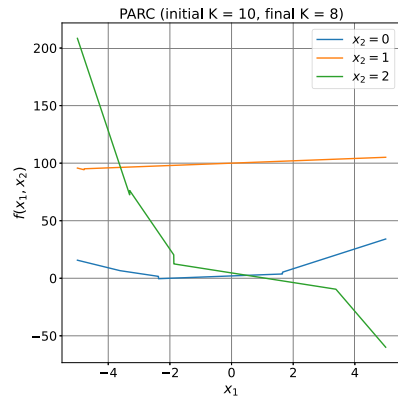
Fig. 5 **a** Function (5) evaluated analytically, and **b** Function (5) evaluated by the surrogates fitted by PARC with $K = 8$ final partitions

partitions associated with higher function evaluations (less relevant for optimization) will be sampled less frequently, and accurate prediction models for these partitions are not necessary. On the other hand, the regions with promising test points will be more frequently visited, resulting in better (more accurate) PWA surrogates within these partitions. An illustrative example is shown in Appendix A to demonstrate this remark.

3.2.1 Mixed-Integer Linear Encoding of the Surrogate

After learning the coefficients of ϕ and $\hat{f}$ by applying the PARC algorithm, in order to optimize over the surrogate function to acquire a new sample $\bar{X}_{N+1}$ by MILP, as we will describe in Sect. 3.4, we introduce K binary variables $\zeta_j \in \{0, 1\}$ and K real

variables $v_j \in \mathbb{R}$, for $j = 1, \ldots, K$. Here, $\zeta_j = 1$ if and only if $\bar{X}_{N+1}$ belongs to the jth polyhedral region of the partition induced by ϕ. The PWA separation function ϕ can be modeled by the following mixed-integer inequalities via the big-M method [9]:

$$
\begin{aligned}
\omega_j^\mathsf{T} \bar{X}_{N+1} + \gamma_j &\geq \omega_h^\mathsf{T} \bar{X}_{N+1} + \gamma_h - M_\phi (1 - \zeta_j), \ \forall h = 1, \ldots, K, h \neq j \\
\sum_{j=1}^{K} \zeta_j &= 1,
\end{aligned}
\tag{6}
$$

where M_ϕ is a large-enough constant, i.e., satisfies the inequality

$$
M_\phi \geq \max_{j,h=1,\ldots,K, \ \bar{X} \in D} (\omega_h - \omega_j)^\mathsf{T} \bar{X} + \gamma_h - \gamma_j.
$$

The PWA surrogate function $\hat{f}$ can be modeled by setting

$$
\hat{f}(\bar{X}) = \sum_{j=1}^{K} \zeta_j (a_j^\mathsf{T} \bar{X}_{N+1} + b_j) = \sum_{j=1}^{K} v_j,
$$

subject to

$$
\begin{aligned}
v_j &\leq a_j^\mathsf{T} \bar{X}_{N+1} + b_j - M_{sj}^-(1 - \zeta_j) \\
v_j &\geq a_j^\mathsf{T} \bar{X}_{N+1} + b_j - M_{sj}^+(1 - \zeta_j) \\
v_j &\geq M_{sj}^- \zeta_j \\
v_j &\leq M_{sj}^+ \zeta_j,
\end{aligned}
\tag{7}
$$

where M_{sj}^+, M_{sj}^- are large-enough constants satisfying the inequalities

$$
M_{sj}^+ \geq \max_{\bar{X} \in D} a_j^\mathsf{T} \bar{X} + b_j, \quad M_{sj}^- \leq \min_{\bar{X} \in D} a_j^\mathsf{T} \bar{X} + b_j,
$$

for $j = 1, \ldots, K$.

3.3 Exploration Function

Solely minimizing the surrogate function $\hat{f}$ may easily miss the global optimum. In order to properly explore the admissible set D we need to introduce an *exploration function* $E : \bar{\Omega} \mapsto \mathbb{R}$. Due to the different numerical properties of continuous, integer, and categorical variables, we consider different exploration strategies for each of them that admit a MILP representation. Specifically, we use a distance-based exploration method for continuous and integer variables if the latter are not one-hot encoded (as described in Sect. 3.1.2) and a frequency-based exploration method for one-hot encoded categorical and integer variables (in the alternative scenario described in Sect. 3.1.1). In the following, we discuss the distance-based and frequency-based methods in a general

manner, and we will dive into specifics of the exploration functions for our problem of interest when we discuss the acquisition function in Sect. 3.4.

3.3.1 Distance-Based Exploration: "Max-box" Method

We want to define a function $E_{ct} : \mathbb{R}^{n_{ct}} \mapsto \mathbb{R}$ mapping a generic numeric vector $\bar{x} \in \mathbb{R}^{n_{ct}}$ into a nonnegative value $E_{ct}(\bar{x})$ with the following features: (i) is zero at given samples $\bar{x}_1, \ldots, \bar{x}_N$; (ii) grows away from them; and (iii) admits a PWA representation. To this end, we consider the boxes $B_i(\beta_{ct}) = \{\bar{x} : \|\bar{x} - \bar{x}_i\|_\infty \leq \beta_{ct}\}$ and set $E_{ct}(\bar{x}) = \min\{\beta_{ct} \geq 0 : \bar{x} \in B_i(\beta_{ct}) \text{ for some } i = 1, \ldots, N\}$. Then, maximizing $E_{ct}(\bar{x})$ is equivalent to finding the largest value β_{ct} and a vector $\bar{x}^*$ outside the interior of all boxes $B_i(\beta_{ct})$, a problem that can be solved by the following MILP

$$
\begin{aligned}
\bar{x}^* \in \quad & \arg \max_{\bar{x}, \beta_{ct}, \delta^+, \delta^-} \beta_{ct} \\
\text{s.t} \quad & \bar{x}^l - \bar{x}_i^l \geq \beta_{ct} - M_E(1 - \delta_{il}^+), \quad \forall l = 1, \ldots, n_{ct}, \quad \forall i = 1, \ldots, N \\
& -\bar{x}^l + \bar{x}_i^l \geq \beta_{ct} - M_E(1 - \delta_{il}^-), \quad \forall l = 1, \ldots, n_{ct}, \quad \forall i = 1, \ldots, N \\
& \delta_{il}^+ \leq 1 - \delta_{il}^-, \qquad\qquad\qquad \forall l = 1, \ldots, n_{ct}, \quad \forall i = 1, \ldots, N \\
& \sum_{l=1}^{n_{ct}} \delta_{il}^+ + \delta_{il}^- \geq 1, \qquad\qquad \forall i = 1, \ldots, N \\
& \beta_{ct} \geq 0, \;\; \bar{x} \in D,
\end{aligned}
\tag{8}
$$

where l denotes the lth component of vector $\bar{x}$, $\delta_{il}^-, \delta_{il}^+ \in \{0, 1\}$ are auxiliary optimization variables introduced to model the violation of at least one of the linear inequalities that define the box $B_i(\beta_{ct})$, and M_E is a large-enough constant satisfying the following inequality

$$
M_E \geq 2 \left(\max_{l=1,\ldots,n_{ct}} \bar{u}_x^l - \min_{l=1,\ldots,n_{ct}} \bar{\ell}_x^l \right),
$$

where $\bar{u}_x^l$ and $\bar{\ell}_x^l$ are the upper and lower bounds, respectively, of the lth component of vector $\bar{x}$.

Figure 6 shows an example, where we apply the max-box exploration method to $\bar{x} \in \mathbb{R}^2$ and $D = [-3, 9] \times [-2, 8]$. We start with three existing samples $\bar{x}_1, \bar{x}_2$, and $\bar{x}_3$. After 20 samples, we get the samples reported in the figure, which shows that, indeed, the max-box exploration method effectively explores the feasible region D.

3.3.2 Frequency-Based Exploration: "Hamming Distance" Method

Unlike the case of continuous variables treated in the previous section, to account for the frequency of occurrence of a particular combination of binary variables we use the Hamming distance, defined as follows: given two binary vectors $z = [z^1, \ldots, z^d]^\mathsf{T} \in \{0, 1\}^d$ and $z_i = [z_i^1 \;\; \cdots \;\; z_i^d]^\mathsf{T} \in \{0, 1\}^d$, the Hamming distance between z and z_i is defined by the number of different components between them
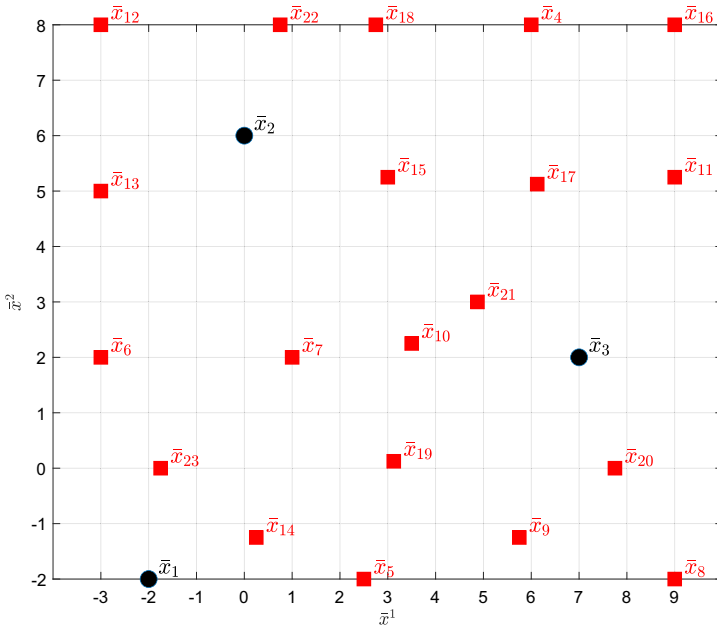
Fig. 6 Illustrative example of the max-box exploration function in 2D. The black dots denote the initial samples. The red squares denote the samples generated using the max-box exploration method. The subscript number indicates the order of the point generated

$$d_H(z, z_i) = \sum_{m=1}^{d} |z^m - z_i^m|, \tag{9}$$

which can be encoded as the following linear expression

$$d_H(z, z_i) = \sum_{m:z_i^m=0} z^m + \sum_{m:z_i^m=1} (1 - z^m). \tag{10}$$

We consider the exploration function $E_{dt} : \{0, 1\}^d \mapsto \mathbb{R}$ such that $E_{dt}(z)$ quantifies the average number of different binary components between z and the given N vectors $z_1, \ldots, z_N$

$$E_{dt}(z) = \frac{1}{dN} \sum_{i=1}^{N} d_H(z, z_i).$$

Hence, a binary vector z^* with maximum average Hamming distance $E_{dt}(z^*)$ from the current samples $z_1, \ldots, z_N$ can be determined by solving the following MILP

$$z^* \in \arg\max_{z \in D} \; E_{dt}(z). \tag{11}$$

Table 1 Illustrative example of the Hamming distance exploration function

Sample	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

Sample	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

Table 1 shows an example in which we have three categorical variables $Z = [Z^1, Z^2, Z^3]$, where $Z^1 \in \{A, B\}$, $Z^2 \in \{A, B, C, D, E\}$, and $Z^3 \in \{A, B, C\}$. We start with three initial samples $Z_1 = [A, E, C]$, $Z_2 = [B, B, B]$, and $Z_3 = [A, D, C]$. First, we binary encode the categorical variables, getting the corresponding vectors z_1, z_2, $z_3 \in \{0, 1\}^{10}$. Then, we solve the optimization problem (11) to identify $z_4 = z^*$ and its corresponding decoded form Z_4. The table shows the categorical values $Z_4, \ldots, Z_{23}$ generated in 20 subsequent sampling steps, which shows that a diverse set of categorical variables are obtained when applying the Hamming distance exploration method.

We remark that both exploration functions (distance-based and frequency-based) are independent from the surrogate and does not require explicit uncertainty measurements, making it flexible to be integrated with different types of surrogates (e.g., polynomials).

3.4 Acquisition Function

The surrogate and exploration functions defined in Sects. 3.2 and 3.3 can be combined into the following *acquisition problem*

$$\text{find } \bar{X}^* \in \arg \min_{\bar{X} \in D} \hat{f}(\bar{X}) - \delta(E_{ct}(\bar{x}) + E_{dt}([\bar{y}^\mathsf{T} \ z^\mathsf{T}]^\mathsf{T})), \tag{12a}$$

when integer variables are treated as categorical as described in Sect. 3.1.1, or into

$$\text{find } \begin{bmatrix} \bar{X}^* \\ y^* \end{bmatrix} \in \arg \min_{\bar{X} \in D, y \in [\ell_y, u_y] \cap \mathbb{Z}} \hat{f}(\bar{X}) - \delta(E_{ct}([\bar{x}^\mathsf{T} \ \bar{y}^\mathsf{T}]^\mathsf{T}) + E_{dt}(z)), \tag{12b}$$

when the integer vector y is scaled as described in Sect. 3.1.2. In (12), the nonnegative scalar δ is called the *exploration parameter* and decides the tradeoff between exploiting the surrogate $\hat{f}(\bar{X})$ and promoting the exploration of the feasible domain D. In the sequel, we will refer to the cost function $a : \bar{\Omega} \mapsto \mathbb{R}$ in (12a) or (12b) as the *acquisition function*. By construction, Problem (12) can be solved by MILP. An optimal vector $\bar{X}^*$ of its solution, once scaled and decoded back, defines the next sample X_{N+1} to query

for the corresponding function value $f_{N+1} = f(X_{N+1})$. Note that X_{N+1} satisfies all the constraints in (1) since $\bar{X}^* \in D$.

The direct formulation (12) can be further improved to ease the selection of δ and make the exploration more homogenous with respect to all types of variables (continuous, integer, or categorical). In fact, the formulation in (12) has the following possible drawbacks:

(i) The relative magnitude between $\hat{f}(\bar{X})$ and $E(\bar{X})$ is hard or impossible to estimate a priori, making the value for the exploration parameter δ hard to select.

(ii) By using the same exploration parameter δ for the exploration function of each type of variable, we implicitly assumed that the relative magnitude of each exploration function is comparable, which may not be the case.

(iii) When the integer variables y are not one-hot encoded as described in Sect. 3.1.2, the max-box exploration function is applied to the combined vector $[\bar{x}^\mathsf{T} \ \bar{y}^\mathsf{T}]^\mathsf{T}$ (see (12b)) and two problems can occur. Firstly, as shown in (2), even though $\bar{y}_i$ is a continuous variable, because of the presence of the corresponding auxiliary integer variable y_i, it can only be changed in discrete steps, unlike the remaining variables $\bar{x}_j$. As a result, when finding the max box, one unit change in an integer variable can be reflected as a more significant change of its corresponding scaled variable $\bar{y}$, therefore promoting the exploration of directions with more variations in the integer variables $\bar{y}_i$ than in the continuous variables $\bar{x}_j$. Secondly, due to possibly different lower and upper bounds and therefore scaling factors of integer variables, unit changes of them may cause different changes in size of their corresponding scaled variables.

To address the aforementioned issues, given N samples $(\bar{X}_i, f(S(\bar{X}_i)))$, for $i = 1, \ldots, N$, we reformulate the acquisition problems (12), respectively, as follows:

$$\text{find } \bar{X}_{N+1} \in \arg \min_{\bar{X} \in D} \frac{\hat{f}(\bar{X})}{\Delta F} - \delta_1 E_{ct}(\bar{x}) - \delta_2 E_{ct}(\bar{y}) - \delta_3 E_{dt}(z), \tag{13a}$$

$$\text{find } \begin{bmatrix} \bar{X}_{N+1} \\ y_{N+1} \end{bmatrix} \in \arg \min_{\bar{X} \in D, y \in [\ell_y, u_y] \cap \mathbb{Z}} \frac{\hat{f}(\bar{X})}{\Delta F} - \delta_1 E_{ct}(\bar{x}) - \delta_2 E_{dt}(\bar{y}) - \delta_3 E_{dt}(z), \tag{13b}$$

where

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

and $\epsilon_{\Delta F} > 0$ is a threshold to prevent division by zero. The scaling factor ΔF eases the selection of the exploration parameters δ_1, δ_2, and δ_3 by making the surrogate term comparable to the exploration terms (cf. [8]).

An alternative to solve the optimization problem (13) in one step is to consider only one exploration term at a time, therefore solving the problem in three consecutive steps (that will be referred to as the "multi-step" approach), where, at each step, the problem

is only solved with respect to one variable type. The remaining variables are treated as constants at either the value associated with the current best vector $X_{N_{\text{curr}}^*}$ or at the new value optimized during the multi-step operation. The advantage of serializing the optimization is that the relative value of δ_1, δ_2, and δ_3 is no longer relevant, and therefore we set $\delta_1 = \delta_2 = \delta_3 = \delta$, where δ is the only tradeoff hyperparameter to choose. Empirical findings indicate that a multi-step optimization approach for the acquisition function often yields superior results compared to a one-step approach. The one-step approach poses challenges in effectively tuning the exploration parameters δ_1, δ_2, and δ_3, which can hinder performance. Additionally, these findings also motivate our heuristic for different treatments of integer variables as discussed in Sect. 3.1. When optimize the acquisition function in multiple steps, if the number of possible integer combinations is relatively small compared to the maximum allowed number of black-box function evaluations, the integer variable combinations may be exhaustively enumerable within the allowed evaluations. Treating them categorically helps prevent premature convergence and ensures a more thorough exploration of the solution space.

A further heuristic is applied to restrict the number of binary variables used to encode the max-box exploration function (8) and therefore limit them as the number N of samples increases. Specifically, given an upper bound N_{Emax} defined by the user depending on the computational power available, we only consider the most recent N_S samples in the exploration function (we will use $N_S = 20$ in our experiments) when $Nn_c \geq N_{\text{Emax}}$ or $Nn_{\text{int}} \geq N_{\text{Emax}}$ (when integer variables are not one-hot encoded). We note that the surrogate function is approximated using all the existing samples. The rationale behind the heuristic is that, as the number N of queried samples grows, the surrogate itself should already discourage the exploration around the older samples not included in the exploration term, where the surrogate function, most likely, takes large values.

3.5 Initial Sampling Strategies

The values of the initial samples $X_1, \ldots, X_{N_{\text{init}}}$ can significantly impact the final solution X^* obtained after N_{max} steps. Moreover, one of the main motivations of the proposed method is its ability to handle mixed-integer constraints on the optimization variables. We propose different initial sampling strategies to efficiently acquire N_{init} scattered feasible samples depending on the constraints and types of optimization variables present in the problem:

(i) When only box constraints are present, we use the *Latin Hypercube Sampling* (LHS) [39] method as in [8].
(ii) When both box constraints and linear equality and/or inequality constraints are present, we consider the following alternatives:

  – If only continuous variables are present, we use the Double Description Method [46] to generate the n_V vertices of the convex polytope given by the linear and box constraints. If $n_V < N_{\text{init}}$ and only inequality constraints are involved in (1), additional feasible samples can be generated via linear combinations of the vertices; if $n_V < N_{\text{init}}$ and equality constraints are

also present, the generated n_V vertices can be used to define initial boxes and additional scattered feasible samples are generated by solving MILPs sequentially with the max-box exploration function discussed in Sect. 3.3.1 as the objective function. In this way, the constraints in (1) can be enforced in the formulation.

– If integer and/or categorical variables are also present, the algorithm first attempts to generate samples using LHS and filters out the infeasible ones. If the number of generated feasible samples is insufficient, which may happen when the constraints are hard to fulfill by random sampling, we generate scattered samples by solving MILPs sequentially using the exploration functions discussed in Sect. 3.3 as the objective functions and incorporating the given constraints to ensure sample feasibility.

4 Preference-Based Learning

We want to extend the global optimization method introduced in the previous sections to handle cases in which quantifying an objective function $f(X)$ as in (1) can be hard, if not impossible. For example, if multiple objectives are involved, defining their relative weights a priori to form a single objective function can be difficult. In such cases, expressing a *preference* between the outcomes of two decision vectors X_1 and X_2 can be much simpler for a decision maker than quantifying the outcomes. Accordingly, we define the following *preference function* $\pi : \Omega \times \Omega \to \{-1, 0, 1\}$ (cf. [10])

$$\pi(X_1, X_2) = \begin{cases} -1 & \text{if } X_1 \text{ "better" than } X_2 \\ 0 & \text{if } X_1 \text{ "as good as" } X_2 \\ 1 & \text{if } X_2 \text{ "better" than } X_1. \end{cases} \tag{14}$$

In this case, we are interested in finding a feasible optimization vector X^* that wins or ties the pairwise comparisons with any other feasible X according to the preference function π, i.e., the optimization problem (1) is replaced by

$$\text{find } X^* \text{ such that } \pi(X^*, X) \le 0, \ \forall X \in D. \tag{15}$$

We describe next a variant of Algorithm 1, that we call as PWASp, for solving the preference-based optimization problem (15).

Let the optimization vector X be first pre-processed to $\bar{X}$ (e.g., scaling and/or encoding) as described in Sect. 3.1. Given N samples $\bar{X}_1, \ldots, \bar{X}_N$ and M_c preferences $\pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) \in \{-1, 0, 1\}$, for $k = 1, \ldots, M_c$, where $M_c = N - 1$, we aim to fit a PWA surrogate model reflecting the preference relations among different samples. Since function evaluations are not available, here, the preferences $\pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k}))$

are used to shape the surrogate function $\hat{f}(\bar{X})$ by imposing the following constraints:

$$
\begin{aligned}
\hat{f}(\bar{X}_{1,k}) &\leq \hat{f}(\bar{X}_{2,k}) - \sigma \quad \forall k : \pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) = -1 \\
\hat{f}(\bar{X}_{2,k}) &\leq \hat{f}(\bar{X}_{1,k}) - \sigma \quad \forall k : \pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) = 1 \\
|\hat{f}(\bar{X}_{1,k}) - \hat{f}(\bar{X}_{2,k})| &\leq \sigma \quad \forall k : \pi(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) = 0,
\end{aligned}
\tag{16}
$$

where $(S(\bar{X}_{1,k}), S(\bar{X}_{2,k})) = (X_{1,k}, X_{2,k})$ are pairs of compared samples, with $X_{1,k}$, $X_{2,k} \in \{X_1, \ldots, X_N\}$, for $k = 1, \ldots, M_c$. Here, $\sigma > 0$ is a given constant, used to avoid the trivial solution $\hat{f}(\bar{X}) \equiv 0$.

To identify the PWA separation function $\phi(\bar{X})$, we first use K-means [36] to cluster the samples, and then use softmax regression [16, 55] to fit the coefficients. The assignment $j(\bar{X})$ of each sample $\bar{X}$ to each region of the partition is then determined. Following that, different from the PARC algorithm, we determine the coefficients a_j, b_j defining the PWA surrogate function $\hat{f}(\bar{X})$ by minimizing the sum $\sum_{k=1}^{M_c} \epsilon_k$ of the violations of the preference constraints (16) under an additional ℓ_∞-regularization term. Specifically, the coefficients a_j, b_j are obtained by solving the following linear programming (LP) problem:

$$
\begin{aligned}
\min_{\epsilon_k, \xi, a, b} \quad & \sum_{k=1}^{M_c} \epsilon_k + \alpha \xi \\
\text{s.t.} \quad & \hat{f}(\bar{X}_{1,k}) + \sigma \leq \hat{f}(\bar{X}_{2,k}) + \epsilon_k \quad \forall k : \pi(X_{1,k}, X_{2,k}) = -1 \\
& \hat{f}(\bar{X}_{2,k}) + \sigma \leq \hat{f}(\bar{X}_{1,k}) + \epsilon_k \quad \forall k : \pi(X_{1,k}, X_{2,k}) = 1 \\
& |\hat{f}(\bar{X}_{1,k}) - \hat{f}(\bar{X}_{2,k})| \leq \sigma + \epsilon_k \quad \forall k : \pi(X_{1,k}, X_{2,k}) = 0 \\
& \xi \geq \pm a_j^l, \ l = 1, \ldots, n \\
& \xi \geq \pm b_j,
\end{aligned}
\tag{17}
$$

where $\alpha > 0$ is the regularization parameter; $\xi \in \mathbb{R}$ is a new optimization variable introduced to linearly encode the ℓ_∞-regularization of the coefficients; and l denotes the lth component of the vector.

Once the surrogate model is obtained, the same procedure as in PWAS can be followed to construct the acquisition function. This acquisition function is then optimized to identify the next sample, $X_{N+1} = S(\bar{X}_{N+1})$. The new sample is subsequently compared with the current best vector, $X_{N_{\text{curr}}^*}$, to obtain the new preference assessment. The various steps involved in PWASp are summarized in Algorithm 2.

5 Optimization Benchmarks

To illustrate the effectiveness of PWAS and PWASp in solving the target problems (1) and (15), we have considered different mixed-variable global optimization benchmarks, including three unconstrained synthetic benchmarks as well as two unconstrained real-world benchmarks (taken from [51]) and two constrained mixed-

variable synthetic problems. Computations are performed on an Intel i7-8550U 1.8-GHz CPU laptop with 24GB of RAM. The MILP problem in the acquisition step is formulated with the PuLP library [41] and solved by Gurobi's MILP solver [23].

For each benchmark, the function evaluations are fed into PWAS to fit the surrogate, while the explicit function expressions remain unknown to PWAS. As in [51] the benchmark problems are solved via maximization, we use the values $-f(X)$ when running PWAS. As for PWASp, the objective function serves as a synthetic decision-maker whose evaluations are only used to express the preference between two decision vectors, namely $\pi(X_1, X_2) = -1$ if $f(X_1) > f(X_2)$, $\pi(X_1, X_2) = 1$ if $f(X_1) < f(X_2)$, or zero otherwise. In other words, PWASp only has access to the queried preferences (14) and not the explicit function expressions nor their evaluations $f(X_N)$. Specifically, $N_{max} - 1$ pairwise comparisons are obtained for each benchmark when solved by PWASp.

The performance of PWAS and PWASp on the unconstrained benchmarks are compared with the following solvers: CoCABO-auto [51], CoCABO-0.5 [51], One-hot BO [21], SMAC [27], TPE [11], and EXP3BO [20] as noted in [51] as well as MISO [47] and NOMAD [4, 5]. A brief summary of each algorithm is provide in Appendix C. CoCABO-auto and CoCABO-0.5 are selected because the authors noted that they consistently show competitive performance [51]. MISO and NOMAD are selected since they are noted as the best performers among all the algorithms tested in [50]. The settings of the first six algorithms compared are available in [51]. As for MISO

Algorithm 2 PWASp: Preference-based Optimization Using Piecewise Affine Surrogates

Input: Lower and upper bounds ℓ_x, u_x, ℓ_y, u_y; linear constraint matrices $A_{eq}, B_{eq}, C_{eq}, A_{ineq}, B_{ineq}$, and C_{ineq}; linear constraint right-hand-side vectors b_{eq} and b_{ineq}; number n_d of categorical variables and n_i of possible categories, for $i = 1, \ldots, n_d$; initial number K of polyhedral partitions; number $N_{init} \geq 2$ of initial samples to compare; maximum number $N_{max} - 1$ of comparisons, $N_{max} \geq N_{init}$; $\delta_1 \geq 0, \delta_2 \geq 0$ and $\delta_3 \geq 0$ if solve (13) in one step or $\delta \geq 0$ if solve (13) in multiple steps; solving strategy for (13): {"one-step" or "multi-steps"}.

1. Pre-process the optimization variables as described in Section 3.1;
2. $N \leftarrow 1, i^* \leftarrow 1$;
3. Generate N_{init} random and encoded samples $\bar{X} = \{\bar{X}_1, \ldots, \bar{X}_{N_{init}}\}$ using one of the initial sampling methods described in Section 3.5 based on the problem setup;
4. **While** $N < N_{max}$ **do**

    (a) **If** $N \geq N_{init}$ **then**

        i. Update and fit the PWA separation function ϕ and PWA surrogate function $\hat{f}$ as described in Section 4;

        ii. Define the acquisition function a as in (13);

        iii. Solve the MILP problem (13) and get $\bar{X}_{N+1}$, either in one step or multiple steps;

    (b) $i(N) \leftarrow i^*, j(N) \leftarrow N + 1$

    (c) Query preference $\pi(X_{i(N)}, X_{j(N)})$;

    (d) **If** $\pi(X_{i(N)}, X_{j(N)}) = 1$ **then set** $i^* \leftarrow j(N)$

    (e) $N \leftarrow N + 1$;

5. **End**.

Output: Best vector $X^* = X_{i*}$ encountered.

Table 2 Benchmark problem specifications

Benchmark	n_c	n_{int}	n_d	n_i
Func-2C	2	0	2	{3, 3}
Func-3C	2	0	3	{3, 3, 3}
Ackley-5C	1	0	5	{17, 17, 17, 17, 17}
XG-MNIST	4	1	3	{2, 2, 2}
NAS-CIFAR10	21	1	5	{3, 3, 3}
Horst6-hs044-modified	3	4	2	{3, 2}
ros-cam-modified	2	1	2	{2, 2}

and NOMAD, we kept their default algorithm settings, with integer and categorical variables declared as referenced by the corresponding algorithms [4, 5, 47]. To have fair comparisons, we performed the same initial and maximum number of black-box function evaluations ($N_{\text{init}} = 20$ and $N_{\text{max}} = 100$) as indicated in [51] on the synthetic and real-world benchmarks for all the tested algorithms except for NOMAD. NOMAD starts the optimization process with an initial guess [5]. When solving the problems using PWAS or PWASp, the multi-step solution strategy is applied in the acquisition step with $\delta_1 = \delta_2 = \delta_3 = 0.05$ or $\delta_1 = \delta_2 = \delta_3 = 1$, respectively for PWAS and PWASp. The initial number K of polyhedral partitions is set to 20 for both PWAS and PWASp in all the benchmarks. We stress that here the function evaluations for PWASp are solely reported for performance comparisons and are not attainable to PWASp during optimization. Table 2 summarizes the tested benchmark problems, while a detailed description of the benchmarks is reported in Appendix B.

The optimization results obtained by CoCaBO-auto, CoCaBO-0.5, One-hot BO, SMAC and TPE for unconstrained benchmark were read from Figure 4 in [51] using GetData Graph Digitizer [19]. Regarding MISO and NOMAD (version 4), we retrieved their packages from the GitHub repository. We performed 20 random repetitions for the unconstrained synthetic problems (Func-2C, Func-3C, and Ackley-5C) and 10 random repetitions for the unconstrained real-world problems (XG-MNIST and NAS-CIFAR10) as reported in [51].

Regarding problems with constraints, we consider the two mixed-variable synthetic problems reported in Appendix B.3, whose specifications are also reported in Table 2. We do not consider other solvers than PWAS and PWASp as they either do not support constraint handling with mixed variables or allow infeasible samples during the optimization process. Thus, a systematic comparison is not performed for the constrained problems. Instead, the results are compared against the analytic global optimum. Here, we set $N_{\text{max}} = 100$ and $N_{\text{init}} = \lceil N_{\text{max}}/4 \rceil = 25$, $K = 20$ initial clusters, and the exploration parameters $\delta_1 = \delta_2 = \delta_3 = 0.05$ when using PWAS or $\delta_1 = \delta_2 = \delta_3 = 1$ with PWASp. We run PWAS and PWASp 20 times from different random seeds on these two problems.

Table 3 Best value found on benchmark Func-2C [51] after 100 and 200 black-box function evaluations (maximum = 0.2063). Statistics are obtained over 20 runs

Algorithm	After 100 evaluations		After 200 evaluations	
	Mean	Std	Mean	Std
PWAS	0.2049	0.0022	0.2061	0.0002321
PWASp	0.1813	0.0443	0.1889	0.0449
CoCaBO-auto	0.1219	0.0172	0.2041	0.0057
CoCaBO-0.5	0.1352	0.01620	0.2041	0.0057
One-hot BO	0.009524	0.02158	0.01524	0.02064
SMAC	0.06381	0.01746	0.07714	0.01556
TPE	0.1273	0.0184	0.1743	0.01650
EXP3BO	0.05524	0.01429	0.1105	0.01650
MISO	0.2063	0.0000	0.2063	0.0000
NOMAD	0.1700	0.0736	0.1754	0.07557

Table 4 Best value found on benchmark Func-3C [51] after 100 and 200 black-box function evaluations (maximum = 0.7221). Statistics are obtained over 20 runs

Algorithm	After 100 evaluations		After 200 evaluations	
	Mean	Std	Mean	Std
PWAS	0.5282	0.2117	0.6450	0.0972
PWASp	0.4542	0.2078	0.5106	0.1665
CoCaBO-auto	0.4993	0.0299	0.6912	0.0169
CoCaBO-0.5	0.5371	0.0503	0.6991	0.0205
One-hot BO	0.007670	0.04956	0.1076	0.0606
SMAC	0.1084	0.04016	0.1965	0.0339
TPE	0.2672	0.0472	0.4914	0.5308
EXP3BO	0.1784	0.0393	0.2515	0.0330
MISO	0.7221	0.0000	0.7221	0.0000
NOMAD	0.6618	0.1610	0.6860	0.1615

5.1 Results and Discussions

The optimization results for unconstrained benchmarks are reported in Tables 3, 4, 5, 6 and 7, and illustrated in Figs. 7, 8 and 9. The convergence plots (Fig. 7) show how different algorithms perform across various benchmarks in terms of increasing (reducing) the objective function value over time for maximization (minimization) problems. To avoid clutter, only mean values are plotted in Fig. 7 with the standard deviations at evaluations 100 and 200 reported in Tablea 3, 4, 5, 6 and 7. The performance and data profiles (Figs. 8, 9) are generated based on the guidelines noted in [45] with convergence tolerance τ set to 0.5 and 0.1. Performance profiles compare the efficiency of algorithms by looking at the performance ratio (relative performance) across a set of problems, while data profiles illustrate the percentage of problems that

Table 5 Best value found on benchmark Ackley-5C [51] after 100 and 200 black-box function evaluations (maximum = 0). Statistics are obtained over 20 runs

Algorithm	After 100 evaluations		After 200 evaluations	
	Mean	Std	Mean	Std
PWAS	−1.1148	0.4077	−0.7108	0.3320
PWASp	−1.8857	0.5795	−1.6462	0.5422
CoCaBO-auto	−2.5120	0.602	−1.9244	0.5512
CoCaBO-0.5	−2.8415	0.0488	−2.0073	0.0488
One-hot BO	−3.076	0.0483	−2.5341	0.3024
SMAC	−3.0073	0.2488	−1.710	0.2393
TPE	−3.4659	0.2000	−2.7976	0.2487
MISO	−1.6389	0.1388	−1.5582	0.06218
NOMAD	−2.0175	0.2015	−1.5467	0.01437

Table 6 Best value found on benchmark XG-MNIST [51] after 100 and 200 black-box function evaluations. Statistics are obtained over 10 runs

Algorithm	After 100 evaluations		After 200 evaluations	
	Mean	Std	Mean	Std
PWAS	0.9585	0.0030	0.9609	0.0029
PWASp	0.9576	0.0036	0.9615	0.0028
CoCaBO-auto	0.9639	0.0004	0.9653	0.0004
CoCaBO-0.5	0.9731	0.0008	0.9741	0.0008
One-hot BO	0.9541	0.0019	0.9556	0.0015
SMAC	0.9651	0.0012	0.9681	0.0012
TPE	0.9656	0.0007	0.9679	0.0007
EXP3BO	0.9691	0.0005	0.9706	0.0005
MISO	0.9574	0.0071	0.9594	0.0078
NOMAD	0.9528	0.0138	0.9564	0.0146

can be solved as a function of the equivalent number of simplex gradients (function evaluations) [45]. These profiles are particularly useful for comparing and analyzing the short-term behavior of algorithms, especially when computational resources are limited.

In general, PWAS and PWASp can effectively increase (decrease for minimization problems) the objective function values within a small number of function evaluations or comparisons. In fact, as shown in Tables 3, 4, 5, 6 and 7, the best values achieved by PWAS and PWASp after 100 evaluations are often already better or comparable to the results obtained by some other algorithms after 200 evaluations. These observations are also reflected in Figs. 7, 8 and 9, where we observe that PWAS, PWASp, and MISO consistently demonstrate superior early-stage performance. For $\tau = 0.5$, they can solve all the benchmark problems with fewer than 10 equivalent simplex gradients. This suggests that these three algorithms are well-suited for applications

Table 7 Best value found on benchmark NAS-CIFAR10 [51] after 100 and 200 black-box function evaluations. Statistics are obtained over 10 runs

Algorithm	After 100 evaluations		After 200 evaluations	
	Mean	Std	Mean	Std
PWAS	0.9440	0.0024	0.9462	0.0016
PWASp	0.9409	0.0052	0.9454	0.0019
CoCaBO-auto	0.9446	0.0017	0.9454	0.0017
CoCaBO-0.5	0.9458	0.0014	0.9468	0.0004
One-hot BO	0.9438	0.0006	0.9451	0.0006
SMAC	0.9422	0.0004	0.9436	0.0004
TPE	0.9427	0.0006	0.9443	0.0007
MISO	0.9442	0.0020	0.9447	0.0035
NOMAD	0.9385	0.0231	0.9391	0.0341
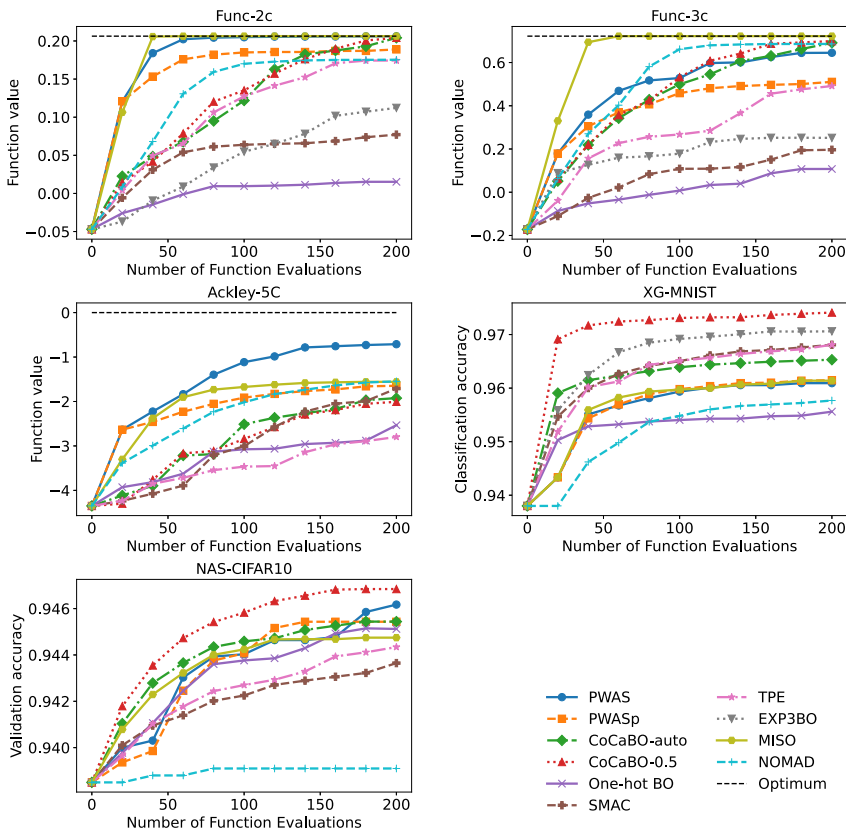
Fig. 7 Convergence graph of tested algorithms on the unconstrained benchmarks. The mean values are plotted, which are averaged over 20 runs for Func-2C, Func-3C, and Ackley-5C, and over 10 runs for XG-MNIST and NAS-CIFAR10
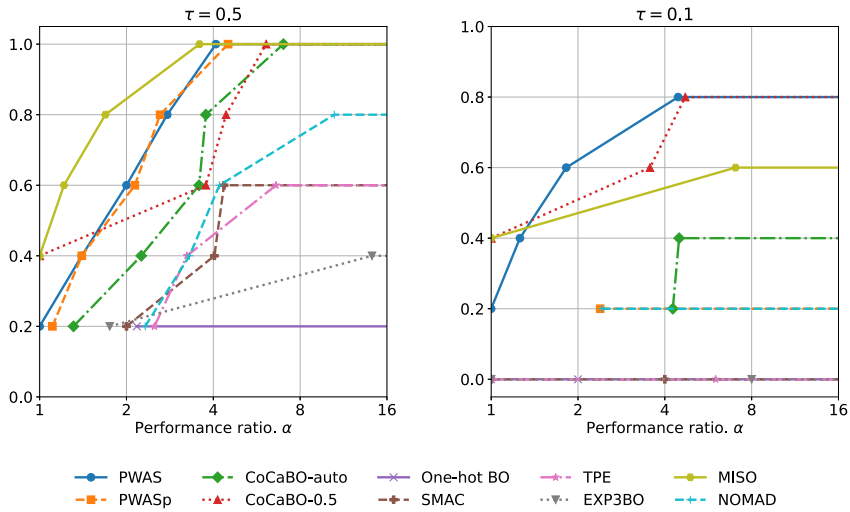
Fig. 8 Performance profiles of tested algorithms across varying performance ratios (α) with convergence tolerance τ equals to 0.5 and 0.1. Performance ratio (x-axis) is in logarithmic scale with base 2
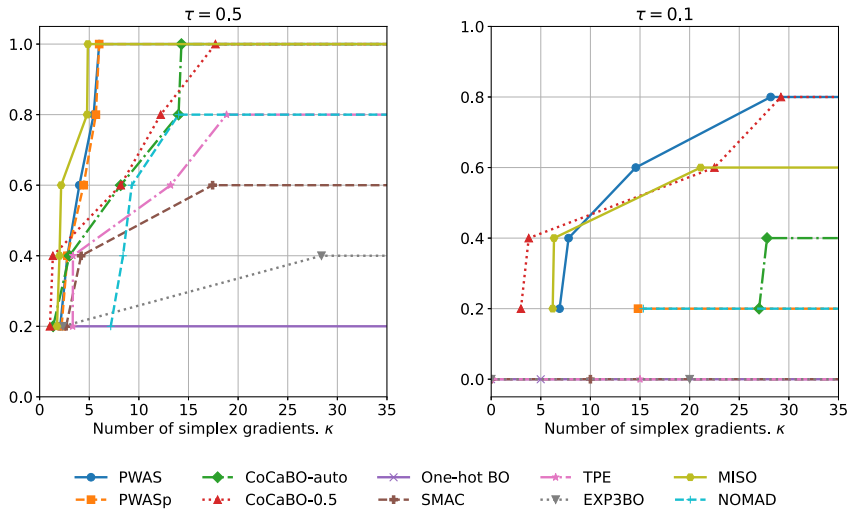

Fig. 9 Data profiles of tested algorithms as a function of the number of equivalent simplex gradients

where rapid, early-stage reduction in function value is critical. However, for $\tau = 0.1$, the performance of PWASp quickly drops. It is also observed that PWAS performs consistently better than PWASp. It is because that PWAS has access to function evaluations, while PWASp only receives pairwise comparisons, making it struggle to obtain highly accurate results. Nonetheless, in spite of the more limited information it gets, PWASp outperforms several other solvers (One-hot BO, SMAC, TPE, and EXP3BO) in most of the tested benchmarks. We also stress that PWASp is advantageous when the objective function is not easily quantifiable, but pairwise comparisons can be made.

Table 8 Performance of PWAS/PWASp on constrained mixed-variable synthetic problems

Algorithm	Horst6-hs044-modified		ros-cam-modified	
	Mean	Std	Mean	Std
PWAS	−62.579	3.5275e−08	−1.1151	0.3167
PWASp	−56.5539	8.3454	0.90540	1.7200
Global optimum	−62.579		−1.81	

For both PWAS and PWASp, the optimum is obtained after 100 black-box function evaluations
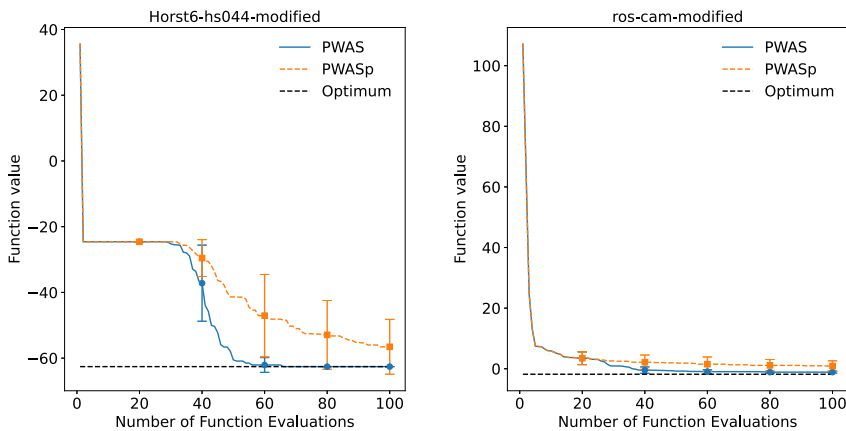Statistics are obtained with 20 random repetitions

Fig. 10 Convergence graphs for the constrained benchmark problems. Error bars indicate standard deviations, which are obtained with 20 random repetitions

On the other hand, When τ decreases, the performances of PWAS and MISO remain in the top three, with the performance of MISO drops slightly. CoCaBO-0.5 also holds a competitive position but is not as dominant as PWAS in early performance ratio (up to 2). However, as the performance ratio increases, the performance of CoCaBO-0.5 remains steady. Additionally, CoCaBO-0.5 showed superior performance in both real-world problems (XG-MNIST and NAS-CIFAR10). Although, it is worth noting that the performance differences are rather small in scale.

The optimization results for the constrained benchmarks are shown in Table 8 with the convergence graphs shown in Fig. 10. We observe that both PWAS and PWASp can quickly reduce the objective function values after a small number of black-box function evaluations, demonstrating their ability to handle mixed-variable linear QUAK constraints Also, PWAS achieves better results than PWASp regarding the best values obtained after the maximum allowed black-box function evaluations and consistency over multiple repetitions.

To show the efficiency of PWAS and PWASp, the average CPU time spent by PWAS and PWASp to fit the surrogate and solve the acquisition problem to suggest the next sample to query is reported in Table 9 for each tested benchmark problem. Considering that often evaluating the black-box function f or comparing samples

Table 9 CPU time (s) for surrogate fitting and acquisition optimization, averaged over $N_{max} - N_{init}$ active sampling steps

	Func-2C	Func-3C	Ackley-5C	XG-MNIST	NAS-CIFAR10	Horst6-hs044-modified	ros-cam-modified
Surrogate fitting							
PWAS	0.565	0.556	0.889	0.327	0.329	0.211	0.198
PWASp	0.221	0.289	0.544	0.312	0.422	0.177	0.162
Acquisition optimization							
PWAS	0.231	0.196	1.250	0.505	1.871	0.327	0.311
PWASp	0.270	0.420	1.352	0.589	1.700	0.387	0.364

involves expensive-to-evaluate simulations or experiments, such a CPU time can be considered negligible in real-life applications.

6 Conclusion

The algorithms PWAS and PWASp introduced in this paper can handle global and preference-based optimization problems involving mixed variables subject to linear preference-based optimization problems involving mixed variables subject to linear quantifiable unrelaxable a priori known (QUAK) constraints. Tests on different synthetic and real-world benchmark problems show that PWAS and PWASp can obtain better or comparable performance than other existing methods. In addition, the proposed acquisition strategies in PWAS and PWASp does not require uncertainty measurements, which can be extended to other simpler surrogate models such as polynomials. Although convergence to global optimizers cannot be guaranteed, we observed that PWAS and PWASp could quickly reduce (increase for maximization problems) the objective function values within a limited number of black-box function evaluations, despite the presence of integer and categorical variables and mixed-integer linear constraints. Therefore, both PWAS and PWASp can be considered as good heuristic algorithms for mixed-variable black-box optimization problems.

Future research will be devoted to extend this approach to handle mixed-variable problems under mixed-variable nonlinear constraints. Several approaches can be used, such as replacing them by piecewise affine approximations. Additionally, it can be interesting to integrate the proposed exploration function with other surrogate models (e.g., tree-based BO and polynominals).

Appendix A: Illustrative Example - Surrogate Fitting

To illustrate the remark noted at the end of Sect. 3.2, we optimize the following PWA function from [9],

$$f(x) = \max \left\{ \begin{bmatrix} 0.8031 \\ 0.0219 \\ -0.3227 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.2458 \\ -0.5823 \\ -0.1997 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.0942 \\ -0.5617 \\ -0.1622 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.9462 \\ -0.7299 \\ -0.7141 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, $$
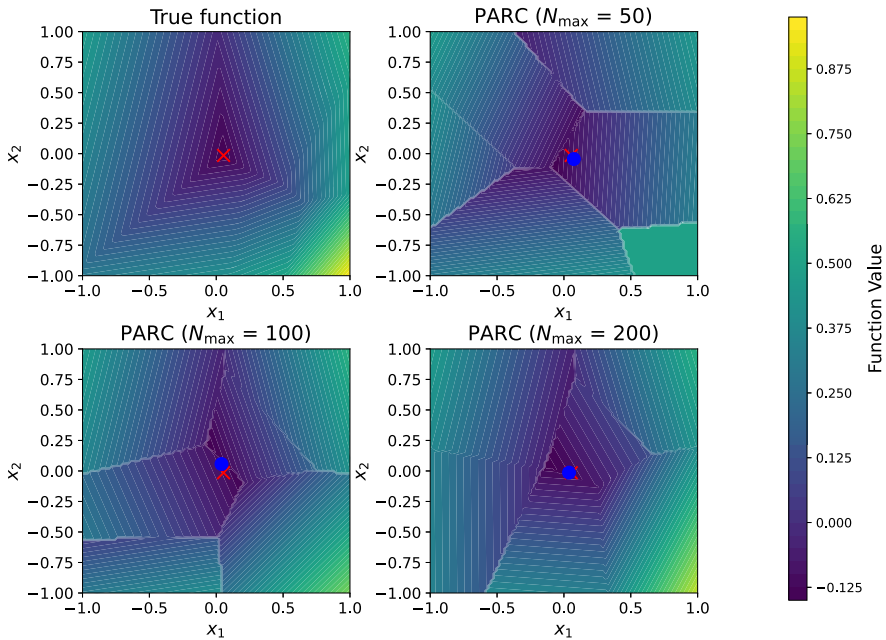
Fig. 11 Contour plots of Function (18), with the ground truth (top-left) and approximations generated by PWA surrogates with different numbers of samples (N_{max}). The approximations are based on 50, 100, and 200 samples. Red cross: optimum; blue dot: best sample obtained within N_{max} function evaluations

$$\left\{ \begin{bmatrix} -0.4799 \\ 0.1084 \\ -0.1210 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.5770 \\ 0.1574 \\ -0.1788 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \right\}. \tag{18}$$

We employ PWAS with varying maximum numbers of function evaluations N_{max}, using the samples collected during optimization to fit (18). As shown in Fig. 11, after 50 function evaluations, the surrogate effectively identifies the optimal region (dark purple region in Fig. 11), and after 200 evaluations, it closely approximates the optimal region. Furthermore, Fig. 12 demonstrates that as N_{max} increases from 50 to 200, the surrogate's predictions align more closely with the true values across the test dataset. With a smaller N_{max}, predictions are most accurate near the optimum; as N_{max} grows, the surrogate's accuracy improves also in regions further from the optimum. This indicates that PARC can be effectively used for our purpose as noted in the remark. Note: default solver settings of PWAS are used when collecting fitting samples for the illustrative example.

Appendix B: Benchmark

Note: the unconstrained mixed-variable synthetic and real-world problems are adopted from [51] for our comparisons. Note that the objective function is maximized in [51], so we consider the minimization of $-f(X)$ in PWAS and PWASp.
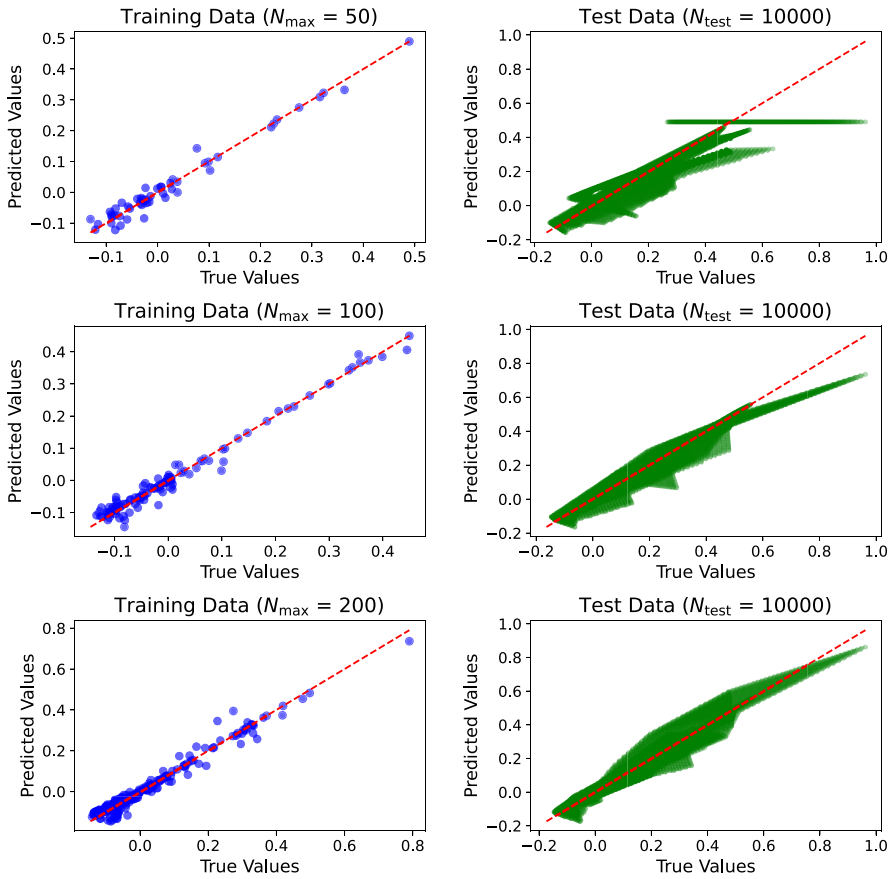
Fig. 12 Comparisons between the true function values of (18) and the predicted function values using the PWA surrogates for both training (with different values of N_{max}) and testing samples (with 10,000 samples). Blue dots: training samples collected during optimization; green dots: testing samples; red dashed lines: ideal case where true values equal to the predicted values

B.1 Unconstrained Mixed-variable Synthetic Benchmarks

Func-2C [7, 17, 43, 44, 51, 53]: $n_c = 2$, $n_{int} = 0$, and $n_d = 2$ with $n_i = 3$ for each categorical variable (denoted as n_{di}, for $i = 1, 2$). Each categorical variable is in $\{0, 1, 2\}$. The bounds are $\ell_x = [-1.0\ -1.0]^T$, $u_x = [1.0\ 1.0]^T$. The global maximum $f(X) = 0.20632$ is attained at $X = [0.0898\ -0.7126\ 1\ 1]^T$ and $[-0.0898\ 0.7126\ 1\ 1]^T$.

$$f(X) = \begin{cases} f_1 + f_{ros}(x) & n_{d2} = 0 \\ f_1 + f_{cam}(x) & n_{d2} = 1 \\ f_1 + f_{bea}(x) & n_{d2} = 2, \end{cases}$$

$$\text{where} \quad f_1(x, y) = \begin{cases} f_{\text{ros}}(x) & n_{d1} = 0 \\ f_{\text{cam}}(x) & n_{d1} = 1 \\ f_{\text{bea}}(x) & n_{d1} = 2 \end{cases}$$

$$f_{\text{ros}}(x) = -(100(x_2 - x_1^2)^2 + (x_1 - 1)^2)/300$$

$$f_{\text{cam}}(x) = -(a_1 + a_2 + a_3)/10$$

$$a_1 = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2$$

$$a_2 = x_1 x_2$$

$$a_3 = (-4 + 4x_2^2)x_2^2$$

$$f_{\text{bea}}(x) = -((1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2$$
$$+ (2.625 - x_1 + x_1 x_2^3)^2)/50 \tag{19}$$

Func-3C [7, 17, 43, 44, 51, 53]: $n_c = 2$, $n_{\text{int}} = 0$, and $n_d = 3$ with $n_i = 3$ for each categorical variable (denoted as n_{di}, for $i = 1, 2, 3$). Each categorical variable is in $\{0, 1, 2\}$. The bounds are $\ell_x = [-1.0 \ -1.0]^\mathsf{T}$, $u_x = [1.0 \ 1.0]^\mathsf{T}$. The global maximum $f(X) = 0.72214$ is attained at $X = [0.0898 \ -0.7126 \ 1 \ 1 \ 0]^\mathsf{T}$ and $[-0.0898 \ 0.7126 \ 1 \ 1 \ 0]^\mathsf{T}$.

$$f(X) = \begin{cases} f_2 + 5f_{\text{cam}}(x) & n_{d3} = 0 \\ f_2 + 2f_{\text{ros}}(x) & n_{d3} = 1 \\ f_2 + n_{d2}f_{\text{bea}}(x) & n_{d3} = 2, \end{cases}$$

$$\text{where} \quad f_2(x, y) = \begin{cases} f_1 + f_{\text{ros}}(x) & n_{d21} = 0 \\ f_1 + f_{\text{cam}}(x) & n_{d2} = 1 \\ f_1 + f_{\text{bea}}(x) & n_{d2} = 2 \end{cases} \tag{20}$$

$$f_1(x, y) = \begin{cases} f_{\text{ros}}(x) & n_{d1} = 0 \\ f_{\text{cam}}(x) & n_{d1} = 1 \\ f_{\text{bea}}(x) & n_{d1} = 2 \end{cases}$$

$f_{\text{ros}}(x)$, $f_{\text{cam}}(x)$, and $f_{\text{bea}}(x)$ are defined in (19)

Ackley-5C [1, 51, 53]: $n_c = 1$, $n_{\text{int}} = 0$, and $n_d = 5$ with $n_i = 17$ for each category (denoted as n_{di}, for $i = 1, \ldots, 5$). Each categorical variableis in $\{0, 1, \ldots, 16\}$. The bounds are $\ell_x = -1.0$, $u_x = 1.0$. The global maximum $f(X) = 0$ is attained at

$X = [0\ 8\ 8\ 8\ 8\ 8]^\mathsf{T}$.

$$f(X) = a \exp\left(-b\left(\frac{s_1}{n}\right)^{\frac{1}{2}}\right) + \exp\left(\frac{s_2}{n}\right) - a - \exp(1),$$

where $a = 20, b = 0.2, c = 2\pi$

$$s_1 = x^2 + \sum_{i=1}^{5} z_i^2(n_{di}) \tag{21}$$

$$s_2 = \cos(cx) + \sum_{i=1}^{5} \cos(cz_i(n_{di}))$$

$$z_i(n_{di}) = -1 + 0.125 n_{di}$$

B.2 Unconstrained Mixed-variable Real-world Benchmarks

XG-MNIST [14, 35, 51]: $n_c = 4$, $n_{int} = 1$, and $n_d = 3$ with $n_i = 2$, for $i = 1, 2, 3$. Each categorical variable (n_{di}) can be either 0 or 1. The bounds are $\ell_x = [10^{-6}, 10^{-6}\ 0.001\ 10^{-6}]^\mathsf{T}$, $u_x = [1\ 10\ 1\ 5]^\mathsf{T}$; $\ell_y = 1$, $u_y = 10$.
Notes on the optimization variables:
The 0.7/0.3 stratified train/test split ratio is applied as noted in [51]. The *xgboost* package is used [14] on MNIST classification [35]. The optimization variables in this problem are the parameters of the *xgboost* algorithm. Specifically, the continuous variables x_1, x_2, x_3, and x_4 refer to the following parameters in *xgboost*, respectively: 'learning_rate', 'min_split_loss', 'subsample', and 'reg_lambda'. The integer variable y stands for the 'max_depth'. As for the categorical variables, n_{d1} indicates the booster type in *xgboost*, where $n_{d1} = \{0, 1\}$ corresponding to {'gbtree', 'dart'}. n_{d2} represents the 'grow_policy', where $n_{d2} = \{0, 1\}$ corresponding to {'depthwise', 'lossguide'}. n_{d3} refers to the 'objective', where $n_{d3} = \{0, 1\}$ corresponding to {'multi:softmax', 'multi:softprob'}.
Notes on the objective function:
The classification accuracy on test data is used as the objective function.
NAS-CIFAR10 [31, 51, 60]: $n_c = 21$, $n_{int} = 1$, and $n_d = 5$ with $n_i = 3$ for each category (denoted as n_{di} for $i = 1, \ldots, 5$). Each categorical variable is in $\{0, 1, 2\}$. The bounds are $\ell_x^i = 0$, $u_x^i = 1$, $\forall i = 1, \ldots, n_c$; $\ell_y = 0$, $u_y = 9$.
Notes on the optimization problem:
The public dataset NAS-Bench-101 [59] is used. This dataset maps convolutional neural network (CNN) architectures to their trained and evaluated performance on CIFAR-10 classification. As a result, we can quickly look up the validation accuracy of the proposed CNN architecture by PWAS/PWASp. The same encoding method for the CNN architecture topology as noted in [51, 59] is used.
Notes on the optimization variables:
The CNN architecture search space is described by a directed acyclic graph (DAG) which has 7 nodes with the first and the last nodes being the input and output nodes.

The continuous variables represent the probability values for the 21 possible edges in the DAG. The integer variable y is the number of edges present in the DAG. The categorical variables are the operations for the 5 intermediate nodes in the DAG, for which $n_{di} = \{0, 1, 2\}$ corresponding to {`3x3 conv`, `1x1 conv`, `3x3 max-pool`}. Within the 21 possible edges, only y edges with the highest probability are activated. The DAG that leads to invalid CNN architecture topology specifications will result in zero validation accuracy.

Notes on the objective function:

The validation accuracy of the defined CNN architecture topology on CIFAR-10 classification is used as the objective function.

B.3 Constrained Mixed-variable Synthetic Problems

Horst6-hs044-modified [26, 34]: $n_c = 3$, $n_{\text{int}} = 4$, and $n_d = 2$ with $n_1 = 3$ and $n_2 = 2$ (the first (n_{d1}) and the second (n_{d2}) categorical variable are in $\{0, 1, 2\}$ and $\{0, 1\}$, respectively. $\ell_x = [0\ 0\ 0]^T$, $u_x = [6\ 6\ 3]^T$; $\ell_y = [0\ 0\ 0\ 0]^T$, $u_y = [3\ 10\ 3\ 10]^T$. The global minimum $f(X) = -62.579$ is attained at $X = [5.21066\ 5.0279\ 0\ 0\ 3\ 0\ 4\ 2\ 1]^T$.

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

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

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

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

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

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

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

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

$$B_{\text{ineq}} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$b_{\text{ineq}} = [2.86506, -1.49161, 0.51959, 1.58409, 2.19804, -1.30185,$$
$$- 0.73829, 8, 12, 12, 8, 8, 5]^{\mathsf{T}} \tag{22}$$

ros-cam-modified [8, 28]: $n_c = 2$, $n_{\text{int}} = 1$, and $n_d = 2$ with $n_i = 2$ for each categorical variable (denoted as n_{di}, for $i = 1, 2$). Each categorical variable is in $\{0, 1\}$. $\ell_x = [-2.0 \; -2.0]^{\mathsf{T}}$, $u_x = [2.0 \; 2.0]^{\mathsf{T}}$; $\ell_y = 1$, $u_y = 10$. The global minimum $f(X) = -1.81$ is attained at $X = [0.0781 \; 0.6562 \; 5 \; 1 \; 1]^{\mathsf{T}}$.

$$f(X) = \begin{cases} f_1 + f_{\text{ros}}(x, y) & n_{d2} = 0 \\ f_1 + f_{\text{cam}}(x, y) & n_{d2} = 1 \end{cases}$$

$$\text{s.t} \quad A_{\text{ineq}} x \leq b_{\text{ineq}},$$

$$\text{where} 1(x, y) = \begin{cases} f_{\text{ros}}(x, y) & n_{d1} = 0 \\ f_{\text{cam}}(x, y) & n_{d1} = 1 \end{cases}$$

$$f_{\text{ros}}(x, y) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2 + (y - 3)^2$$

$$f_{\text{cam}}(x, y) = a_1 + a_2 + a_3 + (y - 5)^2$$

$$a_1 = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2$$

$$a_2 = x_1 x_2$$

$$a_3 = (-4 + 4x_2^2)x_2^2$$

$$A_{\text{ineq}} = \begin{bmatrix} 1.6295 & 1 \\ 0.5 & 3.875 \\ -4.3023 & -4 \\ -2 & 1 \\ 0.5 & -1 \end{bmatrix}$$

$$b_{\text{ineq}} = [3.0786, 3.324, -1.4909, 0.5, 0.5]^{\mathsf{T}} \tag{23}$$

Appendix C: Algorithms

C.1 CoCaBO

Continuous and Categorical Bayesian Optimization (CoCaBO) [51] is an algorithm proposed to optimize box-constrained expensive black-box problems with both continuous and categorical variables, specifically for problems with *multiple* categorical variables with *multiple* possible values. CoCaBO model the input space with a Gaus-

sian Process kernel, which is designed to allow information sharing across different categorical variables to enhance data efficiency.

C.2 EXP3BO

EXP3BO [20] is an algorithm proposed to optimize box-constrained, expensive black-box problems, which is modified from BO via the EXP3 algorithm [6, 52]. It can deal with mixed categorical and continuous input spaces by utilizing multi-armed bandits. Specifically, EXP3BO constructs a Gaussian Process surrogate specific to each chosen category, making it unsuitable for handling problems with multiple categorical classes.

C.3 One-hot BO

One-hot BO [21] handles the input space with continuous and categorical variables by one-hot encoding the categorical variables and treating these encoded variables as continuous. Then, the standard BO procedure is used to solve the optimization problem on the transformed input space.

C.4 TPE

Tree-structured Parzen Estimator (TPE) [11] is a black-box optimization algorithm based on tree-structured Parzen density estimators. TPE uses nonparametric Parzen kernel density estimators to model the distribution of good and bad configurations w.r.t. a reference value. Due to the nature of kernel density estimators, TPE also supports continuous and categorical spaces.

C.5 MISO

Mixed-Integer Surrogate Optimization (MISO) [47] is an algorithm that targets expensive black-box functions with mixed-integer variables. It constructs a surrogate model using the radial basis function to approximate the unknown function. MISO follows a general procedure of surrogate-based optimization methods. Additionally, MISO combines different sampling strategies (e.g., coordinate perturbation, random sampling, expected improvement, target value, and surface minimum) and local search to obtain high-accuracy solutions.

C.6 NOMAD

NOMAD [4, 5] is a C++ implementation of the Mesh Adaptive Direct search (MADS). It is designed to solve difficult black-box optimization problems. In particular, it can handle nonsmooth, nonlinearly constrained, single or bi-objective, and mixed variable optimization problems. It handles the categorical variable using the extended poll, which is defined as following [32]:

The extended poll first calls the user-provided procedure defining the neighborhood of categorical variables. The procedure returns a list of points that are neighbors of the current best point (incumbent) such that categorical variables are changed and the other variables may or may not be changed. These points are called the extended poll points and their dimension may be different than the current best point, for example when a categorical variable indicates the number of continuous variables.

C.7 SMAC

Sequential Model-based Algorithm Configuration (SMAC) [27] is a surrogate-based black-box optimization method originally proposed to tackle *algorithm configuration* problems with continuous and categorical variables. Its model is based on random forests [12], so it can handle categorical variables explicitly. SMAC uses empirical mean and variance within a tree ensemble to identify uncertain search space regions and optimizes the acquisition function by combing local and random search.

Data Availibility Statement The PWAS package and the tested benchmarks are available on the GitHub repository (https://GitHub.com/mjzhu-p/PWAS). The MNIST dataset analysed for benchmark XG-MNIST in this study is publicly available and retrieved via sklearn.datasets.load_digits. The NASBench dataset analysed for benchmark NAS-CIFAR10 is publicly available from the repository hosted on GitHub (https://GitHub.com/google-research/nasbench) under Apache License 2.0 and can be downloaded via https://storage.googleapis.com/nasbench/nasbench_only108.tfrecord. The MISO package is publicly available from the repository hosted on GitHub (https://github.com/Julie2901/miso). The NOMAD4 package is publicly available from the repository hosted on GitHub (https://github.com/bbopt/nomad) under LGPL-3.0 license.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose

References

1. Ackley, D.H.: The Model, pp. 29–70. Springer US (1987). https://doi.org/10.1007/978-1-4613-1997-9_2
2. Audet, C., Hallé-Hannan, E., Le Digabel, S.: A general mathematical framework for constrained mixed-variable blackbox optimization problems with meta and categorical variables. Oper. Res. Forum **4**(12) (2023). https://doi.org/10.1007/s43069-022-00180-6
3. Audet, C., Hare, W.: Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-68913-5
4. Audet, C., Le Digabel, S., Rochon Montplaisir, V., Tribes, C.: The NOMAD project. Software available at https://www.gerad.ca/nomad
5. Audet, C., Le Digabel, S., Rochon Montplaisir, V., Tribes, C.: Algorithm 1027: NOMAD version 4: nonlinear optimization with the MADS algorithm. ACM Trans. Math. Software **48**(3), 35:1–35:22 (2022). https://doi.org/10.1145/3544489

6. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multiarmed bandit problem. SIAM J. Comput. **32**(1), 48–77 (2002). https://doi.org/10.1137/S0097539701398375

7. Beale, E.M.L.: On an iterative method for finding a local minimum of a function of more than one variable. Tech. Rep. 25, Statistical Techniques Research Group, Section of Mathematical Statistics, Department of Mathematics, Princeton University (1958)

8. Bemporad, A.: Global optimization via inverse distance weighting and radial basis functions. Comput. Optim. Appl. **77**, 571–595 (2020). https://doi.org/10.1007/s10589-020-00215-w

9. Bemporad, A.: A piecewise linear regression and classification algorithm with application to learning and model predictive control of hybrid systems. IEEE Trans. Autom. Control. **68**(6), 3194–3209 (2022). https://doi.org/10.1109/tac.2022.3183036

10. Bemporad, A., Piga, D.: Global optimization based on active preference learning with radial basis functions. Mach. Learn. **110**, 417–448 (2021). https://doi.org/10.1007/s10994-020-05935-y

11. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger (eds.) Advances in Neural Information Processing Systems, vol. 24. Curran Associates, Inc. (2011)

12. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001). https://doi.org/10.1023/A:1010933404324

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

14. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016). https://doi.org/10.1145/2939672.2939785

15. Costa, A., Nannicini, G.: RBFopt: an open-source library for black-box optimization with costly function evaluations. Math. Program. Comput. **10**, 597–629 (2018). https://doi.org/10.1007/s12532-018-0144-7

16. Cox, D.R.: Some procedures connccted with the logistic qualitative response curve. In: David, F.N. (ed.) Research Papers in Statistics: Essays in Honour of J. Neyman's 70th Birthday. Wiley (1966)

17. Dixon, L.C.W., Szegö, G.P.: The global optimization problem: an introduction. In: Dixon, L.C.W., Szegö, G.P. (eds.) Towards Global Optimiation 2, pp. 1–15. North-Holland, Amsterdam, Netherlands (1978)

18. Forgione, M., Piga, D., Bemporad, A.: Efficient calibration of embedded mpc. IFAC-PapersOnLine **53**(2), 5189–5194 (2020). https://doi.org/10.1016/j.ifacol.2020.12.1188

19. Getdata Graph Digitizer: Getdata graph digitizer: Version 2.26 (2013). http://getdata-graph-digitizer.com/

20. Gopakumar, S., Gupta, S., Rana, S., Nguyen, V., Venkatesh, S.: Algorithmic assurance: An active approach to algorithmic testing using bayesian optimisation. In: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (eds.) Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc. (2018)

21. GPyOpt: Gpyopt: a bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt (2016)

22. Greenhill, S., Rana, S., Gupta, S., Vellanki, P., Venkatesh, S.: Bayesian optimization for adaptive experimental design: a review. IEEE Access **8**, 13937–13948 (2020). https://doi.org/10.1109/ACCESS.2020.2966228

23. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023). https://www.gurobi.com

24. Holmström, K.: An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization. J. Global Optim. **41**, 447–464 (2008). https://doi.org/10.1007/s10898-007-9256-8

25. Hooker, J., Osorio, M.: Mixed logical-linear programming. Discrete Appl. Math. **96–97**, 395–442 (1999). https://doi.org/10.1016/S0166-218X(99)00100-6

26. Horst, R., Pardalos, P.M., Van Thoai, N.: Introduction to Global Optimization. Springer Series in Nonconvex Optimization and Its Applications, vol. 3. Springer (2000)

27. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) Learning and Intelligent Optimization, pp. 507–523. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40

28. Jamil, M., Yang, X.S.: A literature survey of benchmark functions for global optimisation problems. Int. J. Math. Model. Numer. Optim. **4**(2), 150–194 (2013). https://doi.org/10.1504/IJMMNO.2013.055204

29. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Global Optim. **13**(4), 455 (1998). https://doi.org/10.1023/A:1008306431147

30. Kim, S.H., Boukouvala, F.: Surrogate-based optimization for mixed-integer nonlinear problems. Comput. Aided Chem. Eng. **140**, 106,847 (2020). https://doi.org/10.1016/j.compchemeng.2020.106847

31. Klein, A., Hutter, F.: Tabular benchmarks for joint architecture and hyperparameter optimization. arXiv (2019). arXiv:1905.04970

32. Le Digabel, S., Tribes, C., Montplaisir, V.R., Audet, C.: Nomad user guide version 3.9.1 (2019). https://www.gerad.ca/software/nomad/Downloads/user_guide.pdf

33. Le Digabel, S., Wild, S.M.: A taxonomy of constraints in black-box simulation-based optimization. Optim. Eng. **25**(2), 1125–1143 (2024). https://doi.org/10.1007/s11081-023-09839-3

34. Le Thi, H.A., Vaz, A.I.F., Vicente, L.: Optimizing radial basis functions by DC programming and its use in direct search for global derivative-free optimization. Trans. Oper. Res. **20**, 190–214 (2012). https://doi.org/10.1007/s11750-011-0193-9

35. LeCun, Y., Cortes, C., Burges, C.: MNIST handwritten digit database (2010). http://yann.lecun.com/exdb/mnist

36. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inform. Theory **28**(2), 129–137 (1982). https://doi.org/10.1109/tit.1982.1056489

37. Luo, G.: A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Netw. Model. Anal. Health Inform. Bioinform. **5**, 1–16 (2016). https://doi.org/10.1007/s13721-016-0125-6

38. Makhorin, A.: GNU linear programming kit. reference manual. version 5.0 (2020). https://www.chiark.greenend.org.uk/doc/glpk-doc/glpk.pdf

39. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979). https://doi.org/10.1080/00401706.2000.10485979

40. Mistry, M., Letsios, D., Krennrich, G., Lee, R.M., Misener, R.: Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded. INFORMS J. Comput. **33**(3), 1103–1119 (2021). https://doi.org/10.1287/ijoc.2020.0993

41. Mitchell, S., OSullivan, M., Dunning, I.: PuLP: a linear programming toolkit for python (2011). https://optimization-online.org/wp-content/uploads/2011/09/3178.pdf

42. Močkus, J.: On bayesian methods for seeking the extremum. In: G.I. Marchuk (ed.) Optimization Techniques IFIP Technical Conference, pp. 400–404. Springer (1975). https://doi.org/10.1007/3-540-07165-2_55

43. Molga, M., Smutnicki, C.: Test functions for optimization needs (2005). Accessed 1 Feb 2023, from https://robertmarks.org/Classes/ENGR5358/Papers/functions.pdf

44. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981). https://doi.org/10.1145/355934.355936

45. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM J. Optim. **20**(1), 172–191 (2009). https://doi.org/10.1137/080724083

46. Motzkin, T.S., Raiffa, H., Thompson, G.L., Thrall, R.M.: The Double Description Method **2**, 51–73 (1953). https://doi.org/10.1515/9781400881970-004

47. Müller, J.: MISO: mixed-integer surrogate optimization framework. Optim. Eng. **17**, 177–203 (2016). https://doi.org/10.1007/s11081-015-9281-2

48. Nguyen, V.: Bayesian optimization for accelerating hyper-parameter tuning. In: IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering, pp. 302–305. IEEE (2019). https://doi.org/10.1109/AIKE.2019.00060

49. Papalexopoulos, T.P., Tjandraatmadja, C., Anderson, R., Vielma, J.P., Belanger, D.: Constrained discrete black-box optimization using mixed-integer programming. In: K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, S. Sabato (eds.) Proceedings of the 39th International Conference on Machine Learning, vol. 162, pp. 17,295–17,322. PMLR (2022)

50. Ploskas, N., Sahinidis, N.V.: Review and comparison of algorithms and software for mixed-integer derivative-free optimization. J. Global Optim. pp. 1–30 (2022). https://doi.org/10.1007/s10898-021-01085-0

51. Ru, B., Alvi, A.S., Nguyen, V., Osborne, M.A., Roberts, S.J.: Bayesian optimisation over multiple continuous and categorical inputs. In: H. Daumé III, A. Singh (eds.) Proceedings of the 37th International Conference on Machine Learning. JMLR (2020)

52. Seldin, Y., Szepesvári, C., Auer, P., Abbasi-Yadkori, Y.: Evaluation and analysis of the performance of the EXP3 algorithm in stochastic environments. In: M.P. Deisenroth, C. Szepesvári, J. Peters (eds.) Proceedings of the 10th European Workshop on Reinforcement Learning, vol. 24, pp. 103–116 (2013)

53. Surjanovic, S., Bingham, D.: Virtual library of simulation experiments: test functions and datasets. Accessed 01 Feb 2023, from http://www.sfu.ca/~ssurjano

54. Swiler, L.P., Hough, P.D., Qian, P., Xu, X., Storlie, C., Lee, H.: Surrogate Models for Mixed Discrete-Continuous Variables, pp. 181–202. Springer (2014). https://doi.org/10.1007/978-3-319-04280-0_21

55. Theil, H.: A multinomial extension of the linear logit model. Int. Econ. Rev. **10**(3), 251–259 (1969). https://doi.org/10.2307/2525642

56. Torrisi, F., Bemporad, A.: HYSDEL: a tool for generating computational hybrid models. IEEE Trans. Control Syst. Technol. **12**(2), 235–249 (2004). https://doi.org/10.1109/TCST.2004.824309

57. Williams, H.: Model Building in Mathematical Programming, 5th edn. Wiley, West Sussex (2013)

58. Wu, J., Chen, X.Y., Zhang, H., Xiong, L.D., Lei, H., Deng, S.H.: Hyperparameter optimization for machine learning models based on Bayesian optimization. J. Electron. Sci. Technol. **17**(1), 26–40 (2019). https://doi.org/10.11989/JEST.1674-862X.80904120

59. Yang, K.K., Wu, Z., Arnold, F.H.: Machine-learning-guided directed evolution for protein engineering. Nat. Methods **16**(8), 687–694 (2019). https://doi.org/10.1038/s41592-019-0496-6

60. Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F.: NAS-Bench-101: Towards reproducible neural architecture search. In: K. Chaudhuri, R. Salakhutdinov (eds.) Proceedings of the 36th International Conference on Machine Learning, **97**, pp. 7105–7114. PMLR (2019)