Hybrid Parameter Search and Dynamic Model Selection for Mixed-Variable Bayesian Optimization

Hengrui Luo (hrluo@lbl.gov),[*]
Younghyun Cho (younghyun@berkeley.edu)[†],
James W. Demmel (demmel@berkeley.edu) [†],
Xiaoye S. Li (xsli@lbl.gov)[*],
Yang Liu (liuyangzhuan@lbl.gov)[*]

Abstract

This paper presents a new type of hybrid model for Bayesian optimization (BO) adept at managing mixed variables, encompassing both quantitative (continuous and integer) and qualitative (categorical) types. Our proposed new hybrid models (named hybridM) merge the Monte Carlo Tree Search structure (MCTS) for categorical variables with Gaussian Processes (GP) for continuous ones. hybridM leverages the upper confidence bound tree search (UCTS) for MCTS strategy, showcasing the tree architecture's integration into Bayesian optimization. Our innovations, including dynamic online kernel selection in the surrogate modeling phase and a unique UCTS search strategy, position our hybrid models as an advancement in mixed-variable surrogate models. Numerical experiments underscore the superiority of hybrid models, highlighting their potential in Bayesian optimization.

Keywords: Gaussian processes, Monte Carlo tree search, categorical variables, online kernel selection.

1 Introduction

Our motivating problem is to optimize a "black-box" function with "mixed" variables, without an analytic expression. "Mixed" signifies that the function's input variables comprise

[*]Lawrence Berkeley National Laboratory, Berkeley, CA, 94701.
[†]University of California, Berkeley, Berkeley, CA, 94720.

Type	**Continuous**	**Integer**	**Categorical**
Representation	$\mathbb{R}$	$\mathbb{Z}$	finite set
Magnitude	Yes	Yes	No
Order	Yes	Yes	No
Encoding	No	Yes	Yes
Gradient	continuous gradient	discrete gradient	No

Table 1: Comparison among different types of variables. The discussion of different types of encodings can be found in Cerda et al. (2018).

both continuous (quantitative) and categorical (qualitative) variables, common in machine learning and scientific computing tasks such as the performance tuning of mathematical libraries and application codes at runtime and compile-time (Balaprakash et al., 2018).

Bayesian optimization (BO) with Gaussian process (GP) surrogate models is a common method for optimizing noisy and expensive black-box functions, designed primarily for continuous-variable functions (Shahriari et al., 2016; Sid-Lakhdar et al., 2020). In BO we draw (possibly noisy) sequential samples from black-box functions at any query points. Then we use a GP surrogate model fitted with sequential samples $(\boldsymbol{x}, y)$ chosen by optimizing the criteria based on surrogate. Since optimizing criteria (e.g., expected improvement by Jones et al. (1998)) is defined on a continuous-variable domain, extending BO to mixed-variable functions presents challenges due to variable type differences (Table 1).

Continuous variables have uncountably many values with magnitudes and intrinsic ordering, allowing natural gradient definition. In contrast, categorical variables, having finitely many values without intrinsic ordering or magnitude, require encoding in the GP context, potentially inducing discontinuity and degrading GP performance (Luo et al., 2021). In addition to continuous and categorical variables, *integer variables* (a.k.a. ordinal variables with intrinsic ordering) have discrete values, natural ordering and discrete gradients. The empirical rule of thumb for handling an integer variable (Karlsson et al., 2020) is to treat it as a categorical variable if the number of integer values (i.e., number of categorical values) is small, or as a continuous variable with embedding (a.k.a. rounding-off encoding) otherwise. We follow this empirical rule for integer variables, and treat mixed-variable problems which contains all these kinds of variables.

Existing strategies for addressing challenges in mixed-variable Bayesian Optimization (BO) can be categorized into four non-exclusive classes.

First, encoding transforms categorical variables into numerical format, enabling algorithms to process and analyze non-numeric data effectively. Encoding methods such as one-hot encoding (representing categories with binary numbers (Snoek et al., 2012)) and graph-based encoding (using graph vertices to represent categories (Karlsson et al., 2020)), are widely used for mixed variables. However, their theoretical justification and practical performance are highly problem-specific (Cerda et al., 2018; Garrido-Merchán and Hernández-Lobato, 2020), motivating more advanced mixed-variable techniques.

Second, new acquisition functions have been designed to address specific problems in discrete or categorical space (Deshwal et al., 2021; Oh et al., 2021; Willemsen et al., 2021). These advanced designs preserve the BO framework and the Gaussian Process (GP) surrogate. But these new acquisition functions are often difficult to optimize and require special implementations that may introduces further optimization challenges in the inner loop.

Third, modification of the surrogate has led to new GP models such as additive GP models (Deng et al., 2017; Xiao et al., 2021), latent variable GP (Zhang et al., 020a,b), models based on non-GP types of basis functions (Bliek et al., 2021), graphical models (Biau and Cadre, 2021; Head et al., 2020; Olson and Moore, 2019), and models with novel kernels capturing the categorical variables using a density estimator (TPE, (Bergstra et al., 2011)) or random forests (SMAC, (Hutter et al., 2011)). These novel surrogate models are kind of generic modeling strategies that can be used for mixed-variable input space, however, they usually face challenges in higher-dimensional problems with a lot of categories.

Fourth, bandit-based search methods utilize either multiarmed bandits (MAB) (EXP3BO (Gopakumar et al., 2018), Bandit-BO (Nguyen et al., 2019), CoCaBO (Ru et al., 2020)), or Monte Carlo tree search (MCTS) (MOSAIC, (Rakotoarison et al., 2019)) to select the optimal categorical variables. The model then fits the remaining continuous variables using a standard GP surrogate that includes all of these variables. However, the regular bandit (EXP3BO, BanditBO, CoCaBO) needs to construct all possible combinations of categories and cannot handle large number of categories. On the other hand, the tree-like bandit (MOSAIC) is more scalable for large number of combinations. However, MOSAIC requires independent GPs and a large budget per GP to behave reasonably. For problems with many categories, this approach can also be inefficient and overlooks interactions between continuous and categorical variables.

Our proposed *hybrid model* (henceforth dubbed hybridM) uses tree structure for categorical space and GP for continuous space. hybridM presents a "tree-and-GP" architecture for search phase and utilizes a novel *dependent* Gaussian Process (GP) surrogate with a selection of mixed-variable kernels to capture correlations between continuous and categorical variables (See Figure 1) for modeling phase. In addition, our unique approach integrates a dynamic kernel selection strategy, balancing optimism (Luo and Zhu, 2023; Ye, 1998) and likelihood. This method enhances the efficiency of model selection and optimization, effectively addressing the limitations of the MOSAIC large tuning sample budget requirement.

This approach is our first attempt to the open problem of conducting model selection

[1]This includes the implementation for roundrobin MAB and random MAB.

	Categorical	Continuous
Hybrid models		
hybridM (proposed)	MCTS	dependent GPs
MOSAIC (Rakotoarison et al., 2019)	MCTS	independent GPs
GP (skopt) (Head et al., 2020)	sampling/encoding	independent GPs
CoCaBO (Ru et al., 2020)	1-layer bandit	dependent GPs
EXP3BO[1] (Gopakumar et al., 2018)	1-layer bandit	independent GPs
Non-hybrid models		
SMAC (Hutter et al., 2011)	*	*
TPE (Bergstra et al., 2011)	tree estimator	GPs
forest (skopt) (Head et al., 2020)	tree estimator	tree estimator

Table 2: Comparison among different mixed-variable surrogate models. Dependent GPs refer to GPs with mixed variables, while independent GPs refer to GPs with only continuous variables.
*For SMAC, we use SMAC3 (1.2.0) and refer to their paper (Hutter et al., 2011) for the details of their complex model.

in an online context. The overall workflow of the proposed method is as follows. We first employ tree structures for heuristic search of the next sample's categorical part with the upper confidence bound tree search (UCTS) strategy (see Sections 2.1 and Appendix A). We then dynamically select the optimal kernel for the mixed-variable GP surrogate (see Section 2.2). Finally, we use acquisition maximization to search for the next sample's continuous variables using the best continuous GP kernel, conditioned on the categorical variables.

Our contributions are summarized below.

- We develop a novel hybrid model with a tree and dependent GP structure integrated with MCTS extending the state-of-the-art architectures in Table 2.
- We experiment with existing kernels for mixed-variable situations and introduce a family of candidate kernels for the mixed-type variable BO (Appendix B).
- We propose a numerically stable and efficient dynamic kernel selection criterion (Section 2.2.2) to enhance tuning performance.
- We demonstrate the superior tuning performance of our hybrid model through various synthetic benchmarks as well as real application codes, including tunings of a neural network and a parallel sparse solver STRUMPACK(Ghysels et al., 2017).

Tuning scientific applications like neural networks involves optimizing numerous different types of hyperparameter (Elsken et al., 2019), which affects performance outcomes.

For another example, optimizing STRUMPACK (Ghysels et al., 2017), a software package for solving large sparse linear systems, is rather expensive to tune due to the time cost per execution with certain configurations and its data-dependent nature. Both applications present unique challenges and require efficient tuning, emphasizing the need for effective auto-tuning strategies. Comprehensive benchmarks for mixed-variable methods are provided in Section 3, which showcasing the superiority of our hybrid models in diverse scenarios.

2 New Hybrid Model

This section describes our hybrid methodology, which is illustrated by Figure 1 and detailed later in Algorithm 1. The section is organized as follows: In Section 2.1, we first introduce the MCTS, which will be used to decide the categorical part of the next sequential sample when used with UCTS (alternatively, we provide Bayesian strategies in Appendix A). With GP surrogate and families of covariance kernels (detailed in Appendix B), which will be used to decide the continuous part of the next sequential sample, we search the categorical part via a tree strategy and the continuous part via a GP strategy. In Section 2.2, after deciding both the categorical and continuous part of the variable,we explain how to choose the mixed-variable kernel for the GP, which models the overall variable (including both categorical and continuous part via selected mixed kernels) and gives us a balance between exploration and exploitation. In Section 2.3, we summarize our hybridM algorithm when all components are combined.

2.1 UCTS Update Strategy

UCTS is one of the MCTS search strategies that constructs a tree structure as the Bayesian Optimization (BO) sampling proceeds. Each sequential sample is composed of categorical and continuous parts, denoted as $\boldsymbol{x} = (\boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{con}})$ and the totality of responses as $\boldsymbol{y}$.

In the MCTS methodology, the tree structure represents the categorical variables n_c of $\boldsymbol{x}_{\text{cat}}$. Nodes at the same level represent different choices of categorical values for the corresponding variable. If a tree has $L \leq n_c$ levels, then each level i has K_i nodes. The tree has a total of C combinations of categorical values, represented by C leaves, where $C = \prod_i K_i$. When $L = n_c$, each layer of the tree corresponds to a categorical variable. Often, one can prune nodes and collapse layers of the tree to reduce the categorical search space. Also, the tree can respect the sequential ordering of the variables by using parent nodes for choice of an algorithm and children nodes for algorithm-specific parameters.It is worth mentioning that we choose MCTS instead of the bandit-based algorithms (e.g.,

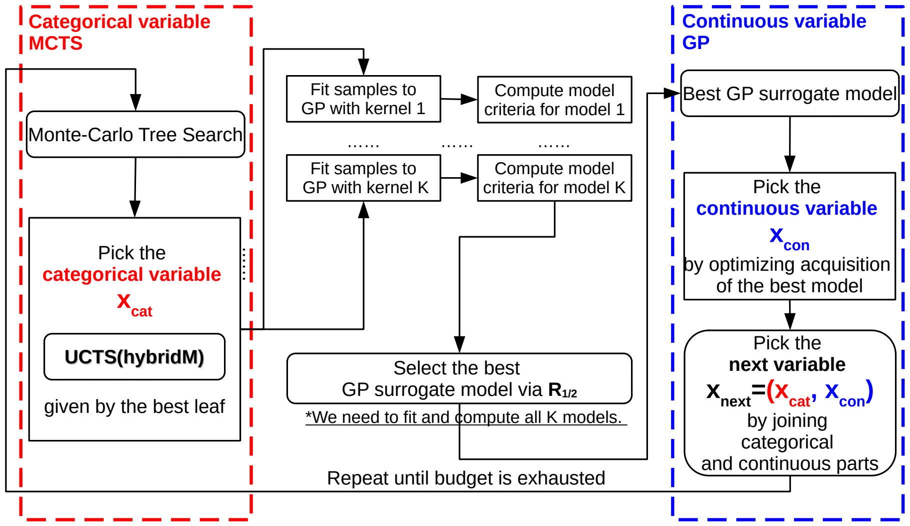

Categorical variable MCTS
Monte-Carlo Tree Search
Pick the categorical variable x_{cat}
UCTS(hybridM)
given by the best leaf
Fit samples to GP with kernel 1
Compute model criteria for model 1
Fit samples to GP with kernel K
Compute model criteria for model K
Select the best GP surrogate model via $R_{1/2}$
*We need to fit and compute all K models.
Continuous variable GP
Best GP surrogate model
Pick the continuous variable x_{con} by optimizing acquisition of the best model
Pick the next variable $x_{next}=(x_{cat}, x_{con})$ by joining categorical and continuous parts
Repeat until budget is exhausted

Figure 1: Algorithmic representation for the update step (i.e., heuristic search for the next sequential sample) in the proposed hybrid model, where we provide two different stretegies UCTS (hybridM) . Note that the fitting of GPs with different kernels can be parallelized rather than fitted sequentially.

CoCaBO and EXP3BO in Table 2) and sampling-based algorithm (e.g., skopt in Table 2) because bandit- and sampling-based algorithms perform poorly for large C (see Section 5.2 of Ru et al. (2020)).

The workflow of MCTS (Kocsis and Szepesvári, 2006) is summarized as follows: Consider a complete path $\boldsymbol{s} = (s_1, s_2, \cdots, s_L)$ of length L, where $s_1, s_2, \cdots, s_L$ represent the nodes (i.e., categorical values) at each level. This complete path corresponds to a full categorical part of the variable $\boldsymbol{x}_{\text{cat}}$ and can be evaluated to generate a reward function $r(\boldsymbol{s})$. To be precise, we can consider a continuous-variable GP conditioned on the path $\boldsymbol{s}$, i.e., the corresponding categorical value. When maximizing a function f, we use the black-box function $r(\boldsymbol{s}) = f(\boldsymbol{s}, \boldsymbol{x}^*_{\text{con}})$ as the reward value to be maximized, following Rakotoarison et al. (2019); Ru et al. (2020). We can use negative function values when minimizing the black-box function. The notation $\boldsymbol{x}^*_{\text{con}}$ is the new continuous variable value proposed by the maximization of the acquisition of the surrogate corresponding to the leaf node. In other words, a new function sample is generated to evaluate $\boldsymbol{s}$, see Line 13 of Algorithm 1.

Since the same path $\boldsymbol{s}$ can be visited multiple times during the search, we use the mean of the historically generated $r(s_1, s_2, \cdots, s_L)$ as the averaged reward function $\bar{r}(s_1, s_2, \cdots, s_L)$ for the leaf nodes. For each non-leaf interior node s_i, $i < L$, the reward function $\bar{r}(s_1, s_2, \cdots, s_i)$ is simply the average of $\bar{r}(s_1, s_2, \cdots, s_i, s_{i+1})$ for all s_{i+1} visited so far. Finally, we use the maximization of the following function, namely the upper confidence bound (UCB) in (Auer, 2002), as the *upper confident bound tree search (UCTS) policy*, to search for new categorical variable values:

$$\bar{r}(s_1, s_2, \cdots, s_i) + C_{UCB}\sqrt{\frac{\log n(s_1, s_2, \cdots, s_{i-1})}{n(s_1, s_2, \cdots, s_{i-1}, s_i)}}, \tag{1}$$

where the first and second terms serve respectively as the exploitation and exploration factors, C_{UCB} is a constant, and $n(s_1, s_2, \cdots, s_i)$ is the number of times the path $(s_1, s_2, \cdots, s_i)$ has been visited.

Once the new categorical values have been sampled, we follow an *update strategy*, by updating $n(\boldsymbol{s})$ and $\bar{r}(\boldsymbol{s})$ where $\boldsymbol{s}$ denotes a vector like $(s_1, s_2, \cdots, s_i)$, and the full path $(s_1, s_2, \cdots, s_L)$ corresponds to a categorical variable value and propagates the updated value along the complete path for each non-leaf node. The updated value is propagated along the complete path for each non-leaf node. After a new sample location is evaluated, the reward and visit count of all nodes along the path of the tree that led to the sample are updated. For each node s_i in the path, the visit count $n(s_1, s_2, \cdots, s_i)$ is incremented by 1, and the average reward $\bar{r}(s_1, s_2, \cdots, s_i)$ is updated to reflect the new sample's reward.

2.2 Dynamic Model Selection

Based on the hybridization of tree and GP structures, we could claim that our proposed model with the suggested search strategies can serve as generalizations of most existing hybrid models listed in Table 2. Specifically, MCTS reduces to MAB when $L = 1$, and random sampling when the sequential sample count *maxSampleSize* in Algorithm 1 is 0; the single mixed-variable GP can be separated into independent continuous-variable GPs if block-diagonal kernels that assume no correlation between categories are used for y (see Appendix B for more detail). As a result, with some tailoring, our proposed model structure can reduce to MOSAIC, GP (skopt), CoCaBO or EXP3BO.

Now, we introduce another important component in hybrid model, that is, the dynamic model selection. In the current paper, we consider the commonly used kernels for GP surrogates, summarized in Table 3 and detailed in Appendix B.

The transition from MCTS for parameter space search to dynamic surrogate model selection can be understood as follows. MCTS strategies determine sequential actions based on average rewards and potential improvement either deterministically or probabilistically. Similarly, in an online context like Bayesian optimization, model selection involves choosing the best surrogate model and the acquisition function, which balance the model's fit to observed data (i.e., goodness-of-fit) and the potential improvement (i.e., acquisition). Classical model selection criteria like AIC and BIC focus on the model's fit but overlook the potential improvement. To address this, we propose incorporating the acquisition func-

k	k_{cat}	k_{con}
$k_{\text{cat}} + k_{\text{con}}$	$k_{\text{MLP}}^{\sigma^2,\sigma_b^2,\sigma_w^2}$	$k_{\text{Matern}}^{5/2,\ell}$
$k_{\text{cat}} + k_{\text{con}}$	$k_{\text{Matern}}^{5/2,\ell}$	$k_{\text{Matern}}^{5/2,\ell}$
$k_{\text{cat}} + k_{\text{con}}$	$k_{\text{MLP}}^{\sigma^2,\sigma_b^2,\sigma_w^2} + k_{\text{Matern}}^{5/2,\ell}$	$k_{\text{Matern}}^{5/2,\ell}$
$k_{\text{cat}} \times k_{\text{con}}$	$k_{\text{MLP}}^{\sigma^2,\sigma_b^2,\sigma_w^2}$	$k_{\text{Matern}}^{5/2,\ell}$
$k_{\text{cat}} + k_{\text{con}} + k_{\text{cat}} \times k_{\text{con}}$	$k_{\text{MLP}}^{\sigma^2,\sigma_b^2,\sigma_w^2}$	$k_{\text{Matern}}^{5/2,\ell}$

Table 3: Default candidate kernels in hybrid model, see Appendix B for details.

tion into model selection, as an analog to the "potential improvement" concept in MCTS. This leads to dynamic model selection, where we adaptively select a covariance kernel from candidate kernels in each iteration, maximizing optimization performance. The dynamic approach is partly evidenced by empirical studies like Zeng and Luo (2017).

2.2.1 Classical kernel selection criteria

To maximize the performance of the model, we want to select a covariance kernel k from the candidates in Table 3, dynamically in each iteration during sequential sampling. Our framework allows to extend and add more candidate kernels at the cost of longer computational time. This dynamic kernel selection in BO has not been studied before and requires kernel selection criteria that incorporate both how well a model fits the data and how much potential improvement the model can lead to. We summarize a few classical kernel selection criteria with a discussion about their drawbacks, followed by the proposed rank-based criterion.

In an offline context, the model selection for GP regression falls into two different categories: one category is based on the marginal likelihood of the GP model; and the other category is the leave-one-out cross-validation (Rasmussen and Williams, 2006). In an online context, the Bayesian information criterion (BIC) has been used in continuous variable BO (Malkomes et al., 2016):

$$\text{BIC}_k = 2 \log \mathbb{P}_k(\boldsymbol{y}) - n_k \log n \tag{2}$$

where n_k is the number of kernel parameters of kernel k and n is the number of samples in the sample response vector $\boldsymbol{y}$. Other similar ones are the Akaike information criterion (AIC), where $n_k \log n$ is replaced by $2 \cdot n_k$, and the Hannan–Quinn information criterion (HQC) (Hannan and Quinn, 1979), where $n_k \log n$ is replaced by $2n_k \log(\log n)$.

In addition, one can also use the (negative) log marginal likelihood $\log \mathbb{P}_k(\boldsymbol{y})$ (indicating how well a model fits the data) or the maximal acquisition function value $\mathbb{A}_k(\boldsymbol{y}) := \max_x \text{EI}(x, \mathbb{P}_k(\boldsymbol{y}))$ (indicating the potential improvement of the next sequential sample) as the kernel selection criteria. However, the balance between log likelihood, sample size, and acquisition function is critical for these criteria to show stable and satisfactory results.

2.2.2 Online kernel selection criteria

Kernel selection in GPs, traditionally a manual and expert-driven task, is computationally intensive (Abdessalem et al., 2017; Bitzer et al., 2022). Consider two extreme scenarios in kernel selection for Bayesian optimization – one based purely on likelihood and the other on pure acquisition (optimism). The pure likelihood approach may lead to overly conservative model choices, possibly ignoring promising but less certain alternatives (See Figure 4). Conversely, a pure acquisition-based approach risks favoring models that are overly optimistic but not necessarily accurate when away from the possibly local optimum, as in the case of a trivial surrogate that returns infinity everywhere in the design space in minimization contexts.

To choose among these candidate kernels in Table 3 for our surrogate model, a novel kernel selection criterion is proposed next. In an online context for BO, we combine the strengths of optimism (highest acquisition) with likelihood measures for kernel selection. This combination is crucial to balance the explorative nature of acquisition with the data-based realism of likelihood assessments.

Our decision to combine the ranks of these criteria, rather than their numerical values, is based on the need to mitigate the extremes of both approaches. Ranking allows us to integrate the optimistic explorative nature of acquisition with the grounded realism of likelihood assessments. This method ensures a more balanced and nuanced kernel selection process, and can be shown (Figure 3) to outperform fixed kernels in online scenarios.

Proposed rank-based kernel selection criterion. We propose a stable and effective selection criterion that well balances likelihood and acquisition function by considering the rank $R_{\mathbb{P}}(k)$ of log likelihood $\log \mathbb{P}_k(\boldsymbol{y})$ and the rank $R_{\mathbb{A}}(k)$ of acquisition $\mathbb{A}_k(\boldsymbol{y})$ (larger quantities correspond to higher ranks), both among all k. A generic formulation of such kernel selection criteria could take the form of $R_{\mathbb{P}}(k) + \alpha R_{\mathbb{A}}(k)$. In what follows, we focus on the following *rank-based kernel selection* criterion

$$R_{1/2}(k) = R_{\mathbb{P}}(k) + \frac{1}{2} R_{\mathbb{A}}(k), \tag{3}$$

where the factor $\frac{1}{2}$ puts less emphasis on the acquisition function (analogous to BIC) since it is usually more sensitive than log likelihood due to the fact that it incorporates uncertainty, and also largely avoids tied ranks. Sometimes an adaptive coefficient that puts more emphasis on the acquisition function as more samples are generated can lead to more stable results, yielding e.g., an adaptive selection criterion:

$$R_{ad}(k) = R_{\mathbb{P}}(k) + \frac{2 \cdot i}{n} R_{\mathbb{A}}(k), i = 1, 2, \cdots, n, \tag{4}$$

where n is the deterministic sampling budget (i.e., *maxSampleSize* in Algorithm 1) and i means the i-th step among all n steps. In this paper, we use $R_{1/2}$ as the default criterion.

The traditional kernel selection criteria (e.g., AIC, BIC) focus solely on the surrogate model's goodness-of-fit (like an "exploitation" part), making it apt for offline fixed datasets. Analogous to the Bayesian optimization's acquisition function, which balances exploration and exploitation, we suggest integrating an "exploration" component into the kernel selection criteria (3) and (4). Direct summation of these components can be problematic due to their differing magnitudes. Empirically, rank sums have proven more consistent. The coefficients $1/2$ and $2 \cdot i/n$ serve as weights to balance exploration and exploitation during kernel selection. The former emphasizes the surrogate model's goodness-of-fit, while the latter, increasing with sequential sampling, accentuates exploration, especially when the goodness-of-fit near existing optima approaches perfection, as observed in standard Bayesian optimization.

We use the following example to illustrate how the rank-based criterion $R_{1/2}$ works, where $1/2$ is chosen empirically, and refer to Figure 4 comparing different selection criteria in the optimization context. The adaptive variant R_{ad} with weights other than $1/2$ can be computed in a similar fashion.

Example. (Computation of $R_{1/2}$) Consider the candidate surrogate models with kernels k_1, k_2, k_3, and suppose that we have already computed their likelihoods based on the same set of samples $\boldsymbol{y}$. Then, we can compute the relevant quantities below for kernel selection purposes.

Based on the data below, we prefer the best-fitted model and the best acquisition function, which is attained by a GP model with the kernel k_1. The efficacy of the proposed kernel selection criterion will be demonstrated with optimization examples in Section 3.1.3.

kernel k	k_1	k_2	k_3
$\log \mathbb{P}_k(\boldsymbol{y})$	$\log \mathbb{P}_{k_1}(\boldsymbol{y}) = 2.6$	$\log \mathbb{P}_{k_2}(\boldsymbol{y}) = 2.5$	$\log \mathbb{P}_{k_3}(\boldsymbol{y}) = -2.1$
$R_{\mathbb{P}}(k)$	3	2	1
$\mathbb{A}_k(\boldsymbol{y})$	$\mathbb{A}_{k_1}(\boldsymbol{y}) = 2$	$\mathbb{A}_{k_2}(\boldsymbol{y}) = -1.5$	$\mathbb{A}_{k_3}(\boldsymbol{y}) = 9.5$
$R_{\mathbb{A}}(k)$	2	1	3
$R_{1/2}$	4	2.5	2.5

After generating a new categorical variable sample $\boldsymbol{s}$ via MCTS (Section 2.1), and selecting a mixed-variable GP kernel $y(\boldsymbol{x})$ (Appendix B) via the rank-based selection criterion, one can search for new continuous variable samples using the regular GP with the chosen kernel, conditioned on the categorical sample $\boldsymbol{s}$.

2.3 Putting It Together: Algorithm for New Hybrid Model

In our hybrid model, we consider the optimization for an objective function $f(\boldsymbol{x}) = f(\boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{con}})$ with $\boldsymbol{x}_{\text{cat}}$ denoting the part of the categorical variables and $\boldsymbol{x}_{\text{con}}$ denoting the part of the continuous vaiables, for which the proposed hybridM model builds a mixed-variable GP but separate the search phase using MCTS for $\boldsymbol{x}_{\text{cat}}$ and GP for $\boldsymbol{x}_{\text{con}}$. Algorithm 1 summarizes the complete pipeline when using the hybrid model, which efficiently constructs the categorical parameter space as a tree structure, and associates each leaf node with a value of $\boldsymbol{x}_{\text{cat}}$, i.e. a combination of categories.

Our hybrid models use tree and GP for the search phase, but only mixed-variable GP for the surrogate modeling. In MOSAIC (Rakotoarison et al., 2019) with fixed kernels, each categorical value combination is mapped to an independent GP. In our hybrid model, we employ a shared GP for all samples. During the search phase, categorical variables utilize a tree, while continuous ones use a GP. In the surrogate fitting phase, a single GP surrogate is applied, complemented by the novel online kernel selection. This unique construction leverages the efficiency of tree-search while ensuring adaptable GP model fitting with a list of mixed variable covariance kernels (See Appendix B).

The algorithm assumes n_0 pilot samples, which are initial samples that are observed prior to any sequential sampling or BO surrogate training. We are allowed to search for *maxSampleSize* sequential samples. The hybrid model relies on three essential steps to generate the sequential samples:

1. Heuristic MCTS search for the categorical part in new samples and update the MCTS via back-propagation (Lines 3 and 13 of Algorithm 1).

2. Dynamic kernel selection for the best mixed-variable GP surrogate based on both the categorical and continuous parts of the next variable (Lines 6-8 of Algorithm 1).

3. Acquisition maximization of the continuous value conditioned on the categorical values to add new samples to the GP surrogate. (Lines 9-13 of Algorithm 1).

We need to fit the model with all candidate kernels separately to get different acquisition functions and selection criteria, which is expensive. However, this step can be parallelized and we can ensure that the selected kernel and surrogate will have the optimal balance between exploring and exploiting.

Algorithm 1: hybrid model algorithm with dynamic kernel selection and surrogate fitting.

```
Data: $X_{n_0,d}$ and $Y_{n_0}$ (data matrices consisting of $n_0$ pilot samples of $d$ coordinates
      in the variable, i.e., $d$ means the total number of continuous and categorical
      parts.)
Input: $maxSampleSize$ (the maximal number of evaluations of $f$), $L_k$ (the list of
       covariance kernels we want to consider).
Result: One tree structure with searching information, one GP surrogate model
        with trained covariance kernel.
1  Initialize $X = X_{n_0,d}$, $Y = Y_{n_0}$ and $i = n_0$ to be the sample size. Let $X_{\text{cat}}$ denote
     the categorical part of $X$;
2  Train MCTS with $X_{\text{cat}}$ and $Y$;
3  Train the GP surrogate with $X$ and $Y$, and a randomly selected kernel from the
     candidate kernels (see Section B);
4  while $i \leq maxSampleSize$ do
5      Find the next categorical variable $\boldsymbol{x}^*_{\text{cat}}$ according to UCTS in Section 2.1 (i.e.,
         hybridM) ;
6      for $k$ in $L_k$ do
7          Train the GP surrogate $g_k$ with $X, Y$ and the covariance kernel $k$ via
             likelihood maximization (MLE);
8          Compute $R_{1/2}(k)$ defined in (3) (or (4)) for the surrogate $g_k$ and kernel $k$;
9      Find the kernel $k^*$ that maximizes (3) (or (4));
       /* The best kernel $k^*$ may be different for each iteration.      */
10     ;
11     $\boldsymbol{x}^*_{\text{con}} = \arg\max_{\boldsymbol{x}_{\text{con}}} \text{EI}((\boldsymbol{x}^*_{\text{cat}}, \boldsymbol{x}_{\text{con}}), g_{k^*})$;
12     $\boldsymbol{x}_{next} = (\boldsymbol{x}^*_{\text{cat}}, \boldsymbol{x}^*_{\text{con}})$;
13     Append $\boldsymbol{x}_{next}$ to $X$, and $f(\boldsymbol{x}_{next})$ to $Y$;
14     Append $f(\boldsymbol{x}_{next})$ to the leaf node history and back-propagate;
15     Let $i = i + 1$;
```

3 Experiments and Analyses

3.1 Synthetic Functions

We compare the proposed hybrid models against the following methods we discussed above: TPE (Bergstra et al., 2011), SMAC (Hutter et al., 2011), EXP3BO (Gopakumar et al., 2018), CoCaBO (Ru et al., 2020), along with skopt optimizers (Head et al., 2020) as baselines. For skopt optimizers, skoptGP uses one-hot encoding for categorical variables by default; skoptForest uses a plain random forest estimator; skoptDummy randomly samples among all possible combinations of categories. Recall that C is the total number of possible combinations of categorical values with n_c categorical variables. In the case of small C, we also provide roundrobin BO (Bergstra et al., 2011) results, since roundrobin BO is clearly not favored in the large C case.

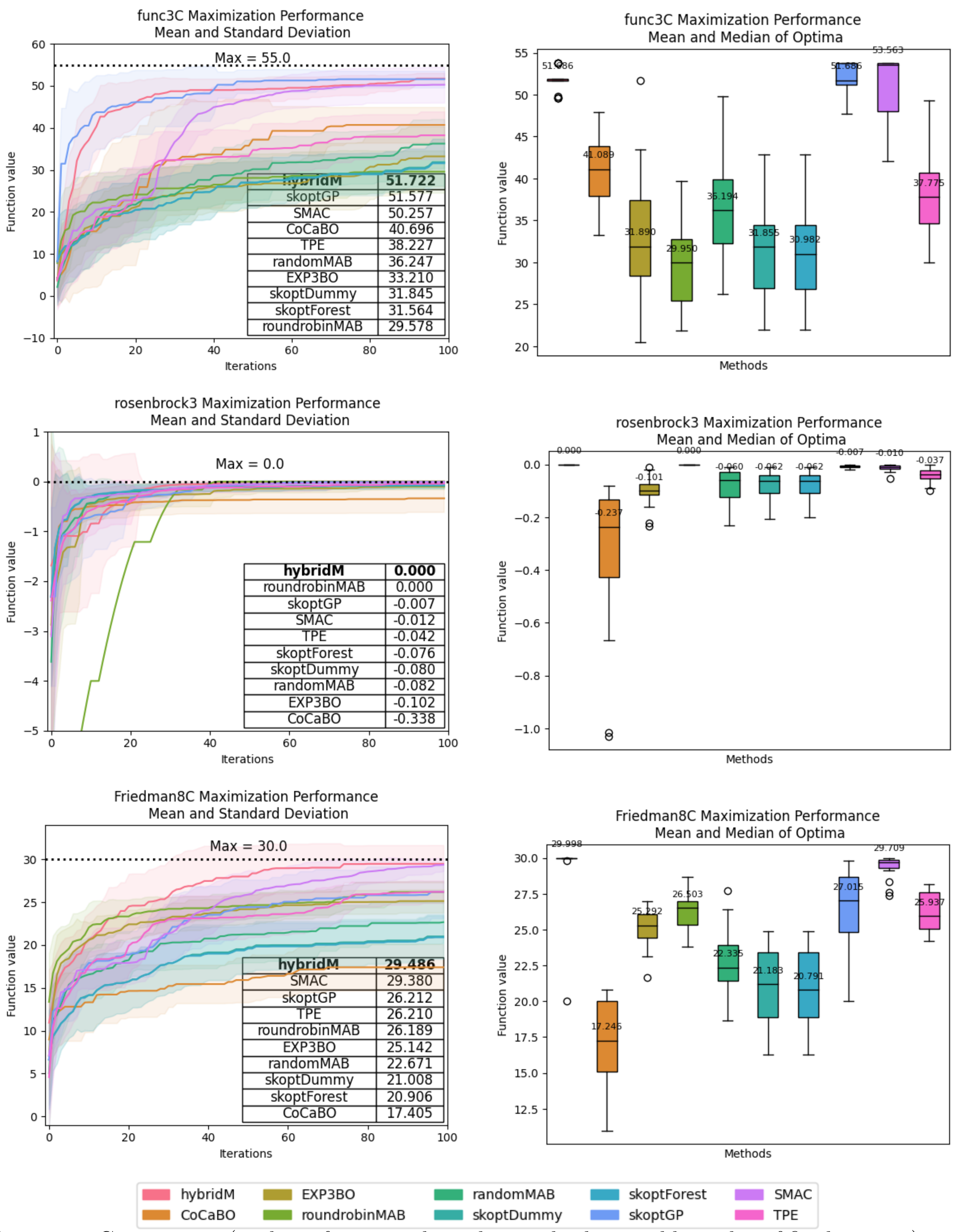
We compare the following methods for scientific applications (specific versions in parentheses): TPE ($\texttt{hyperopt} == \texttt{0.2.5}$), SMAC ($\texttt{smac} == \texttt{1.2.0}$), skopt ($\texttt{scikit} - \texttt{optimize} == \texttt{0.9.0}$ with $\texttt{scikit} - \texttt{learn} == \texttt{1.0}$) and our hybrid models. Note that we do not include the bandit-based methods (CoCaBO, EXP3BO, roundrobinMAB, randomMAB) here as they require a predefined scaling factor in the objective functions to avoid overflow problems in updating the reward.

The metric we use to express the computational budget (x-axis) is the number of function evaluations (i.e., samples observed from the black-box function). In the case of black-box functions, all function evaluations are equally expensive in terms of time. The performance measure (y-axis) is the maximum objective black-box function value (or minimum in minimization problems) that we observed so far at the sample size. For a single batch, we can plot a curve for this "best-so-far" maximum value against the sample sizes explored. For multiple batches (with different random seeds), we display the point-wise means and standard deviations for multiple curves. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. We use 10 pilot samples and 90 sequential samples in all hybrid models and assign the same budget to other methods. Our study opted for a fixed number of pilot samples to maintain consistent comparisons across various mixed-variable methods. It is important to note that the scaling of pilot samples for higher dimensions across different methods remains an open problem in the field. Therefore, our approach reflects a compromise to balance methodological consistency with the diverse requirements of the different models in comparison following the existing comparison (Ru et al., 2020).

There are three kinds of important benchmarking functions that we should be aware of in the mixed-variable setting. It is worth pointing out that different approaches behave quite differently when applied to different kinds of benchmark functions. This is caused by the different correlations between categories, which is usually not explicitly modeled in surrogates, especially for non-hybrid models.

3.1.1 Categorical benchmarking

The first kind of benchmark functions are created by assigning different continuous-variable functions to different categories as shown in the first row of Figure 2. This kind of function is represented by a (scaled version of) the function func3C ($n_c = 3, C = 3 \times 5 \times 4 = 60$) from (Ru et al., 2020). The correlation between categories is not clear at all or non-existent. The categorical benchmarking function is self-explanatory, like the piece-wise constant functions involving discontinuity. This kind of synthetic function mainly aims at checking if the mixed model is suitable when the correlation is not necessarily continuous,



hybridM	**51.722**
skoptGP	51.577
SMAC	50.257
CoCaBO	40.696
TPE	38.227
randomMAB	36.247
EXP3BO	33.210
skoptDummy	31.845
skoptForest	31.564
roundrobinMAB	29.578

hybridM	**0.000**
roundrobinMAB	0.000
skoptGP	-0.007
SMAC	-0.012
TPE	-0.042
skoptForest	-0.076
skoptDummy	-0.080
randomMAB	-0.082
EXP3BO	-0.102
CoCaBO	-0.338

hybridM	**29.486**
SMAC	29.380
skoptGP	26.212
TPE	26.210
roundrobinMAB	26.189
EXP3BO	25.142
randomMAB	22.671
skoptDummy	21.008
skoptForest	20.906
CoCaBO	17.405

Figure 2: Comparison (with performance line plot per budget and box plot of final optima) of performance over 20 batches for: (row 1) a scaled version of the function func3C in Ru et al. (2020) with maximum 55; (row 2) a scaled version of the discrete Rosenbrock function of 7 dimensions (4 continuous variables in $[-5, 5]$ and 3 categorical variables in $\{-5, -4, \cdots, 4, 5\}$) in Malkomes et al. (2016) with maximum 0; (row 3) function (6) with maximum 30.

which is the typical setting the previous research focused on.

We can observe that hybrid models are among the best models, and enjoys a fast convergence rate. For the categorical benchmarking black-box functions, hybridM needs a lower sampling budget to reach a place near the optimal value and keeps improving until reaching the actual optima. In the extreme case, this is similar to creating C independent surrogate models for each category (see Table 2). However, we only use one surrogate model for better usage under a limited sampling budget, as explained in Appendix B.

In this kind of benchmarking, the bandit strategy is quite suitable for exploring the categories compared to covariance models. From our empirical studies, hybridM and the one-hot encoded GP (denoted by skoptGP, where categorical is differentiated from continuous by encoding) are all competitive. When C is small, the one-hot encoding is effective for the first kind of benchmarking.

3.1.2 Integer-like benchmarking

The second kind of benchmark function is constructed by discretizing the functions with continuous variables only. That is, the continuous variables are forced to take integer values only and treated as if they are categorical. This kind of function is represented by discrete Rosenbrock (Malkomes et al., 2016), Ackley-cC (Nguyen et al., 2019) and GP sample path functions considered by Garrido-Merchán and Hernández-Lobato (2020). The correlation between categories is induced by the original dependence between continuous variables.

The well-known Rosenbrock function (Surjanovic and Bingham, 2022) can be scaled and defined for an arbitrary dimensional domain as:

$$f(\boldsymbol{x}) = \frac{-1}{10000}\sum_{i=1}^{d-1}\left[100(\boldsymbol{x}_{i+1}-\boldsymbol{x}_i^2)^2+(\boldsymbol{x}_i-1)^2\right], [-5,5]^d \subset \mathbb{R}^d \to \mathbb{R}. \tag{5}$$

For example, when $d = 7$ we may keep the first 4 dimensions continuous. However, for the last 3 dimensions, instead of letting $\boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7$ vary continuously in $[-5, 5]$, we force these variables to assume values in the set $\{-5, -4, \cdots, 4, 5\}$ of 11 integers as categorical values. These integers are taken as categorical encodings ($n_c = 3, C = 11 \times 11 \times 11 = 1331$). This kind of discretization is another way of creating mixed-variable benchmarking functions.

From the second row of Figure 2, we can observe that our hybridM model has a slow start, but eventually reaches the actual optima as bandit-based surrogate models like roundrobinMAB. The variances of all the surrogate models considered are relatively low, which echoes the results in Malkomes et al. (2016). In this kind of benchmarking, the bandit strategy is still reasonably good in exploring the categories, but clearly less efficient compared to covariance models. This is because the correlations between categories are induced by discretizing the continuous domain, and before discretizing, the relation is already well modeled by (stationary) kernels.

3.1.3 Inactive benchmarking

The third kind of benchmark function contains one or more "inactive" categorical or continuous variables. In other words, some of the variables do not affect the function value at all. We highlight the challenges in identifying inactive variables in black-box optimization, which is a task not easily addressed by existing methods like post-hoc sensitivity analysis. Although our method does not directly address variable selection, emphasizing these difficulties underlines the critical need for advanced BO methods to discern and manage inactive variables.

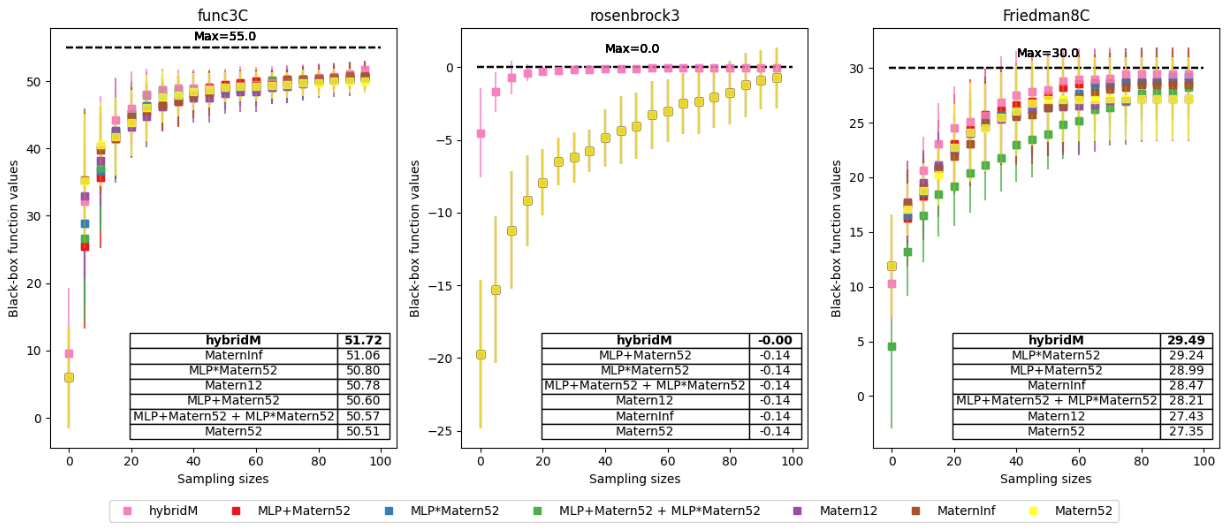
A representative example of this kind is the Friedman function. The Friedman function is defined in such a way that not all categorical variables are active; this function is a modified version of the well-known "inactive function" by Friedman (1991) and has been extensively studied in different contexts of variable selections (Chipman et al., 2010). We modify this function by appending several more inactive variables to the Friedman-8C function:

$$\begin{aligned} f(\boldsymbol{x}) &= 10\sin(\pi \boldsymbol{x}_1\boldsymbol{x}_2)\cdot \mathbf{1}(\boldsymbol{x}_7=0) + 20(\boldsymbol{x}_3-0.5)^2 \\ &\quad + 10\boldsymbol{x}_4\cdot\mathbf{1}(\boldsymbol{x}_9=0) - 10\boldsymbol{x}_4\cdot\mathbf{1}(\boldsymbol{x}_9=1) + 5\boldsymbol{x}_4\cdot\mathbf{1}(\boldsymbol{x}_9=2) + 5\boldsymbol{x}_5 \\ f :& \boldsymbol{x}\in[0,1]^6\times\{0,1,2\}\times\{0,1,2,3,4\}\times\{0,1,2\}\times\{0,1,2,3\}^3\times\{0,1\}^2\to\mathbb{R}. \end{aligned} \tag{6}$$

The function comprises 6 continuous and 8 categorical variables, with limited combinations due to $\boldsymbol{x}_{10},\cdots,\boldsymbol{x}_{14}$ being restricted to $0,1,2,3$ ($n_c=8, C=11520$). Its non-stationarity, particularly as $\boldsymbol{x}_9=1$ shows different correlations with $\boldsymbol{x}_9=0$ and 2, makes it unsuitable for stationary kernels and well-suited for our non-stationary categorical kernel described in Appendix B.

Testing the Friedman8C function (Figure 2), our hybridM model surpasses others in convergence and final optima, followed by SMAC. The non-stationary MLP kernel amplifies the bandit's effectiveness, focusing on active categories. Most batches show hybridM reaching the true maximum efficiently. Stationary kernels often fail in mixed spaces, as our experiments confirm. HybridM excels with inactive variables or hierarchical spaces, and the bandit approach is a subset of hybrid models.

Using the Friedman-8C function ((6)), we illustrate the dynamic kernel selection's superiority in hybrid models (Figure 3). It enhances convergence and optima quality, with our expanded kernel families and a shared GP surrogate modeling both continuous and categorical variables. Figure 4 shows our rank-based criterion outperforming others in kernel selection (Section 2.2.2).

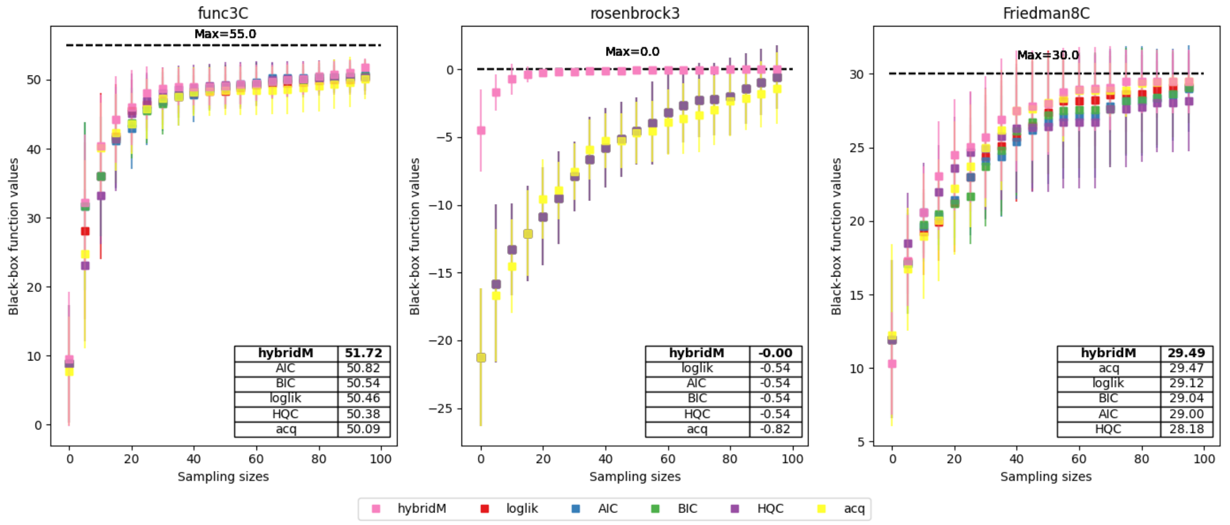


hybridM	51.72
MaternInf	51.06
MLP*Matern52	50.80
Matern12	50.78
MLP+Matern52	50.60
MLP+Matern52 + MLP*Matern52	50.57
Matern52	50.51

hybridM	-0.00
MLP+Matern52	-0.14
MLP*Matern52	-0.14
MLP+Matern52 + MLP*Matern52	-0.14
Matern12	-0.14
MaternInf	-0.14
Matern52	-0.14

hybridM	29.49
MLP*Matern52	29.24
MLP+Matern52	28.99
MaternInf	28.47
MLP+Matern52 + MLP*Matern52	28.21
Matern12	27.43
Matern52	27.35

Figure 3: Comparison of performance between different but fixed kernels of GP surrogate with the selection criterion 3 and fixed kernel GP surrogates on the functions func3C in Ru et al. (2020), (5)and (6) over 20 batches. The actual maximum is shown by dashed lines.



hybridM	51.72
AIC	50.82
BIC	50.54
loglik	50.46
HQC	50.38
acq	50.09

hybridM	-0.00
loglik	-0.54
AIC	-0.54
BIC	-0.54
HQC	-0.54
acq	-0.82

hybridM	29.49
acq	29.47
loglik	29.12
BIC	29.04
AIC	29.00
HQC	28.18

Figure 4: Comparison of performance between the hybrid models with different selection criteria: acq (acquisition function only), AIC, BIC, HQC, loglik (log likelihood) and $R_{1/2}$ in (3), on the functions func3C in Ru et al. (2020), (5)and (6) over 20 batches. The actual maximum is shown by dashed lines.

To sum up, the hybrid model has the best performance among all tested methods on the benchmark functions of the first and third kinds in the sense that it exhibits both fast or competitive convergence and best average optima. For the second kind of benchmarking function, the hybrid model still exhibits comparable average optima, but does not have the fastest convergence rate (e.g., compared to skoptGP and skoptForest).

Type	Hyper-parameter	Values
Categorical	Number of Layers	$\{3,4,5\}$[2]
	Activation for Layers	$\{\tan h, \text{ReLU}, \text{sigmoid}, \text{linear}\}$ [3]
Continuous	Layer Sizes	$[4,16] \cap \mathbb{Z}$[4]
	Initial Learning Rate	$10^{[-5,0]}$
	Layer Batch Size	$[8,64] \cap \mathbb{Z}$
	Dropout Rate for Layers	$[0.0, 0.5]$

Table 4: Different hyper-parameters in the regression neural network.

3.2 Scientific Applications

Next, we switch to expensive black-box functions arising in machine learning and scientific computing applications. As we mentioned earlier, the sampling budget is limited due to the fact that each sample is costly to evaluate. The computational times for surrogate models are neligible compared to the evaluation time of black-box functions.

We cannot standardize the reward, since the black-box function cannot be bounded and scaled reasonably without knowing its range. Rescaling of reward functions have been known to be non-trivial and task-dependent (Fouché et al., 2019; Lykouris et al., 2020). In addition, our hybrid model does not need standardization along the tree structure, in contrast to the bandits.

3.2.1 Neural network regression hyper-parameter tuning

The feed-forward neural network is used for a regression task. The regression task is to use the 13 observed variates in the Boston Housing dataset (Harrison Jr and Rubinfeld, 1978) to predict the response variable of the housing price (positive continuous variable).

We take the negative mean square error (MSE) between the predictor and the observed response as our objective function and set the task to tune the hyper-parameters of the regression network among the range described in Table 4 to maximize the negative MSE. In our study, there is a collection of multivariate regression data sets from the public repository with heterogeneity and a red-shift dataset (Elsken et al., 2019; Luo and Pratola, 2022) from cosmology simulations. The 3D-HST galaxy dataset (Skelton et al., 2014) underwent preprocessing: invalid redshifts and negative flux values were removed; flux was converted to magnitude; and the data was split into an 80-20 training-test ratio. The target for regression was the z_best column, indicating redshift.

[2]the number of layers includes the input/output layer but not the dropout layer.

[3]all dense layers share the same activation function.

[4]all dense layers share the same size.

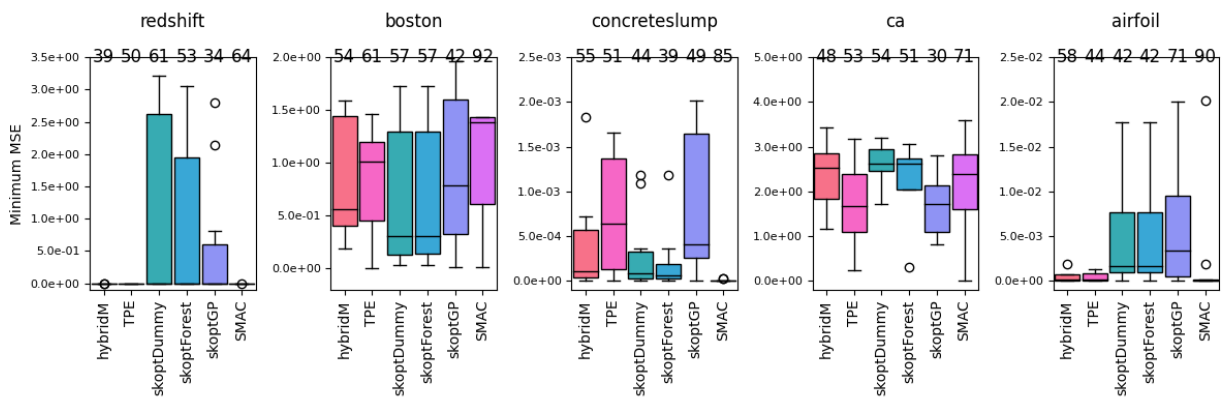


Figure 5: Comparison boxplots of performance between different methods on the different tuning methods on the regression neural network specified in Table 4 for datasets in Table 5 over 10 repeated batches. The theoretical minimum in-sample MSE is 0. We also display the average iterations needed to attain optima at the top of each box.

name	n (sample size)	d (dimension)	source
Galaxy redshift	4173	30	Skelton et al. (2014)
Boston housing	506	13	StatLib[5]
Concrete slump test	103	10	uci_ml[6]
California housing	20640	8	scikit-learn[7]
Airfoil self-noise	1503	6	uci_ml[8]

Table 5: Data format and source for the neural network regression task.

Figure 5 indicates that our methods have attained (near-)best optima on redshift, airfoil and concreteslump. Hybrid models also have reasonable convergence rates on the boston and california housing datasets, with comparable optima to the rest tuners. This echoes the observation of Rakotoarison et al. (2019) that a tree-based search is better able to handle larger categorical spaces. For the galaxy redshift dataset, we found that the usual tuners from skopt (not addressing the categorical nature of the hyper-parameters) do not have good performance. TPE, SMAC and hybrid models show reasonable tuning results, by taking categorical structure into consideration (especially in airfoil and redshift). But SMAC usually takes more budgets to attain the optimum, compared to relatively quick convergence shown by TPE and hybrid models. When we have some apriori information, the tree structure in TPE and hybrid models could help in searching categorical space efficiently. In this tuning neural network example, we do not claim generic improvement but the robustness (compared to large variation of skoptForest, skoptGP) and efficiency (compared to slow exploration of SMAC, skoptDummy) of the hybrid models. It remains

[5]http://lib.stat.cmu.edu/datasets/boston
[6]https://archive.ics.uci.edu/dataset/182/concrete+slump+test
[7]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html
[8]https://archive.ics.uci.edu/dataset/291/airfoil+self+noise

Type	Parameter	Values
Categorical	Fill-in Reduction Ordering	$\{'metis','parmetis','geometric'\}$
	Compression Algorithm	$\{'hss','hodbf','blr'\}$
	Minimum Compressed Separator Size	$\{2,3,4,5\} \times 1000$
Continuous	Leaf Size in Compression	$2^5, 2^6, \cdots, 2^9$
	Compression Tolerance	$10^{-6}, \cdots, 10^{-1}$

Table 6: Different parameters in the STRUMPACK for Poisson3d with grid size 100. *For the continuous variables, we treat their exponents as continuous and then round to the nearest integer.

a nontrivial problem how our tuning performance in this simple neural network problem can be generalized to generic neural architecture search (Elsken et al., 2019).

After the initial comparison and examining the optimal configuration found by the tuning methods, we are more informed that: 3-layer models are not performing well compared to 4- and 5-layer models; 4-layer models are not performing well when the activation function is sigmoid. All models do not behave well for linear activation functions. It is hard to incorporate these pieces of information into SMAC and TPE. In the skopt framework, the typical way is to assign a very small reward value (e.g., negative infinity) to these combination of parameters. In hybrid models, the learned reward history allows us to transfer the information more easily compared to other tuners. The hybridM and hrbridMD do not always find similar optima, but often they will provide similar convergence rates.

3.2.2 STRUMPACK for three-dimensional Poisson problem

In this computationally expensive experiment, we use the STRUMPACK (Ghysels et al., 2017, 2016) sparse solver as a black-box application with several categorical variables. STRUMPACK is a high-performance numerical library for structured matrix factorization and we set the tuning problem to be solving as fast as possible a 3-dimensional Poisson equation on a $100 \times 100 \times 100$ regular grid, leading to a sparse matrix of dimension $1M \times 1M$. We consider 5 tuning-parameters with 3 categorical variables and 2 continuous variables as shown in Table 6. Each function evaluation requires 10 to 500 seconds using 8 Haswell nodes of the Cori, a Cray XC40 machine, at NERSC in Lawrence Berkeley National Laboratory. For each tuner, we execute 10 batches with 100 sequential samples due to a computational time limit (see Figure 6). We measure the "tuning budget" by recording the actual cumulative execution time in the function evaluation.

The main time complexity falls on the execution of STRUMPACK, therefore, we provide the plot of optimal execution time against the average cumulative time taken by the STRUMPACK in Figure 6. We can see that hybridM attains the optimal execution time quickly, using only 50% of the overall time to achieve the same or better optima compared to the other models. And hybridM concentrates within 10% of the optimal configuration at least 4 times faster than the other model (Figures 6 and 7.) The other interesting observation is that simple skopt models reach a reasonable optimal execution time faster than TPE and SMAC. This can be further confirmed by the histogram indicating the exploration pattern in Figure 7. The better surrogate method should spend most of the sample budget in exploring the configurations that take less execution time per execution. In this figure, we saw that hybridM, TPE and SMAC spend most of the sequential samples with near-optimal execution time, while the simple skopt models explore sequential samples with worse execution times. Some badly chosen parameter configurations can take the STRUMPACK up to 500 seconds to complete, so the tuning cost is relatively high even for 100 sequential samples but still far lower than an exhaustive search.

Although in our application the application execution takes the most time, the time complxeity of hybrid models is not worse than that of a regular GP surrogate used in BO. The time complexity of the each selection at the node of a tree algorithm can be computed as $O(m)$ since the UCB selection criterion takes $O(1)$, where m is the number of sequential children nodes. Since the number n_c of categories is the same as the number of layers in the tree and the i-th category has C_i categorical variables, we need $O(n \cdot \sum_{i=1}^{n_c} C_i) = O(n \cdot n_c \cdot \max_i C_i)$ complexity for MCTS search for all n sequential samples, in contrast to $O(n \cdot \prod_{i=1}^{n_c} C_i)$ complexity for a MAB. At each leaf node, we need to fit K different GP surrogate models and select the best kernel based on our criterion. This step requires time complexity is expressed by $O(Kn^3)$, where we use all n samples for the joint GP surrogate, different from MOSAIC. The total time complexity of our hybrid model is $O(n \cdot n_c \cdot \max_i C_i + Kn^3)$, bounded from above by a cubic term in n, which is the same as regular GP.

This experiment reveals an interesting phenomenon: The simple models (e.g., skoptGP and skoptForest) may attain a reasonable optimum relatively quickly but fail to improve further. Advanced models (e.g., TPE and SMAC) may be slow in achieving the same optimal level but keep improving as samples accumulate. In stark contrast, hybridM is unique in the sense that it would automatically select among simple and advanced surrogate models with different kernels as the sampling happens, which empirically balances between convergence rates and optimality.

4 Contributions and Future Work

This paper presents an efficient mixed-variable BO framework, using hybrid models (hybridM) that leverage novel MCTS strategies for searching the categorical variable space and GP for searching the continuous variable space. Hybridization of tree and GP is a uni-

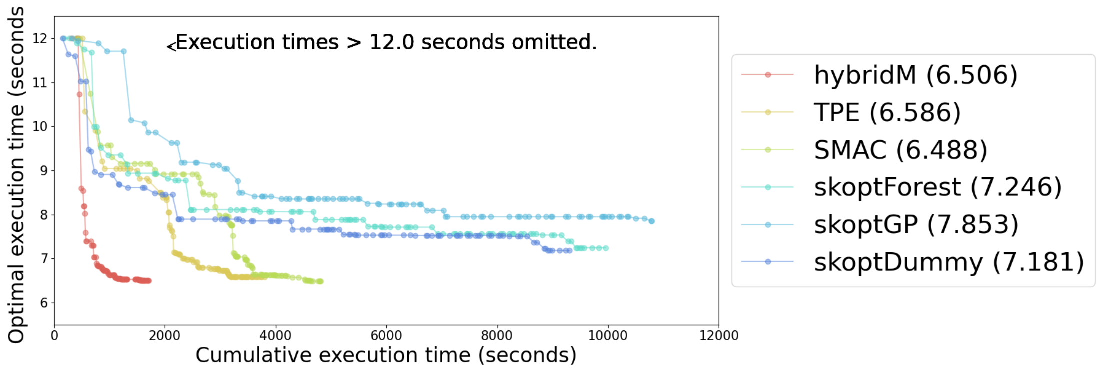


Figure 6: Comparison of performance between different methods on the STRUMPACK for Poisson3d with grid size 100. We display the actual (averaged over 10 batches) cumulative execution time used for each sequential sample, and each line contains 100 sequential samples as in Figure 8.

fied BO framework that encompasses several state-of-the-art mixed-variable optimization methods, leading to efficient search for categorical space. Our approach distinguishes itself by incorporating an improved dynamic kernel selection criteria, featuring a novel family of kernels well-suited for mixed variables, and leveraging tree to reduce the categorical search space.

The key contribution of this chain of thought is the innovative integration of concepts from MCTS into dynamic model selection for Bayesian optimization using a dependent GP, where existing methods (e.g., MOSAIC) cannot handle limited sample per category. This approach introduces a novel perspective on model selection for online data, which traditionally focuses on the goodness-of-fit of the model to observed data. By incorporating the acquisition function, which represents the potential improvement, we extend the model selection criteria to consider not only how well the model fits the data, but also how much potential improvement the model can lead to. This dynamic model selection process, which adaptively selects the best covariance kernel from candidate kernels in each iteration, enhances Bayesian optimization performance.

We identified different types of benchmarking problem and provide up-to-date benchmarks comparing these state-of-the-art BO methods for mixed variables. Moreover, we applied the proposed algorithm to ML and HPC applications to demonstrate its effectiveness compared to most state-of-the-art mixed-variable BO methods. And these innovations are combined into a unified software pipeline.

As mentioned in Appendix A, Bayesian strategies will learn posterior information through the updated reward functions or Dirichlet posteriors at nodes. This opens the door to performing transfer learning across tasks in mixed-variable tuning problems (Liao et al., 2023; Sid-Lakhdar et al., 2019). We recognize the limitations in Tesauro et al. (2012)

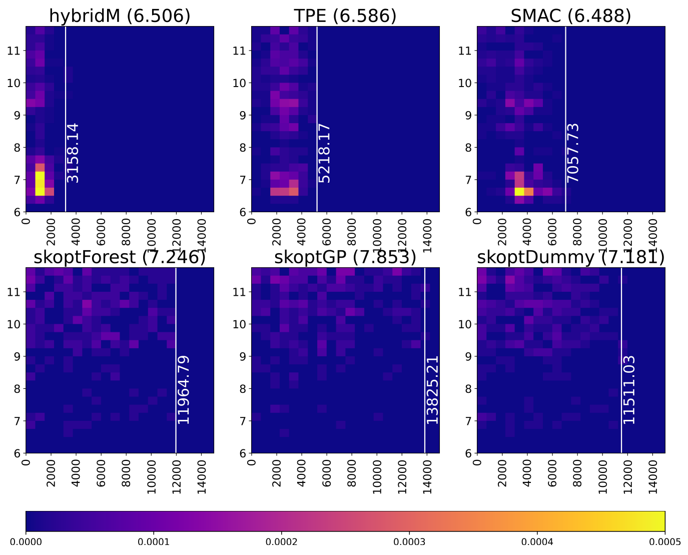


Figure 7: The histogram displaying the density of the cumulative execution time (x-axis, seconds) against actual execution time (y-axis, seconds) for each sample configuration in the STRUMPACK application, visited by each method, over 10 batches. We also display the actual cumulative time for each batch to finish 100 sequential samples in white (seconds).

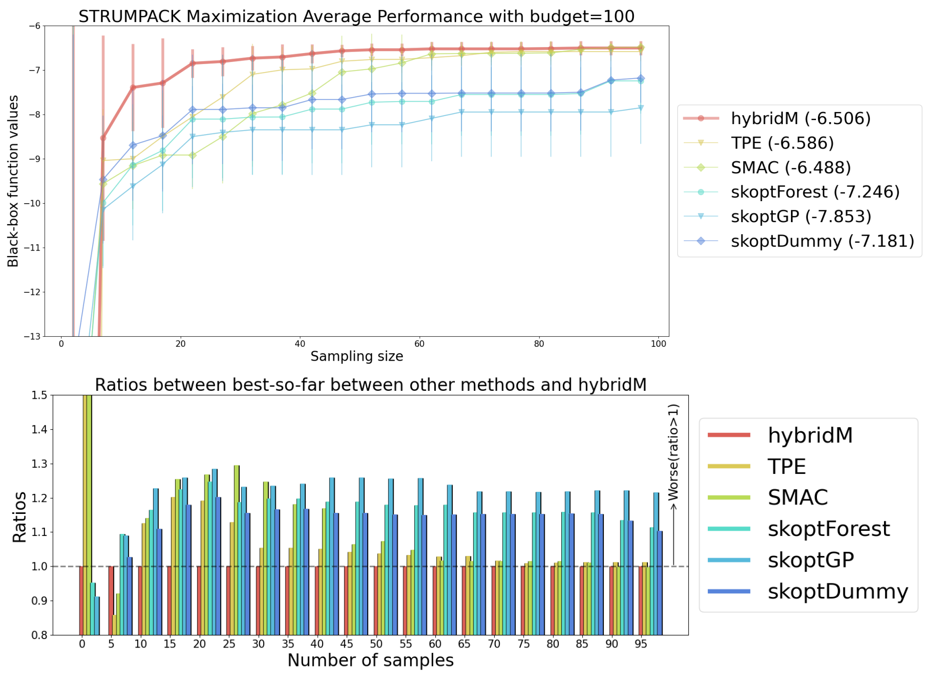


Figure 8: Comparison of performance between different methods on the STRUMPACK for Poisson3d with grid size 100. Note that we use negative optimal execution time in the upper panel to match the line plot format in the previous applications.

regarding the central limit theorem analog of the UCB reward, which is not fully Bayesian due to the inappropriate continuous Gaussian prior for the categorical space. Therefore, we aim to introduce a more suitable prior, potentially enhancing performance under a fully Bayesian framework. Future work for the proposed hybrid model for mixed-variable optimization includes fully Bayesian MCTS strategies, automatic tree level operations, and extensions to the multi-objective optimization context In addition, we propose to develop tree operations that can handle generic constraints for categorical variables as well.

Acknowledgements

We gratefully acknowledge the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. We used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231. We stored our code at `https://github.com/gptune/hybridMinimization` .

We sincerely thank Riley J. Murray and Rahul Jain for additional experiments for randomized Kaczmarz algorithms and constructive suggestions for our hybrid model on various applications.

We are grateful to the editor, the AE, and two anonymous reviewers for constructive comments and suggestions that have significantly improved the article.

References

Abdessalem, A. B., N. Dervilis, D. J. Wagg, and K. Worden (2017). Automatic kernel selection for gaussian processes regression with approximate bayesian computation and sequential monte carlo. *Frontiers in Built Environment 3*, 52.

Auer, P. (2002). Using Confidence Bounds for Exploitation-Exploration Trade-Offs. *Journal of Machine Learning Research 3*, 397–422.

Balaprakash, P., J. Dongarra, T. Gamblin, M. Hall, J. K. Hollingsworth, B. Norris, and R. Vuduc (2018). Autotuning in high-performance computing applications. *Proceedings of the IEEE 106*(11), 2068–2083.

Bergstra, J., R. Bardenet, Y. Bengio, and B. Kégl (2011). Algorithms for Hyper-Parameter Optimization. *Advances in neural information processing systems 24*.

Biau, G. and B. Cadre (2021). Optimization by Gradient Boosting. In *Advances in Contemporary Statistics and Econometrics*, pp. 23–44. Springer.

Bitzer, M., M. Meister, and C. Zimmer (2022). Structural kernel search via bayesian optimization and symbolical optimal transport. *Advances in Neural Information Processing Systems 35*, 39047–39058.

Bliek, L., S. Verwer, and M. de Weerdt (2021). Black-box Combinatorial Optimization using Models with Integer-valued Minima. *Annals of Mathematics and Artificial Intelligence 89*(7), 639–653.

Cerda, P., G. Varoquaux, and B. Kégl (2018). Similarity Encoding for Learning with Dirty Categorical Variables. *Machine Learning 107*(8), 1477–1494.

Chipman, H. A., E. I. George, and R. E. McCulloch (2010). BART: Bayesian Additive Regression Trees. *The Annals of Applied Statistics 4*(1), 266–298.

Cho, Y. and L. Saul (2009). Kernel methods for deep learning. *Advances in neural information processing systems 22*.

Deng, X., C. D. Lin, K.-W. Liu, and R. K. Rowe (2017). Additive Gaussian Process for Computer Models With Qualitative and Quantitative Factors. *Technometrics 59*(3), 283–292.

Deshwal, A., S. Belakaria, and J. R. Doppa (2021). Bayesian Optimization over Hybrid Spaces. *arXiv preprint arXiv:2106.04682*.

Elsken, T., J. H. Metzen, and F. Hutter (2019). Neural architecture search: A survey. *The Journal of Machine Learning Research 20*(1), 1997–2017.

Fouché, E., J. Komiyama, and K. Böhm (2019). Scaling multi-armed bandit algorithms. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1449–1459.

Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 1–67.

Garrido-Merchán, E. C. and D. Hernández-Lobato (2020). Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes . *Neurocomputing 380*, 20–35.

Ghysels, P., G. Chávez, L. Guo, C. Gorman, X. S. Li, Y. Liu, L. Rebrova, F.-H. Rouet, T. Mary, and J. Actor (2017). STRUMPACK.

Ghysels, P., X. S. Li, F.-H. Rouet, S. Williams, and A. Napov (2016). An Efficient Multicore Implementation of a Novel HSS-structured Multifrontal Solver using Randomized Sampling. *SIAM Journal on Scientific Computing 38*(5), S358–S384.

Gopakumar, S., S. Gupta, S. Rana, V. Nguyen, and S. Venkatesh (2018). Algorithmic Assurance: An Active Approach to Algorithmic Testing Using Bayesian Optimisation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 5470–5478.

Gramacy, R. B. (2020). *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences.*

Hannan, E. J. and B. G. Quinn (1979). The Determination of the Order of an Autoregression. *Journal of the Royal Statistical Society: Series B (Methodological) 41*(2), 190–195.

Harrison Jr, D. and D. L. Rubinfeld (1978). Hedonic Housing Prices and the Demand for Clean Air. *Journal of environmental economics and management 5*(1), 81–102.

Head, T., M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbatyi (2020). Scikit-Optimize/Scikit-Optimize. Zenodo.

Hermans, M. and B. Schrauwen (2012). Recurrent kernel machines: Computing with infinite echo state networks. *Neural Computation 24*(1), 104–133.

Hutter, F., H. H. Hoos, and K. Leyton-Brown (2011). Sequential Model-Based Optimization for General Algorithm Configuration. In *International Conference on Learning and Intelligent Optimization*, pp. 507–523. Springer.

Jones, D. R., M. Schonlau, and W. J. Welch (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization 13*, 455–492.

Karlsson, R., L. Bliek, S. Verwer, and M. de Weerdt (2020). Continuous Surrogate-based Optimization Algorithms are Well-suited for Expensive Discrete Problems. In *Benelux Conference on Artificial Intelligence*, pp. 48–63. Springer.

Kocsis, L. and C. Szepesvári (2006). Bandit Based Monte-Carlo Planning. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.), *Machine Learning: ECML 2006*, Volume 4212, pp. 282–293. Berlin, Heidelberg: Springer Berlin Heidelberg.

Liao, Y.-T., H. Luo, and A. Ma (2023). Efficient and robust bayesian selection of hyperparameters in dimension reduction for visualization. *arXiv preprint arXiv:2306.00357*.

Luo, H., J. W. Demmel, Y. Cho, X. S. Li, and Y. Liu (2021). Non-smooth Bayesian Optimization in Tuning Problems. *arXiv preprint arXiv:2109.07563*.

Luo, H. and M. T. Pratola (2022). Sharded Bayesian Additive Regression Trees. *arXiv:2306.00361*, 1–46.

Luo, H. and Y. Zhu (2023+). Optimism and Model Complexity Measure for Linear Models. *In preparation*.

Lykouris, T., V. Mirrokni, and R. P. Leme (2020). Bandits with adversarial scaling. In *International Conference on Machine Learning*, pp. 6511–6521. PMLR.

Malkomes, G., C. Schaff, and R. Garnett (2016). Bayesian Optimization for Automated Model Selection. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), *Advances in neural information processing systems*, Volume 29. Curran Associates, Inc.

Nguyen, D., S. Gupta, S. Rana, A. Shilton, and S. Venkatesh (2019). Bayesian Optimization for Categorical and Category-Specific Continuous Inputs. *arXiv:1911.12473 [cs, stat]*.

Oh, C., E. Gavves, and M. Welling (2021). Mixed Variable Bayesian Optimization with Frequency Modulated Kernels. *arXiv preprint arXiv:2102.12792*.

Olson, R. S. and J. H. Moore (2019). TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning. In F. Hutter, L. Kotthoff, and J. Vanschoren (Eds.), *Automated Machine Learning*, pp. 151–160. Cham: Springer International Publishing.

Rakotoarison, H., M. Schoenauer, and M. Sebag (2019). Automated Machine Learning with Monte-Carlo Tree Search. *arXiv:1906.00170 [cs, stat]*.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning.* Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press.

Ru, B., A. S. Alvi, V. Nguyen, M. A. Osborne, and S. J. Roberts (2020). Bayesian Optimisation over Multiple Continuous and Categorical Inputs. *arXiv:1906.08878 [cs, stat]*.

Sethuraman, J. and R. C. Tiwari (1982). Convergence of Dirichlet Measures and the Interpretation of Their Parameter. In *Statistical Decision Theory and Related Topics III*, pp. 305–315. Elsevier.

Shahriari, B., K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas (2016). Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE 104*(1), 148–175.

Sid-Lakhdar, W. M., M. M. Aznaveh, X. S. Li, and J. W. Demmel (2019, August). Multitask and Transfer Learning for Autotuning Exascale Applications. *arXiv:1908.05792 [cs, stat]*.

Sid-Lakhdar, W. M., Y. Cho, J. W. Demmel, H. Luo, X. S. Li, Y. Liu, and O. Marques (2020). GPTune User Guide.

Skelton, R. E., K. E. Whitaker, I. G. Momcheva, G. B. Brammer, P. G. Van Dokkum, I. Labbé, M. Franx, A. Van Der Wel, R. Bezanson, E. Da Cunha, et al. (2014). 3d-hst wfc3-selected photometric catalogs in the five candels/3d-hst fields: photometry, photometric redshifts, and stellar masses. *The Astrophysical Journal Supplement Series 214*(2), 24.

Snoek, J., H. Larochelle, and R. P. Adams (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *arXiv:1206.2944 [cs, stat]*.

Surjanovic, S. and D. Bingham (2022). Virtual library of simulation experiments: Test functions and datasets. Retrieved June 2, 2022, from `http://www.sfu.ca/~ssurjano`.

Swersky, K., D. Duvenaud, J. Snoek, F. Hutter, and M. A. Osborne (2014). Raiders of the lost architecture: Kernels for bayesian optimization in conditional parameter spaces. *arXiv preprint arXiv:1409.4011*.

Tesauro, G., V. Rajan, and R. Segal (2012). Bayesian Inference in Monte-Carlo Tree Search. *arXiv preprint arXiv:1203.3519*.

Willemsen, F.-J., R. van Nieuwpoort, and B. van Werkhoven (2021). Bayesian Optimization for Auto-Tuning GPU kernels. *arXiv:2111.14991 [cs, math]*.

Xiao, Q., A. Mandal, C. D. Lin, and X. Deng (2021). Ezgp: Easy-to-interpret gaussian process models for computer experiments with both quantitative and qualitative factors. *SIAM/ASA Journal on Uncertainty Quantification 9*(2), 333–353.

Ye, J. (1998). On measuring and correcting the effects of data mining and model selection. *Journal of the American Statistical Association 93*(441), 120–131.

Zeng, X. and G. Luo (2017). Progressive sampling-based bayesian optimization for efficient and automatic machine learning model selection. *Health information science and systems 5*, 1–21.

Zhang, Y., D. W. Apley, and W. Chen (2020a). Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports 10*(1), 4924.

Zhang, Y., S. Tao, W. Chen, and D. W. Apley (2020b). A latent variable approach to Gaussian process modeling with qualitative and quantitative factors. *Technometrics 62*(3), 291–302.

Supplementary Materials

A Bayesian Update Strategies

In contrast to deterministic draws by maximizing (1) in MCTS, the proposed Bayesian update strategy involves sampling a probability vector $\boldsymbol{p}$ from the Dirichlet distribution at each node, and then using this probability vector to define and draw from a multinomial distribution. This multinomial draw selects the next child node to visit and hence determines the categorical part along the path. In the update, parameters of the Dirichlet distribution are updated based on the rewards obtained from the search, allowing the search to dynamically adapt its strategy based on the results of previous searches.

The Dirichlet distribution is a multivariate generalization of the Beta distribution (Tesauro et al., 2012) and is particularly suitable for modeling categorical data or proportions. It is often used as a prior distribution in Bayesian statistics, especially in problems involving multinomial distributions due to their conjugacy. The Dirichlet distribution is parameterized by a vector $\boldsymbol{\alpha}$ of positive reals, which can be interpreted as pseudo-counts for the categories of the Multinomial distribution. Given a vector of parameters $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_K)$, a random vector $\boldsymbol{p} = (p_1, p_2, \ldots, p_K)$ follows a Dirichlet distribution if its probability density function is given by:

$$f(\boldsymbol{p} \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} p_i^{\alpha_i - 1} \tag{7}$$

where $B(\boldsymbol{\alpha})$ is the multivariate Beta function. The vector $\boldsymbol{p}$ represents the probabilities of choosing different K actions and the reward vector $\boldsymbol{r} = \boldsymbol{r}(a)$ (deterministically depending on action a as below) represents the K different rewards corresponding to different actions. The prior, likelihood and posterior can be expressed as follows:

$$\begin{aligned}
\mathbb{P}(\boldsymbol{p} \mid \boldsymbol{\alpha}) &\propto p_1^{\alpha_1 - 1} \cdots p_K{}^{\alpha_K - 1} \sim \mathrm{Dir}(\boldsymbol{\alpha}), & (8)\\
\mathbb{P}(a \mid \boldsymbol{p}, \boldsymbol{\alpha}) &\sim \mathrm{Multi}(\boldsymbol{p}), & (9)\\
\mathbb{P}(\boldsymbol{p} \mid \boldsymbol{r}(a), \boldsymbol{\alpha}) &= \frac{\mathbb{P}(\boldsymbol{r}(a), \boldsymbol{p} \mid \boldsymbol{\alpha})}{\mathbb{P}(\boldsymbol{r}(a) \mid \boldsymbol{\alpha})} = \frac{\mathbb{P}(a \mid \boldsymbol{p}, \boldsymbol{\alpha})\mathbb{P}(\boldsymbol{p} \mid \boldsymbol{\alpha})}{\mathbb{P}(\boldsymbol{r}(a) \mid \boldsymbol{\alpha})} \text{ via Bayes Theorem} & (10)\\
&\propto \mathbb{P}(a \mid \boldsymbol{p}, \boldsymbol{\alpha})\mathbb{P}(\boldsymbol{p} \mid \boldsymbol{\alpha}),\\
&\propto p_1^{r_1 + \alpha_1 - 1} \cdots p_K{}^{r_K + \alpha_K - 1} \propto \mathrm{Dir}(\boldsymbol{\alpha_s} + \boldsymbol{r}),
\end{aligned}$$

where $\mathrm{Dir}(\boldsymbol{\alpha})$ and $\mathrm{Multi}(\boldsymbol{p})$ are Dirichlet and multinomial distributions with respective parameters.

In the context of the tree search strategy, the Dirichlet distribution is used to model the uncertainty about the probabilities of selecting each child node. The parameters $\boldsymbol{\alpha}$ at each node associated with the Dirichlet distribution are updated each time a node is visited, based on the rewards obtained from the search. This allows the search to dynamically adapt its strategy based on the results of previous searches, favoring paths that have led to higher rewards in the past. We initialize the Dirichlet prior as a non-informative prior following the interpretation of Sethuraman and Tiwari (1982) that the base measure $\boldsymbol{\alpha}(\cdot)$ is the prior observed sample size and forcing $\boldsymbol{\alpha}$ to tend to zero means no prior information.

The novel Bayesian update strategy is as follows: We sample a K-vector $\boldsymbol{p}$ from the Dirichlet distribution $\text{Dir}(\boldsymbol{\alpha_s})$ at the parent node along the path $\boldsymbol{s} = (s_1, s_2, \cdots, s_i)$ where the node s_i has K possible categorical values as children, then a K-vector a from the multinomial distribution $\text{Multi}(\boldsymbol{p})$. Then we obtain a reward $r(a)$ from the playout vector a and update the reward distribution.

Next, we convert the reward $r(a)$ into a reward vector $\boldsymbol{r} = (r_1, r_2, \cdots, r_K)$ and update the Dirichlet distribution into $\text{Dir}(\boldsymbol{\alpha_s} + \boldsymbol{r})$. One primitive strategy that considers only exploitation is to set the reward vector $\boldsymbol{r}$ to binary vector with all entries equal to zero, except for the entry corresponding to the action a that has the largest average reward, which is set to a constant 1 (0-1 reward vector).

$$\boldsymbol{r} \coloneqq (r_1, r_2, \cdots, r_K) = (\delta_1, \delta_2, \cdots, \delta_K)\,, \tag{11}$$

where the notation δ_k is 1 if the average reward at children k is the largest among all of its siblings, otherwise 0. A strictly positive variant of this reward $(1+\delta_1, 1+\delta_2, \cdots, 1+\delta_K)$ can be used. However, this leads to empirically slow convergence, and we choose to include the denominator N_k to balance out the exploration-exploitation trade-off as shown in (1). Our reward vector is defined to be

$$\boldsymbol{r} \coloneqq (r_1, r_2, \cdots, r_K) = \left(\frac{1+\delta_1}{N_1}, \frac{1+\delta_2}{N_2}, \cdots, \frac{1+\delta_K}{N_K}\right), \tag{12}$$

where the denominator $N_k = n(s_1, s_2, \cdots, s_i)$ is the number of times the tree path $(s_1, s_2, \cdots, s_i)$ has been visited following the previous notations.

We have to point out that although the 0-1 reward vector seems to be a natural choice, our alternative reward vector (12) converges faster in almost all experiments. Besides the 0-1 reward, another natural reward function is to take a proportional distribution: the reward $r(a)$ is distributed among the entries of the reward vector $\boldsymbol{r}$ in proportion to some predefined weights. These weights could represent the prior belief about the importance or relevance of each action. This strategy allows for a more nuanced assignment of rewards to actions, which can be useful in complex environments where the contribution of each action to the reward is not binary, but varies in degree. Our alternative (12) can be considered as a modification of 0-1 reward, and we show its advantages via experiments but leave theoretical justification as future work.

In the view of reward function, this strategy effectively assigns the entire reward to the action that was taken, and no reward to the other actions. This strategy ensures that the search dynamically adapts its strategy based on the results of previous searches, favoring paths that have led to higher rewards in the past.

The UCTS strategy (denoted as hybridM) in Section 2.1 uses the formula (1) to guide the search in a deterministic manner. The UCB formula balances exploration (visiting fewer explored nodes) and exploitation (visiting nodes that have high average reward). The UCB value of a node is calculated based on the average reward of the node and a confidence interval that depends on the number of times the node and its parent have been visited. The node with the highest UCB value (1) is selected for the next visit. It uses a more deterministic approach (although we introduce ϵ-greedy) that balances exploration and exploitation based on a deterministic maximization.

The Bayesian Dirichlet-Multinomial strategy (denoted as hybridD) in Section A uses a Dirichlet-Multinomial conjugacy (10) to generate a sample of probabilities for each child node. This sample is then used to perform a multinomial draw to select the next node to visit. The parameters of the Dirichlet distribution are updated each time a node is visited, based on the rewards (12) obtained from the search. It also takes uncertainty into account but may converge slowly in situations where the reward structure is complex and the optimal path is not clear from the start.

B Candidate Kernels for the Mixed-variable GP

In what follows, we use a synthetic function to explain these steps:

$$f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \pi \cdot \mathbf{1}(\boldsymbol{x}_1 = 0) + e \cdot \mathbf{1}(\boldsymbol{x}_1 = 1) + 0.618 \cdot \mathbf{1}(\boldsymbol{x}_2 = 2) + \boldsymbol{x}_3 \tag{13}$$

$$y(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) + \epsilon \tag{14}$$

Here $\boldsymbol{x}_1 \in \{0, 1\}$, $\boldsymbol{x}_2 \in \{2, 3\}$, and $\boldsymbol{x}_3 \in \mathbb{R}$, hence $\boldsymbol{x}_{\text{cat}} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ and $\boldsymbol{x}_{\text{con}} = \boldsymbol{x}_3$. y is the desired mixed-variable GP surrogate for the objective function f with noise ϵ and the variable consisting of $d = 3$ coordinates in the notation of Algorithm 1. $X = X_{n_0,d} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{n_0}\}$ will consist of the input variables from n_0 initial samples, where $Y = Y_{n_0} = \{f(\boldsymbol{x}_1) + \epsilon, \cdots, f(\boldsymbol{x}_{n_0}) + \epsilon\}$ are corresponding noisy observations drawn from the objective function, following our assumptions.

Gaussian process (GP) modeling is mainly used for building surrogate models g with continuous variables in Bayesian optimization (Gramacy, 2020). We briefly describe the procedure in

BO and refer our readers to Shahriari et al. (2016) and Gramacy (2020) for formal details. In the procedure of optimization, we sequentially draw (expensive) samples from the black-box function f and update the surrogate model g with the samples drawn. For two sample locations, $\boldsymbol{u}, \boldsymbol{u}'$, their correlation is defined by a covariance kernel function $k(\boldsymbol{u}, \boldsymbol{u}') \in \mathbb{R}$ as we illustrate below. Supposing that the the surrogate GP model g approximates the black-box function f sufficiently well, we can find the true optimum $f_{\max} = \max_{\boldsymbol{x}} f(\boldsymbol{x})$ of the black-box function.

The sequential samples are selected based on the fitted surrogate model g; more precisely, we maximize the acquisition functions based on the surrogate model g to select the next sequential sample. In this paper, we only consider the expected improvement (EI) acquisition function (Shahriari et al., 2016).

When applied to mixed-variable functions, the quality of the GP surrogate depends on the covariance kernel that models both categorical and continuous parts simultaneously. However, existing kernels are developed mainly for continuous variables (Ru et al., 2020) and cannot be modified for categorical variables straightforwardly. Among the variety of different kernels for both continuous and categorical variables, several different observations have been made in the literature arguing whether regular continuous kernels or specialized kernels are suitable for categorical variables (Garrido-Merchán and Hernández-Lobato, 2020; Karlsson et al., 2020).

The form of the single mixed-variable GP surrogate y for the black-box function f is performance critical and highly problem dependent. For example, stationary kernels, both specialized (e.g. CoCaBO) (Ru et al., 2020) and the usual ones (e.g., Matern 5/2, without encoding) (Karlsson et al., 2020) have been demonstrated to show advantages in the context of some mixed-variable tuning, but worse in other scenarios. Besides the stationary kernels, we extend our candidate kernels under consideration to include both non-stationary kernels, which have been rarely used in the context of mixed-variable BO, and composition kernels as explained below using the example function (13).

Stationary kernels. This kind of kernel does not distinguish between continuous and encoded categorical variables, but is used mainly for continuous BO. Kernels in this family only depend on distances between variables, e.g.

$$k_{\text{Matern}}^{\nu,\ell}(\boldsymbol{u}, \boldsymbol{u}') = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{\ell} \|\boldsymbol{u} - \boldsymbol{u}'\| \right)^{\nu} B_{\nu} \left(\frac{\sqrt{2\nu}}{\ell} \|\boldsymbol{u} - \boldsymbol{u}'\| \right) \tag{15}$$

where B_ν is the modified Bessel function of order ν, Γ is the gamma function and l is a kernel parameter (Rasmussen and Williams, 2006). In the function (13), $\boldsymbol{u} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ if the kernel is used for categorical variables, and $\boldsymbol{u} = \boldsymbol{x}_3$ if the kernel is used for continuous variables. Note that for a general categorical variable, e.g., $\boldsymbol{u} = \{a, b, c\}$, we can use the default encoding $\boldsymbol{u} = \{0, 1, 2\}$ in (15). In this paper, we use $\nu = 5/2$, but different smoothness may affect the performance of BO (Luo et al., 2021).

Non-stationary kernels. For encoded categorical variables, we include the following non-stationary MLP (multi-layer perceptron) arc-sine kernel (a.k.a., arc-sine in (3.5) of Hermans and Schrauwen (2012))

$$k_{\text{MLP}}^{\sigma^2,\sigma_b^2,\sigma_w^2}(\boldsymbol{u},\boldsymbol{u}') = \sigma^2\frac{2}{\pi}\text{asin}\left(\frac{\sigma_w^2\boldsymbol{u}^T\boldsymbol{u}'+\sigma_b^2}{\sqrt{\sigma_w^2\boldsymbol{u}^T\boldsymbol{u}+\sigma_b^2+1}\sqrt{\sigma_w^2\boldsymbol{u}'^T\boldsymbol{u}'+\sigma_b^2+1}}\right), \tag{16}$$

Here σ^2, σ_b^2, σ_w^2 are kernel parameters, and $\boldsymbol{u} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ for the function (13). We argue that this is more suitable than the stationary overlapping kernel depending on the Hamming distance between the categorical variables $\boldsymbol{u}, \boldsymbol{u}'$ (CoCaBO (Ru et al., 2020)). The usual stationary kernels relies on the definition of metrics, however, it is hard to characterize even (linear) similarity between categories using regular distance metrics (Cerda et al., 2018). We hope that a non-stationary kernel along with suitable encoding could alleviate this problem and improve the quality of the model, and illustrate this via experiments.

As far as we know, non-stationary kernels are rarely studied for the mixed-variable in BO context, and we want to include such a kernel in our hybrid models. We choose the arc-sine kernel since it introduces inner products between encodings, and describes conditional embeddings and similarity between categories. The form of arc-sine kernel is also closely related to the arc-distance kernel proposed for conditional space BO (Swersky et al., 2014), and the arc-cosine kernel proposed for similarity learning (Cho and Saul, 2009). We select the arc-sine kernel for its implementation simplicity.

Composition kernels. We use the Matern 5/2 and MLP kernels to develop a few composition kernels for mixed-variables using candidate kernels, as shown in Table 3.

Summation kernel models the additive effect between the continuous and categorical variables:

$$k_{\text{sum}}(\boldsymbol{x},\boldsymbol{x}') = k_{\text{cat}}\left((\boldsymbol{x}_1,\boldsymbol{x}_2),(\boldsymbol{x}_1',\boldsymbol{x}_2')\right) + k_{\text{con}}\left(\boldsymbol{x}_3,\boldsymbol{x}_3'\right). \tag{17}$$

Product kernel models the interaction effect between continuous and categorical variables:

$$k_{\text{product}}(\boldsymbol{x},\boldsymbol{x}') = k_{\text{cat}}\left((\boldsymbol{x}_1,\boldsymbol{x}_2),(\boldsymbol{x}_1',\boldsymbol{x}_2')\right) \cdot k_{\text{con}}\left(\boldsymbol{x}_3,\boldsymbol{x}_3'\right). \tag{18}$$

A mixture of summation and product kernels (Ru et al., 2020) is commonly believed to take into account both additive and interaction effects:

$$k_{\text{mix}}(\boldsymbol{x},\boldsymbol{x}') = (1-\lambda)k_{\text{sum}}(\boldsymbol{x},\boldsymbol{x}') + \lambda k_{\text{product}}(\boldsymbol{x},\boldsymbol{x}'), \lambda \in [0,1]. \tag{19}$$

but we do not use this family of kernels.