Abstract

We systematically describe the problem of simultaneous surrogate modeling of mixed variables (i.e., continuous, integer and categorical variables) in the Bayesian optimization (BO) context. We provide a unified hybrid model using both Monte Carlo tree search (MCTS) and Gaussian processes (GP) that encompasses and generalizes multiple state-of-the-art mixed BO surrogates.

Based on the architecture, we propose applying a new dynamic model selection criterion among novel candidate families of covariance kernels, including non-stationary kernels and associated families. Different benchmark problems are studied and presented to support the superiority of our model, along with results highlighting the effectiveness of our method compared to most state-of-the-art mixed-variable methods in BO.

Keywords: Bayesian optimization, Gaussian processes, non-stationary covariance kernel, Monte Carlo tree search, categorical variables.

The authors 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/].
1 Introduction

In this paper, our goal is to handle black-box optimization problems with both continuous and categorical variables that arise in the tuning context, where the sampling budget is limited (e.g., less than 100 function evaluations are allowed). We want to find the maximum (or minimum) of a black-box function which is approximated by a surrogate model.

1.1 Types of Variables in Surrogates

Bayesian optimization (BO) has been proven to work successfully in optimizing (noisy) black-box functions without explicit analytical forms but only evaluations at specific locations (Hernández-Lobato et al., 2015; Shahriari et al., 2016; Snoek et al., 2012). At a high level, classical optimization algorithms with analytic functions ensure validity through the convergence of the sequence $f(x_n)$ based on the analytic gradient $\nabla f$ and the sequence of sequential samples $x_n$, while BO with black-box functions ensures validity through the convergence of surrogate models $g_n(x)$ for the unknown function $f$.

In the current problem setting, we are allowed to draw (possibly noisy) samples from $f$ in the form of location-observation pairs $(x, y) = (x, f(x))$. Then we use a Gaussian process (GP) surrogate model $g_n(x)$ fitted with sequential samples $\{x_1, \ldots, x_n\}$ of size $n$. GP surrogate (Gramacy, 2020; Luo et al., 2021) can be formulated as a regression problem $y(x) = g_n(x) + \epsilon$ where $\epsilon$ is the noise part, and we expect $g_n \approx f$ as a function when the sequential sample size $n$ (guided by acquisition maximization) is sufficiently large.

Most of the existing work from a modeling perspective focuses on continuous input domains for $x$ as the convergence of the function sequence $g_n(x)$ is difficult to control over a discrete domain. Furthermore, the acquisition functions used in BO for integer and categorical variables are usually different from the continuous case (Holmström et al., 2008; Luo et al., 2021). Integer and categorical variables do not have a natural generalization of numerical gradients when optimizing acquisition functions, which poses challenges from an optimization perspective (Ru et al., 2020). In some approaches (Head et al., 2020), random sampling among all possible categories is performed instead of maximization of acquisition when there are different types of variables involved, but sampling-based techniques deteri-
orate rapidly as the number of categories increases. The discrete gradient method shares an idea similar to Oh et al. (2019), which can be used for inner loop optimization in BO if some of the variables are categorical.

Naturally, we need better strategies to handle different types of variables (Garrido-Merchán and Hernández-Lobato 2020), since ordering and magnitude of the variables involved usually have an important influence on parameter estimation and quantifying the dependence between the variables of the black-box function.

Continuous variables (a.k.a. numerical variables, or real variables) have uncountably many values and a natural intrinsic ordering of these values, similar to real numbers $\mathbb{R}$. For example, people’s heights are measured in a continuous manner ranging from 0 to 3 meters. In addition to numerical gradients, the ordering and magnitude information carried by the GP covariance kernel and the statistics (e.g., mean or median) of the data are interpretable.

Integer variables (a.k.a. discrete variables, or ordinal variables) have multiple value categories and there is a natural intrinsic ordering on these categories, like integer numbers $\mathbb{Z}$. For example, human age could be coded as an integer variable with three categories (low, medium and high). Coding low as 0, medium as 1 and high as 2 makes more sense than coding low as 2, medium as 0 and high as 1 (breaking the natural ordering of ages). However, different encodings bring different covariance structures and parameter estimates. Also, the sample statistics (as well as normalization using sample statistics) would have rather different interpretations.

Categorical variables (a.k.a. nominal variables) also have multiple value categories, but there is no natural intrinsic ordering on the categories. For example, gender is a categorical variable that could have three (or more) categories (e.g., male, female, and nonbinary), without a natural ordering between categories. Usually, a nominal coding does not introduce ordering. If we encode male as 0, female as 1 and nonbinary as 2 then such a mapping is not different (or there is no preference for one over another) from male as 1, female as 2 and nonbinary as 0. In addition, the sample statistics, like means (as well as standardization), have no meaning. Therefore, while discrete gradients might be defined for integer variables, no analog of gradients can be defined for categorical variables, and
| Representation | Continuous var. | Integer var. | Categorical var. |
|----------------|-----------------|--------------|-----------------|
| Magnitude      | Yes             | Yes          | No              |
| Order          | Yes             | No           | Yes             |
| Encoding       | No              | Yes          | Yes             |
| Gradient       | continuous gradient | discrete gradient | No               |
| Surrogate      | GP              | GP/Tree      | Tree            |

Table 1: Comparison between different types of variables.

discontinuity may be induced (Luo et al., 2021).

The situation in which more than one type of variable is considered is called an optimization problem with *mixed variables*. We shall also point out that in the modeling context, researchers use “quantitative variables” referring to continuous (and integer variables), and “qualitative variables” referring to categorical (and integer variables). The empirical rule of thumb seems to be (Karlsson et al., 2020): When there is a relatively large number (depending on the application) of values for integer variables, practitioners tend to treat them as continuous variables with embedding $\mathbb{Z} \hookrightarrow \mathbb{R}$ (a.k.a. rounding-off encoding); when there are few values for integer variables, practitioners may consider these variables categorical. There exist some specialized surrogate-based black-box optimization models specifically designed for integer variables (Blick et al., 2016). However, its surrogate model may not belong to the GP family (e.g., its surrogate fitting uses another basis function family).

In the following, we use only the terms *continuous variables* and *categorical variables*. The integer variable is indeed different as shown in Table 1 but we follow the above rule of thumb in the modeling literature for subsequent discussions. The loss of such practice is that we may not be able to study discrete gradient methods (Bagirov et al., 2020) and miss some specialized encodings (Oh et al., 2019) designed for integer variables. However, as we shall see below, the hybrid model can differentiate integer variables if necessary.
1.2 Literature Review

When we perform sequential sampling in the BO, we build an acquisition function based on the GP surrogate model $g_n$ and maximize analytical acquisition functions based on the surrogate model (Wilson et al., 2018). The resulting maximizer is chosen to be the next sequential sample. From a modeling perspective, the surrogate model can be written additively as:

$$y(x) = \sum_{i=1}^{C} g_i(x_{\text{continuous}}, x_{\text{categorical}}) \mathbf{1}(x_{\text{categorical}} = i) + \epsilon$$

for categorical variables $x_{\text{categorical}}$ with $C$ categories corresponding to different combinations of categorical values (Zhang and Notz, 2015).

Example. We consider an additive model for continuous variables $x_{\text{continuous}} = x_3$ and categorical variables $x_{\text{categorical}} = (x_1, x_2)$ so that $(x_{\text{continuous}}, x_{\text{categorical}}) = (x_1, x_2, x_3)$:

$$f(x_1, x_2, x_3) = (\pi + 0.618 + x_3) \cdot \mathbf{1}(x_1 = 0, x_2 = 2) + (\pi + x_3) \cdot \mathbf{1}(x_1 = 0, x_2 \neq 2)$$

$$= \pi \cdot \mathbf{1}(x_1 = 0) + e \cdot \mathbf{1}(x_1 = 1) + 0.618 \cdot \mathbf{1}(x_2 = 2) + x_3$$

Therefore, there are $n_c = 2$ categorical variables $x_1 \in \{0, 1\}$ and $x_2 \in \{2, 3\}$, both of which have 2 categories, so $C = 4$, and their values have no meaning in order or magnitude. The only continuous variable is $x_3 \in \mathbb{R}$ in the black-box function $f$.

In practice, if there are only a few categories (small $C$), we may simply build independent surrogate models for each category as above. Supposing that there is an infinite sampling budget, then each independent surrogate model converges to the marginal black-box function for that combination of categorical variables. When there are generally many categories (large $C$), and the construction of independent surrogates becomes less efficient in using samples and ignores the interaction between continuous and categorical variables.

Encoding methods, including one-hot encoding (Snoek et al., 2012) and graph-based encoding (Karlsson et al., 2020), are still some of the most widely used methods for mixed variables. Assisted by ensemble construction and multi-layer combinations, Erickson et al.\footnote{We only consider the expected improvement (EI) acquisition function in the current paper.}
Klein et al. (2020) outperforms other automatic tuning methods in the neural network regression task of structured data. However, these encoding methods are usually quite problem-specific, and arbitrary encoding may introduce more problems than performance gain, even for encodings designed for categorical variables (Cerda et al., 2018; Garrido-Merchán and Hernández-Lobato, 2020). In input variable encoding, we turn to several lines of model methods for mixed-variable surrogate-based optimizations.

The first line involves the development of new acquisition functions in the GP surrogate model. Certain designs of acquisition functions aim to solve specific problems encountered in discrete or categorical space (Deshwal et al., 2021; Oh et al., 2021; Willemsen et al., 2021). The advantage of this line of methods is that the BO framework is intact and the GP surrogate is preserved.

The second line points to the redesign of surrogate models. Deng et al. (2017) assume independent additive GP components for different types of variables, while Bliek et al. (2021) replaces GP surrogates with different families of basis functions. Graphical surrogate models are proposed to better model the black-box function (Biau and Cadre, 2021; Head et al., 2020; Olson and Moore, 2019). Surrogate models with novel kernels capture the categorical variable using the density estimator (TPE, (Bergstra et al., 2011)) or random forests (SMAC, (Hutter et al., 2011)).

The third line is based on a (partially) exhaustive search with heuristic schemes over all possible combinations of categorical values (Gramacy and Ludkovski, 2015; Stoyanov, 2018). In this way, the BO problem can be viewed as a collection of optimizations when conditioned on categorical variables whose choices are done by a multiarmed bandit (MAB). This collection of methods (EXP3BO (Gopakumar et al., 2018), Bandit-BO (Nguyen et al., 2019), CoCaBO (Ru et al., 2020)) uses multiarmed bandits (e.g. EXP3 (Auer et al., 2002)) to choose the “optimal” ordinal and categorical variables. Then the model fits the remaining continuous variables using one standard GP surrogate that involves all of these variables.

In this paper, we follow the third line. Before concluding this section, we point out that no surrogate-based BO methods have explicitly considered the batch setting for continuous-categorical input, such as when a tuning variable can take values over a mixed domain like
\[0, 2] \cup \{3, 4, 5\}, which is related to Bayesian spike-and-slab models (Ročková and George, 2018) and may occur naturally in phase-transition simulations.

Following the discussion of the literature for surrogate-based online optimization methods that handle mixed variables above, section 2 will introduce the hybrid model. Hybrid models use the Monte Carlo tree search (MCTS) policy, as in section 2.1, to handle categorical variables. Section 2.2 explains which kernels we want to consider and how our model dynamically selects the kernel. Section 3 benchmarks hybrid models against other state-of-the-art mixed-variable models on synthetic functions and real applications.

2 Hybrid Models

After revisiting the problem of handling both continuous and categorical variables, we propose a surrogate model and explain how to execute optimization. Previously, Nguyen et al. (2019) adopted a multiarmed bandit approach and Ru et al. (2020) added dependence modeling. Rakotoarison et al. (2019) adopt a tree-like structure approach (i.e., MCTS) to search heuristically but completely ignores the dependencies. We propose to deploy the bandit strategy for categorical variables that would be used along with a tree-like structure with dynamic kernel selection.

In our hybrid models, we take the GP regression formulation to model both categorical and continuous variables; we use Monte Carlo tree search for categorical variables and acquisition maximization for continuous variables, which brings the best of two methods (Rakotoarison et al., 2019; Ru et al., 2020). For the search policy, we heuristically search the categorical space using MCTS. Conditioned on the categorical variables, we attempt several potential kernels for the GP surrogate model (on all variables). The best model would be chosen by maximizing a model selection criterion we detail in section 2.2 (e.g., negative log likelihood). Then, we search the continuous space by maximizing acquisition based on the best chosen GP model. For the update strategy, we still keep GP updated by adding both the continuous and categorical variables and its corresponding function values; this update strategy allows the GP surrogate to capture the correlation between continuous and categorical variables. Our contribution of the hybrid model can be summarized as follows:
Figure 1: Algorithmic representation for the update step (i.e., heuristic search for the next sequential sample) in the proposed hybrid model. Note that the fitting of GPs with different kernels can be parallelized rather than fitted sequentially.

- We propose a unifying model that can be used to realize several state-of-the-art mixed-variable models.

- We introduce dynamic kernel selection criteria among newly developed families of continuous-categorical kernels.

- We analyze various kinds of synthetic and real benchmarks under different measures for a collection of mixed-variable methods.

The name hybrid model is self-explanatory. The model construction which we propose takes a modeling approach; it provides a search policy and update strategy\(^2\) when the surrogate-based optimization problem involves both types of variables.

\(^2\)N.B. Although we know that both “policy” and “strategy” are arbitrarily used in the literature, we use the term (search) policy for the selection method when moving from root to leaf, and the term (update) strategy for the backpropagation method when moving from leaf to root.
2.1 Monte Carlo Tree Search

Our hybrid model architecture uses MCTS to generalize and unify several state-of-the-art mixed-variable models as shown in Table 2, especially Mosaic (Rakotoarison et al., 2019) and CoCaBO (Ru et al., 2020). This construction serves to unify existing models.

As criticized by Ru et al. (2020) (in section 5.2), MAB-based BO surrogate models cannot handle large $n_c$ (the number of categorical variables) and large $C$ (the total number of different combinations of categories in categorical variable). CoCaBO does not solve this problem, because it does not express searching orders. It also uses the same searching probabilities even if different categorical variables have different number of categories. However, the MCTS used by the hybrid model naturally handles this seamlessly by building the tree structure on the categories that needs to be searched, hence saving space.

We recall the basic notation of MCTS and that the decision space of a multiarmed bandit could be considered as a one-layer tree with leaves corresponding to a combination of categorical variables (Munos, 2014). Consider a path along a tree with $L$ layers, which can be represented as a string $s = (s_1, s_2, \ldots, s_L)$ of length $L$, where $s_1, s_2, \ldots, s_L$ represent the nodes on this path. MCTS (Kocsis and Szepesvári, 2006) presents a heuristic search policy on the tree that guarantees optimality in the sense of being close to the maximum of the reward function “on average”. After a search, at a leaf node, we condition on the

Figure 2: Schema for the proposed hybrid model for the function (3). The hybrid model consists of two parts. The categorical part is searched by the hierarchical structure of the tree; the continuous part is searched by GP acquisition. Both categorical and continuous parts are modeled by the GP surrogate.
path $s$ and play out to get a reward function value. This playout action $a$ at the leaf node at the end of the string $s$ would be evaluated using a reward function $r(a)$. Then, this information would be backpropagated through the tree according to the update strategy. Updated information will also be used to guide the next search.

For example, the upper confidence bound (UCB (Auer, 2002)) strategy would choose the node $s_i$ that maximizes the UCB function:

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

where $\tilde{r}(s_1, s_2, \cdots, s_i)$ is the average reward from all reward values by going through the path $(s_1, s_2, \cdots, s_{i-1})$ and taking the node $s_i$ to form $(s_1, s_2, \cdots, s_i)$; $C_{UCB}$ is a universal constant; $n(s)$ is the number of times we visit the path $s$. Heuristically, the term $\tilde{r}(s_1, s_2, \cdots, s_i)$ denotes the “expected reward at $(s_1, s_2, \cdots, s_i)$” as an exploitation term; the term $C \sqrt{\frac{\log n(s_1, s_2, \cdots, s_{i-1})}{n(s_1, s_2, \cdots, s_i)}}$ serves as an exploration term. We update $n(s)$ and $\tilde{r}(s)$ backward along the path for each interior node.

In terms of BO, we can treat the surrogates $g$ of the black-box function $f$ as the reward function. We adopt the notation $f(s, \cdot)$ which fixes categorical variables in $f$ corresponding to the combination represented by the path $s$. Previously, Rakotoarison et al. (2019) took UCB and constructed a GP surrogate model $g_s(\cdot)$ independently that approximates the black-box function $f(s, \cdot)$. We also introduce the $\varepsilon$-greedy search policy to enhance exploration in place of the deterministic UCB policy, with a probability $\varepsilon \in (0, 1)$, following Auer et al. (2002).

Bayesian strategies are provided in our implementation and explained in the appendix A, but we would consider only UCB for MCTS (a.k.a. UCTS) in this paper. The benefit of adopting a tree-like structure is that we can search the categorical space more efficiently.

---

3This includes the implementation for round-robin MAB and random MAB.
Table 2: Comparison between different mixed-variable models (and unification of hybrid models).

### 2.2 Kernel Selection

Our hybrid model introduces dynamic model selection to improve the performance in mixed-variable models. The kernel selection for online surrogate models, or more generally online model selection, is highly non-trivial. Here, our discussion focuses on choosing the kernel in GP surrogate models in BO.

Among the variety of different kernels for both continuous and categorical variables, different observations have been made in the literature and our pilot experiments. If there are few sequential samples or the encoding of categorical variables is appropriately scaled, separable kernels with specialized distance (e.g., CoCaBO) are observed to lead to better performance (Garrido-Merchán and Hernández-Lobato, 2020; Ru et al., 2020). If there are more sequential samples, it is observed that the GP surrogate with regular kernels (e.g., Matern 5/2, without encoding) outperforms specialized methods for integer variables (Karlsson et al., 2020). This is also observed in our mixed-variable experiments (see Figures 11 and 13 below).

We want to select a covariance kernel $k$ automatically as we build BO, and this covariance kernel may be different per iteration in sequential sampling. It has been pointed out by Rasmussen and Williams (2006) (c.f. 5.4) that model selection in the GP regression setting falls into two different categories: On the one hand, the model selection criterion based on the marginal likelihood of the GP model of sample size $n$ can be calculated with a com-
plexity of $O(n^2)$ based on the probability of the data conditioned on the assumed model. On the other hand, the leave-one-out CV criterion can be computed with a complexity of $O(n^3)$ and gives an estimate of the predictive log probability. General CV procedures lead to criteria that tend to be more robust against model mis specification. The online context in BO with GP brings us another challenge: How should we incorporate the potential improvement into kernel selection as well if we decide to use GP for the continuous variable?

In the mixed-variable BO context, Bayesian information criterion (BIC)

$$\text{BIC}_k = 2 \log P_k(y) - n_k \log n \quad (5)$$

where $n_k$ is the number of kernel parameters of kernel $k$ and $n$ is the number of samples in $y$. The BIC is suggested by [Malkomes et al. (2016)] for kernel selection in continuous variable BO. Highly parameterized kernels with good performance, such as diffusion [Kondor and Lafferty (2002)] and CaCoBO kernels, are seldom selected when both continuous and categorical variables are considered according to the penalty term for complexity in classic model selection criteria (i.e., AIC and BIC), with detailed comparisons in Appendix E.

The (negative) log marginal likelihood $\log P_k(y)$ of the surrogate model serves as a model selection criterion. Log likelihood indicates how well the surrogate model fits the data. The maximal acquisition function (e.g., expected improvement) of the model $A_k(y) := \max_x \text{EI}(x, P_k(y))$ can also be used in practice as a model selection criterion. The acquisition function indicates how much improvement we can get from the next sequential sample suggested by this model. Both criteria can be numerically unstable in practice, but the acquisition function is usually more unstable. A reasonable selection criterion for the surrogate covariance kernel $k$ can be expressed as:

$$C^\lambda_k = \lambda \frac{\log P_k(y)}{\max_\ell \log P_\ell(y)} + (1 - \lambda) \frac{A_k(y)}{\max_\ell A_\ell(y)}, \lambda \in [0, 1], \quad (6)$$

where both quantities are max-normalized to be comparable. Using another way of nor-
malization, we have alternative criteria through sum-normalization:

\[ S^\lambda_k = \lambda \frac{\log \mathbb{P}_k(\mathbf{y})}{\sum_{\ell} \log \mathbb{P}_\ell(\mathbf{y})} + (1 - \lambda) \frac{\mathbb{A}_k(\mathbf{y})}{\sum_{\ell} \mathbb{A}_\ell(\mathbf{y})}, \quad \lambda \in [0, 1]. \]  

(7)

However, the expression above exhibits sensitivity in practice due to the numerical issue introduced by fitting GP. To remedy the issue, we noticed that we can use the rank \( R_{\mathbb{P}}(k) \) of log likelihood \( \log \mathbb{P}_k(\mathbf{y}) \) and the rank \( R_{\mathbb{A}}(k) \) of acquisition \( \mathbb{A}_k(\mathbf{y}) \) (larger quantities correspond to higher ranks), both among all \( k \). Then, we can use the following rank-based model selection criterion.

\[ R^{1/2}_k = R_{\mathbb{P}}(k) + \frac{1}{2} R_{\mathbb{A}}(k), \]  

(8)

where the factor \( \frac{1}{2} \) is to put less emphasis on the acquisition function since it is usually more sensitive than log likelihood due to the fact that it incorporates uncertainty.

We focus exclusively on EI as our acquisition function. A model that has high improvement (to get maximum) but presents a serious lack-of-fit would not preferred by our criteria due to the first term. A model that fits very well (near the maximum) but falls short to produce an improvement over the observed maximum would not be preferred by our criteria due to the second term. We want to emphasize again that dynamically selecting kernels (or more generally, selecting surrogate models) is a novel strategy not studied before.

The closest research that is relevant to our treatment of online model selection is the adaptive model mixing by [Yang, 2001] in an offline context. Instead of performing model averaging, we select the model directly with selection criteria and do not perform data splitting.

**Example.** (Computation of \( R^{1/2}_k \)) 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 \( \mathbf{y} \). Then, we can compute the relevant quantities below for kernel selection purposes. We present two ways of normalization, yet their relative ranks remain the same.
Therefore, we prefer the best-fitted model and the best acquisition function, which is attained by a GP model with the kernel $k_1$.

In different applications, we might be able to adjust this weight of $1/2$ according to the total number of candidate kernels and prior knowledge. It is also possible to adaptively adjust the weight.

Taking $C^\lambda_k$ as an example, the log likelihood describes how well the “lack-of-fit” part and the acquisition function describes the “potentiality of improvement” part. As more samples accumulate, we might care less about “lack-of-fit” but more about “potentiality of improvement” since we know that the surrogate model converges well near optima, hence we might want to decrease $\lambda$ when more samples are available. To give an example,

$$R_{k,i}^a = R_P(k) + \frac{2 \cdot i}{n} R_A(k), i = 1, 2, \cdots, n,$$

where $n$ is the deterministic sampling budget and $i$ means the $i$-th step among all $n$ steps.

In the first $n/2$ steps, $i = 1, 2, \cdots, n/2$, the rank $R_P(k)$ dominates our criterion. In the last $n/2$ steps, $i = n/2 + 1, n/2 + 2, \cdots, n$, the rank $R_A(k)$ is the deciding factor. However, we point out that even such an adaptive criterion would ask for a choice of multiplier (i.e., 2 in the numerator). In short, this criterion is simpler and more flexible when applied to kernel selection. Our criteria consider the goodness-of-fit along with improvement and can
be applied to other acquisition functions as well.

The second term in (6) or (8) concerns the improvement gain from each of the candidate models. The weighted sum of the separate ranks in (8) differs from the classic model selection criteria, since we consider two different criteria simultaneously. Various model selection criteria are usually on different scales, so we do not know which criterion is the most suitable and how important they are respectively. Our approach is to convert two criteria (i.e., the log likelihood and acquisition function) into ranks and aggregate them to summarize them.

We consider two categories of kernels when using the GP surrogate to model the observed samples drawn from the black-box function. We use the function (3) as our example. In addition, all covariance kernels are set to have different scale parameters for different dimensions (i.e., Auto Relevance Determination, ARD).

**Regular kernels.** This kind of kernel is used not only in continuous BO but also in regular GP regressions for fixed samples. In a regular kernel, we do not distinguish between continuous and categorical variables

\[
k(x, x') = k_{\text{Matern}}^{\nu, \ell}((x_1, x_2, x_3), (x'_1, x'_2, x'_3)).
\]

\[
= \frac{1}{\Gamma(\nu)2^{\nu-1}} \left( \frac{\sqrt{2\nu}}{\ell} \| x - x' \| \right)^\nu B_{\nu} \left( \frac{\sqrt{2\nu}}{\ell} \| x - x' \| \right),
\]

\(\nu, \ell\) are kernel parameters.

where \(B_{\nu}\) is the modified Bessel function of order \(\nu\) and \(\Gamma\) is the gamma function (Rasmussen and Williams, 2006). In the function (3), the \(\| x - x' \| = \sum_{i=1}^{3} (x_i - x'_i)^2\). For regular kernels, we focus on the Matern family with different degrees of smoothness \(\nu > 0\). As pointed out by Luo et al. (2021), different smoothness would affect the performance of BO, and we consider three Matern kernels with different \(\nu\): Matern 1/2 (i.e., Exponential), Matern 5/2, Matern \(\infty\) (i.e., RBF).

**Separable kernels.** Below, we discuss different candidate kernels and write down the specific forms of kernels for the function (3).

- Summation form of a kernel models the additive effect between the continuous and
categorical variables. In a separable kernel, we can set one component kernel to be null, so that the corresponding part of the variable \( x \) is ignored, otherwise they model the correlation between continuous and categorical variables.

\[
k_{\text{sum}}(x, x') = k_{\text{categorical}}((x_1, x_2), (x'_1, x'_2)) + k_{\text{continuous}}(x_3, x'_3) .
\] (12)

- Product form of a kernel models the interaction effect between continuous and categorical variables.

\[
k_{\text{product}}(x, x') = k_{\text{categorical}}((x_1, x_2), (x'_1, x'_2)) \cdot k_{\text{continuous}}(x_3, x'_3) .
\] (13)

- Mixture of summation and product forms of kernels is commonly believed to take into account both additive and interaction effects.

\[
k_{\text{mix}}(x, x') = (1 - \lambda)k_{\text{sum}}(x, x') + \lambda k_{\text{product}}(x, x'), \lambda \in [0, 1].
\] (14)

The family of CoCaBO kernels with different weights is a special case of the mixture kernel, where the categorical kernel is chosen to be the overlapping kernel and the continuous kernel is chosen to be in a regular family. The weight \( \lambda \) can be fixed or estimated as a kernel parameter.

By default, we choose the (non-stationary) MLP kernel (a.k.a., arc-sine) as categorical kernel \( k_{\text{categorical}}(x, x') \) and argue that this is more suitable than the stationary overlapping kernel depending on the Hamming distance between the categorical parts \( u, u' \) in the variables \( x, x' \). The MLP kernel has the following expression:

\[
k_{\text{MLP}}^{\sigma^2, \sigma_b^2, \sigma_w^2}(u, u') = \sigma^2 \frac{2}{\pi} \sin \left( \frac{\sigma_w^2 u^T u' + \sigma_b^2}{\sqrt{\sigma_w^2 u^T u + \sigma_b^2 + 1} \sqrt{\sigma_w^2 u'^T u' + \sigma_b^2 + 1}} \right),
\] (15)

\( \sigma^2, \sigma_b^2, \sigma_w^2 \) are kernel parameters.

Our continuous kernel \( k_{\text{continuous}} \) is always chosen to be the Matern 5/2 kernel with ARD in mixture kernels. For separable kernels, we consider three forms of kernels (12), (13) and
Table 3: Default candidate kernels in hybrid models.

| $k$                                      | $k_{\text{categorical}}$ | $k_{\text{continuous}}$ |
|------------------------------------------|---------------------------|--------------------------|
| $k_{\text{categorical}} + k_{\text{continuous}}$ | $k_{\text{MLP}}^{\sigma^2, \sigma_b^2, \sigma_w^2}$ | $k_{\text{Matern}}^{5/2, \ell}$ |
| $k_{\text{categorical}} + k_{\text{continuous}}$ | $k_{\text{MLP}}^{5/2, \ell}$ | $k_{\text{Matern}}^{5/2, \ell}$ |
| $k_{\text{categorical}} + k_{\text{continuous}}$ | $k_{\text{MLP}}^{\sigma^2, \sigma_b^2, \sigma_w^2} + k_{\text{Matern}}^{5/2, \ell}$ | $k_{\text{Matern}}^{5/2, \ell}$ |
| $k_{\text{categorical}} \times k_{\text{continuous}}$ | $k_{\text{MLP}}^{\sigma^2, \sigma_b^2, \sigma_w^2}$ | $k_{\text{Matern}}^{5/2, \ell}$ |
| $k_{\text{categorical}} + k_{\text{continuous}} + k_{\text{categorical}} \times k_{\text{continuous}}$ | $k_{\text{MLP}}^{\sigma^2, \sigma_b^2, \sigma_w^2}$ | $k_{\text{Matern}}^{5/2, \ell}$ |

(14) with MLP and Matern for categorical and continuous parts of the variable $\mathbf{x}$. The non-stationary MLP categorical kernel is more flexible compared to a stationary categorical kernel in [Ru et al., 2020], and as far as we know, is the first application of non-stationary kernels in the mixed-variable context.

These two categories are by no means an exhaustive list of all reasonable kernels. In our experiments, we found several families of kernels that may be suitable for mixed-variable surrogates, including: the diffusion kernel family (Kondor and Lafferty, 2002), the pyramid match kernel (PMK) family (Grauman and Darrell, 2005), the arc kernel family (Hutter and Osborne, 2013), the imputation kernel family (Hutter and Osborne, 2013), the arc-cosine kernel family (Saul et al., 2016), and the indefinite conditional kernel family (Zaefferer, 2018). It is plausible to design a selection strategy for these kernels in a systematic way.

We consider the candidate kernels listed in Table 3. To choose among these candidate kernels for our surrogate model, we conduct model selection using the criteria (8), the detailed algorithm is in the appendix C. We want to emphasize again that the kernel selection problem we consider here is completely different from selection of surrogates for fixed dataset (Goel et al., 2007); we are allowed to draw more samples, because it is an online setting.

3 Experiments

We have already established our methodology where MCTS searches the categorical space, and GP searches and models the continuous space, in the setting of mixed-variable surrogate optimization. In this section, we provide a comprehensive benchmarking with refined
analysis for selected existing methods that are capable of handling mixed types of variables.

3.1 Synthetic Functions

We are going to compare our hybrid model 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.

Following our notational convention, we let $n_c$ be the number of categorical variables in the definition of the function. This quantity would mainly affect the depth of the tree structure when sequential order cannot be ignored. Let $C$ be the total number of possible combinations of categorical values for $n_c$ categorical variables. In the case of small $C$, we also provide roundrobin BO (Bergstra et al., 2011) results, since round-robin BO is clearly not favored in the large $C$ case. In the case of large $C$, we might want to pay attention to rounding-off encoding (i.e., by treating categorical variables as integers) kernels with GP, which is not suitable for small $C$, as discussed in section 1.2 and experimental results in section F.

The metric we use to express the computational budget (x-axis) is the number of function evaluations (or samples observed from the black-box function). The performance measure (y-axis) is the maximum objective black-box function value (or minimum in minimization problems) we observed at this sample size. For a single run, we can plot a curve for this “best-so-far” maximum value against the sample sizes explored. For multiple runs (with different random seeds), we display the point-wise means and standard deviations for multiple curves. In synthetic examples, we could also compute $\Delta f := \|f_{\text{max}} - y_{\text{max}}\|$ as a measure of performance. To exhibit the convergence rate of the surrogate models, for each example, we study the line plot of the current optimal function values against the sampling size. To exhibit the variation across different runs (with different random seeds), we

---

4We do not compare to Bandit-BO for the same reason pointed out by Ru et al. (2020) that EXP3BO is very similar to Bandit-BO (Nguyen et al., 2019).
repeat each experiment with different random seeds and display the distributional optima in Appendix D.

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 correlation between categories. Unfortunately, the existing literature seems to ignore this crucial difference in the choice of benchmarking function.

3.1.1 Categorical benchmarking

The first kind of benchmark functions are created by assigning different continuous-variable functions to different categories. This kind of function is represented by our example (3), func2C ($n_c = 2, C = 3 \times 5 = 15$, appendix [F.1]) and func3C ($n_c = 3, C = 3 \times 5 \times 4 = 60$, [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, which is the typical setting previous research focused on.

In Figure 3 (and 11 in Appendix D), we test with a benchmark function of the first
kind. We can observe that our hybrid model is among the best models, and enjoys a fast convergence rate. When we have the categorical benchmarking black-box functions, the hybrid model needs fewer 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. In this kind of benchmarking, the bandit strategy is quite suitable in exploring the categories compared to covariance models. From our empirical studies, the hybrid model and the one-hot GP (categorical is differentiated from continuous) are both 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 are 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 \cite{Malkomes2016}, Ackley-cC \cite{Nguyen2019} and GP sample path functions considered by Garrido-Merchán and Hernández-Lobato \cite{Garrido2020}. The correlation
between categories is induced by the original dependence between continuous variables.

The integer-like benchmarking function is obtained by restricting the continuous variables to integer values and considering these variables as categorical variables. For example, the well-known rosenbrock function [Surjanovic and Bingham] can be scaled and defined for an arbitrary dimensional domain as:

\[
f(x) = \frac{-1}{10000} \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], [-5, 5]^d \subset \mathbb{R}^d \rightarrow \mathbb{R}. \tag{16}
\]

For example, when \(d = 7\) we may keep the first 4 dimensions continuous. However, for the last 3 dimensions, instead of letting \(x_5, x_6, x_7\) 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 Figure 4 (and 12 in Appendix D), we test with a benchmark function of the second kind. We can observe that our hybrid model has a slow start but eventually reaches the actual optima as other surrogate models. The variances of all surrogate models considered are relatively low, but the convergence rate of the hybrid model might be improved by adopting a better MCTS search policy. This result echoes our rule of thumb mentioned at the end of section 1.1 and 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. From our empirical studies, not much difference in averaged performance is shown, but the convergence rate may be different for this kind of benchmarking, especially when the search policy is not well chosen.

### 3.1.3 Inactive benchmarking

The third kind of benchmark function contain one or more “inactive” categorical or continuous variables. In other words, some of the variables do not affect the function value
Figure 5: Comparison of performance between different methods on the function (17) over 20 batches. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. The actual maximum is 30.

at all. This sort of benchmarking is ignored by previous studies on categorical variables, but arises naturally in applications like screening, sensitivity analysis, and computer experiments (Linkletter et al., 2006; Welch et al., 1992).

This kind of synthetic function mainly aims at checking if the mixed model can handle some inactive variables, and avoid the redundant dimensions brought by these inactive variables. This kind of benchmarking is largely ignored in the mixed-variable surrogate research.

The 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). The function is composed of complicated vertical segments. We modify this function by appending several more inactive variables and yield a Friedman-8C function. There are 6 continuous variables and 8 categorical variables in this function, but the total number of combinations of categorical values is not too large because we restrict the $x_{10}, \cdots, x_{14}$ to
take only values in \{0, 1, 2, 3\} \ (n_c = 8, C = 3 \times 5 \times 3 \times 4 \times 4 \times 2 \times 2 = 11520) \\

\[ f(x) = 10 \sin(\pi x_1 x_2) \cdot 1(x_7 = 0) + 20(x_3 - 0.5)^2 \]
\[ + 10 x_4 \cdot 1(x_9 = 0) - 10 x_4 \cdot 1(x_9 = 1) + 5 x_4 \cdot 1(x_9 = 2) + 5 x_5 \]
\[ f : x \in [0, 1]^6 \times \{0, 1, 2\} \times \{0, 1, 2, 3\} \times \{0, 1, 2\} \times \{0, 1, 2, 3\}^3 \times \{0, 1\}^2 \to \mathbb{R}. \]

This function is also designed to include non-stationarity to fail the stationary kernel, since \( x_9 = 1 \) has equal distances to \( x_9 = 0 \) and 2, but the correlations between categories are clearly different.

From Figure 5 (and 13 in Appendix A), we test with a benchmark function of the third kind. We can observe that our hybrid model outperforms the other mixed-variable models both in convergence speed and final optima. In this situation, the advantage of bandit is enhanced by using a non-stationary MLP kernel. The combination of these two techniques allows us to focus more on active categories. Except for one outlier batch, all repeated batches suggest that the hybrid model reaches the actual maximum at reasonable rates. While it is possible to create covariance kernels that involve both stationary and non-stationary components in separable or convolution form as in Table 3, it is important to see how stationary kernels may be inappropriate in mixed parameter spaces. However, the selection between stationary and non-stationary kernels remains a difficult problem even in fixed dataset modeling. In our empirical studies, the hybrid model is always preferred when there exist inactive variables or a hierarchical space is needed. In addition, the bandit strategy can be treated as a special case of the hybrid model.

We can observe that under these 3 differently designed benchmark functions, the ordering of the performances by different surrogate-based tuning changes.

To sum up, our 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 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.
3.2 Scientific Applications

The MAB-based methods (CoCaBO, EXP3BO, roundrobinMAB, randomMAB) cannot be applied directly to an arbitrary problem without scaling the response value. The problem worsens due to the fact that, for expensive applications (e.g., HPC codes), we usually do not even have enough data to determine what kind of scaling should be adopted. Their EXP3 update strategy would simply cause numeric overflow due to the exponential part in the function. To avoid discussion of machine scaling (CoCaBO and other MAB methods would have overflow problem in updating the reward) for each problem, we compare the following methods (with their implementations indicated in brackets) for scientific applications (their specific implementations are indicated in the bracket): TPE (`hyperopt == 0.2.5`), SMAC (`smac == 1.0.1`), skopt (`scikit-optimize == 0.9.0 with scikit-learn == 1.0`) and our hybrid model.

Our main focus is on expensive black-box functions arising in scientific computing and machine learning 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 negligible compared to the evaluation time of black-box functions.

3.2.1 Hyper-parameter tuning for a feed-forward neural network

In this application, we study the typical AutoML tuning application of a feed-forward neural network with dynamic kernels. 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 choose the measure to be the mean square error (MSE) between the predictor and the observed response, and take the negative MSE 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.

We also notice that the different ranges of hyper-parameters would affect the tuning difficulty. For instance, if we restrict the activation function types to only: tanh, ReLU;
layers to only: $[4, 8] \cap \mathbb{Z}$, then TPE and SMAC would quickly attain the best optima after about 30 iterations. But within 30 iterations, the hybrid model always outperforms its competitors. In fact, for a wider range of hyper-parameter values in Table 4, Figures 6 and 14 indicate the better performance of hybrid model. And in this specific range, the hybrid model also has the best convergence rate and best optima. This echoes the observation of Rakotoarison et al. (2019) that the MCTS is better able to handle larger categorical spaces.

After the initial comparison, 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 the hybrid model, it suffices to prune these nodes (See Appendix B). After pruning, it is clear that the hybrid model converges faster and reaches better optima.

The result of this comparative study favors the hybrid model for tuning the regression network. For categorical space without any information, that is, the choice of one categorical parameter does not depend on the other; then the GP-based estimator, forest-based estimator, and hybrid model estimator all have their merits. The GP-based estimator has an advantage in the task whose objective is similar to the first kind of benchmarking. The forest-based estimator has advantages in a task whose objective is similar to the second kind of benchmarking. The hybrid model has advantages in a task whose objective is similar to the third kind of benchmarking. However, the MCTS (with pruning) seems to be the only natural way to handle constraints on categorical space.

We observe that it is usually the case that the sequential order of the layers would affect the performance. By default, we put the layer corresponding to fewer categories near the root (i.e., the first layer corresponds to the number of layers; the second layer corresponds to the activation for layers). In Figure 6, we also provide the mean performance when we reverse the default sequential order of the layers (ord_hybridM) and we combine and collapse some nodes corresponding to the number of layers = 3 and activation type = $\tanh$. 

25
| Type          | Hyper-parameter       | Values                                                                 |
|--------------|-----------------------|----------------------------------------------------------------------|
| Categorical  | Number of Layers      | $\{3, 4, 5\}$                                                       |
|              | Activation for Layers | $\{\text{tanh}, \text{ReLU}, \text{sigmoid}, \text{linear}\}$       |
| Continuous   | Layer Sizes           | $[4, 16] \cap \mathbb{Z}$                                           |
|              | Initial Learning Rate | $10^{-8, 0}$                                                        |
|              | 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.

Figure 6: Comparison of performance between different methods on the regression neural network for Boston Housing dataset over 10 batches. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. The theoretical maximum is 0.

(prune\_hybridM) (See details in Appendix B).

3.2.2 STRUMPACK for 3-dimensional Poisson problem

In this application, we want to use the STRUMPACK (Ghysels et al., 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 a 3-dimensional Poisson equation on a $100 \times 100 \times 100$ regular grid, leading to a sparse matrix of dimensions $1M \times 1M$. We consider 5 hyper-parameters with 3 categorical variables and 2 continuous variables as shown in Table 5. Each function

---

5 the number of layers includes the input/output layer but not the dropout layer.
6 all dense layers share the same activation function.
7 all dense layers share the same size.
evaluation requires 10 to 500 seconds using 8 Cori Haswell nodes. For each tuning algorithm, we execute 10 batches with 100 sequential samples due to computational time limit (see Figure 7).

In this experiment, the performance measure should be the optimal execution time of the STRUMPACK. However, we could measure the “tuning budget” in two ways. On the one hand, we can still measure the number of sequential samples taken so far as a machine-independent budget metric. On the other hand, we could measure the actual cumulative execution time (excluding the surrogate model fitting time, which is small compared to expensive application\(^8\)) during the tuning as a machine-dependent budget metric.

As a result, the hybrid method outperforms the rest of the surrogate models in terms of convergence rates and optimal configuration corresponding to the optimal execution time, and the hybrid model reaches the optimal parameter configuration among all runs. The possible explanation is that not all tuning variables are active under certain configurations for this application, like we observed in the tuning of (17) in section 3.1. Compared to simple models (skoptGP, skoptForest, skoptDummy) provided by skopt, we can see hybrid, SMAC and TPE present steady improvement and avoid being trapped at a plateau. Therefore, if the optimal execution time is the main concern and we limit the sample budget, then we may want to use the hybrid model, TPE or SMAC due to the optimality they exhibit.

The main time complexity falls to the execution of STRUMPACK, therefore, we provide the plot of optimal execution time against the average cumulative time taken by the STRUMPACK in Figure 7. We can see that the hybrid model attains the optimal execution time quickly compared to the other models. 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 8. 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 the hybrid model, TPE and SMAC spend most of the sequential samples with near-optimal execution time; while the simple skopt models explore sequential samples:\(^8\)Except for SMAC, which requires long fitting time for some pilots.
| 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                      | \(2^{[5,9]}\)                 |
|                   | Compression Tolerance                         | \(10^{-6}, 1\) \times \mathbb{Z}\) |

Table 5: Different parameters in the STRUMPACK for Poisson3d with grid size 100.

![Execution times > 12.0 seconds omitted.](image)

Figure 7: 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 9. In the line plot, the mean of the optima is computed for each method.

with worse execution times. Some badly chosen parameter configurations can take the STRUMPACK up to 400 seconds to complete, so the tuning cost is relatively high even for 100 sequential samples but still far much lower than an exhaustive search. Therefore, if the actual cumulative time (instead of the number of sequential samples) is the main concern, then we may want to use simple surrogate models or hybrid models over TPE and SMAC.

This experiment reveals an interesting phenomena: The simple models (e.g., GP and forest) may attain a reasonable optima 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. The hybrid model is unique in the sense that it would automatically select among surrogate models with different kernels as the sampling happens, which empirically balances between convergence rates and optimality.
4 Conclusion

4.1 Contribution

Motivated by the problem of incorporating both continuous and categorical variables, we choose the storage efficient MCTS algorithm for categorical variable space that solves the bandit problem of choosing an optimal combination of categorical variables for the next sequential sample. Our approach does follow the literature Ru et al. (2020) and Rakotoarison et al. (2019) in the problem formulation that the combination of categorical variables could be considered as a bandit problem; and the combination of continuous variables should be considered as a surrogate-based optimization problem. Our approach distinguishes itself by being an approach whose architecture encompasses several state-of-the-art models, incorporates dynamic model selection and features a novel family of kernels for mixed variables,
Figure 9: Comparison of performance between different methods on the STRUMPACK for Poisson3d with grid size 100. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. Note that we use negative optimal execution time to match the previous applications.

Based on diverse observations in previous work [Garrido-Merchán and Hernández-Lobato, 2020; Karlsson et al., 2020] in mixed-variable Bayesian optimization, we identified different kinds of benchmarking problems and provide up-to-date benchmarks comparing these state-of-the-art BO methods for mixed variables in the spirit of Eggensperger et al. (2013). The effectiveness of our surrogate model is supported by benchmarking, real-world and HPC centered applications in section 3.

We adopt a novel dynamic model selection strategy when there are both continuous and categorical variables in surrogate modeling with sequential samples, which is a specific case of model selection. Unlike the dynamic search adopted continuous variable kernel selection by Malkomes et al. (2016) and Fletcher et al. (2021), we design and select from
several candidate kernels that are suitable for mixed-variable BO, including the novel non-stationary categorical kernel. The fact that over-parameterized covariance kernels (for mixed variables) are not preferred by classical model selection criteria (i.e., AIC, BIC) is stated. As an alternative, we develop a new model selection criterion and its rank-based analog. Using this novel criterion, we improve the performance of the GP surrogate for mixed variables, as mentioned in the beginning of section 2.2.

4.2 Future Work

One of the guiding questions of this research is how to choose the correct model when we are allowed to perform a sequential sampling. The gain of model selection in a mixed-variable surrogate context is observed to be more obvious compared to a continuous variable situation. It would be of interest to see if a more sophisticated dynamic model selection strategy would improve the performance under mis-specified models, and a suitable selection criterion should incorporate the signal along with model complexity as well (Luo and Zhu 2021). It would be interesting if we could link this to an optimality criterion of experimental design in the sense of Srinivas et al. (2012). Considering from the design perspective, we may also want to dynamically change the acquisition function for the GP surrogate to decide the next continuous variable.

We want to point out that \( R_{1/2} \) does not seem to behave like the classic model selection criteria studied by Shao (1997), and it does not appear to present asymptotic normality either. Although our focus in the current paper is limited to small sampling sizes, it remains an interesting open question how \( R_{1/2}^{1/2} \) (or \( C_\lambda^{1/2} \) with \( \lambda \in (0, 1] \)) behaves.

It is of interest to consider multi-objective parallel optimization applications when mixed types of variables exist (e.g., in the neural network tuning task, we may want to minimize (or maximize its negative) both training and testing errors). Extensions to multi-objective optimization is a natural but highly non-trivial generalization of the current hybrid model. One way is to come up with a weighted objective function based on multiple objective functions; the other way is to introduce model averaging with multi-objectives in consideration. Fortunately, the specific architecture of the hybrid model makes both ways possible.
For higher dimensional optimization problems, the MCTS may require deeper structures, and the GP would encounter scalability issues (Luo et al., 2022). In the future, it would be very interesting to consider the high-dimensional BO when mixed variables exist.

Acknowledgment

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.

References

Aitchison, J. (1982). The statistical analysis of compositional data. *Journal of the Royal Statistical Society: Series B (Methodological)* 44(2), 139–160.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3(Nov), 397–422.

Auer, P., N. Cesa-Bianchi, and P. Fischer (2002). Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47(2/3), 235–256.

Auer, P., N. Cesa-Bianchi, Y. Freund, and R. E. Schapire (2002). The nonstochastic multiarmed bandit problem. *SIAM journal on computing* 32(1), 48–77.

Bagirov, A. M., M. Gaudioso, N. Karmitsa, M. M. Mäkelä, and S. Taheri (2020). Numerical nonsmooth optimization: State of the art algorithms.

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.

Bliik, L., H. R. Verstraete, M. Verhaegen, and S. Wahls (2016). Online optimization with costly and noisy measurements using random fourier expansions. *IEEE transactions on neural networks and learning systems* 29(1), 167–182.

Bliik, 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, March). BART: Bayesian additive regression trees. *The Annals of Applied Statistics* **4**(1), 266–298.

Deng, X., C. D. Liu, K.-W. Liu, and R. K. Rowe (2017, July). 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*.

Diaconis, P. and D. Freedman (1990). On the uniform consistency of Bayes estimates for multinomial probabilities. *The Annals of Statistics*, 1317–1327.

Eggensperger, K., M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, K. Leyton-Brown, et al. (2013). Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS Workshop on Bayesian Optimization in Theory and Practice*, Volume 10.

Erickson, N., J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola (2020). Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*.

Fletcher, T., A. Bundy, and K. Nuamah (2021). Gpy-abcd: A configurable automatic bayesian covariance discovery implementation. In *8th ICML Workshop on Automated Machine Learning (AutoML)*.

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.

Gelly, S. and D. Silver (2011). Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence* **175**(11), 1856–1875.

Ghysels, P., G. Chávez, L. Guo, C. Gorman, X. S. Li, Y. Liu, L. Rebrova, F.-H. Rouet, T. Mary, and J. Actor (2016–). 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.

Goel, T., R. T. Haftka, W. Shyy, and N. V. Queipo (2007). Ensemble of surrogates. *Structural and Multidisciplinary Optimization* **33**(3), 199–216.
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.

Görür, D. and C. E. Rasmussen (2010). Dirichlet process gaussian mixture models: Choice of the base distribution. Journal of Computer Science and Technology 25(4), 653–664.

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

Gramacy, R. B. and M. Ludkovski (2015). Sequential Design for Optimal Stopping Problems. SIAM Journal on Financial Mathematics 6(1), 748–775.

Grauman, K. and T. Darrell (2005). The pyramid match kernel: Discriminative classification with sets of image features. In Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1, Beijing, China, pp. 1458–1465 Vol. 2. IEEE.

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, September). Scikit-optimize/scikit-optimize. Zenodo.

Hernández-Lobato, J. M., M. A. Gelbart, M. W. Hoffman, R. P. Adams, and Z. Ghahramani (2015). Predictive Entropy Search for Bayesian Optimization with Unknown Constraints. arXiv:1502.05312 [stat].

Hintze, J. L. and R. D. Nelson (1998). Violin plots: a box plot-density trace synergism. The American Statistician 52(2), 181–184.

Holmström, K., N.-H. Quittineh, and M. M. Edvall (2008). An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. Optimization and Engineering 9(4), 311–339.

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.

Hutter, F. and M. A. Osborne (2013). A kernel for hierarchical parameter spaces. arXiv preprint arXiv:1310.5738.

Karlsson, R., L. Blied, 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.
Klein, A., L. C. Tiao, T. Lienart, C. Archambeau, and M. Seeger (2020). Model-based asynchronous hyperparameter and neural architecture search. *arXiv preprint arXiv:2003.10865*.

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.

Kondor, R. I. and J. Lafferty (2002). Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, Volume 2002, pp. 315–322.

Linkletter, C., D. Bingham, N. Hengartner, D. Higdon, and K. Q. Ye (2006). Variable Selection for Gaussian Process Models in Computer Experiments. *Technometrics 48*(4), 478–490.

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., G. Nattino, and M. T. Pratola (2022). Sparse additive gaussian process regression. *Journal of Machine Learning Research 23*(61), 1–34.

Luo, H. and Y. Zhu (2021+). Optimism and complexity measure. *In preparation*.

Lwin, T. and J. Maritz (1989). Empirical Bayes approach to multiparameter estimation: With special reference to multinomial distribution. *Annals of the Institute of Statistical Mathematics 41*(1), 81–99.

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.

Mittal, G., C. Liu, N. Karianakis, V. Fragoso, M. Chen, and Y. Fu (2020). HyperSTAR: Task-aware hyperparameters for deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8736–8745.

Munos, R. (2014). From bandits to Monte-Carlo Tree Search: The optimistic principle applied to optimization and planning.

Nguyen, D., S. Gupta, S. Rana, A. Shilton, and S. Venkatesh (2019, November). 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*.
Oh, C., J. M. Tomczak, E. Gavves, and M. Welling (2019, October). Combinatorial Bayesian Optimization using the Graph Cartesian Product. arXiv:1902.00448 [cs, stat].

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.

Pitman, J. (1996). Some developments of the Blackwell-MacQueen urn scheme. In Institute of Mathematical Statistics Lecture Notes - Monograph Series, pp. 245–267. Hayward, CA: Institute of Mathematical Statistics.

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.

Roˇckov´a, V. and E. I. George (2018). The Spike-and-Slab LASSO. Journal of the American Statistical Association 113(521), 431–444.

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].

Saul, A. D., J. Hensman, A. Vehtari, and N. D. Lawrence (2016). Chained gaussian processes. In Artificial Intelligence and Statistics, pp. 1431–1440. PMLR.

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.

Shao, J. (1997). An asymptotic theory for linear model selection. Statistica sinica, 221–242.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. (2017). Mastering the game of go without human knowledge. nature 550(7676), 354–359.

Silverman, J. D., K. Roche, Z. C. Holmes, L. A. David, and S. Mukherjee (2019). Bayesian Multinomial Logistic Normal Models through Marginally Latent Matrix-T Processes. arXiv:1903.11695 [stat].

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

In this appendix, we provide a collection of possible Bayesian update strategies for the MCTS part of the hybrid model.

Bayesian formulation (Thompson sampling) shares a similar idea as the random search policy, by assuming a policy-strategy distribution at each node. In his pioneering work,
Tesauro et al. (2012) uses a beta-binomial conjugate model for 0-1 reward distributions where the reward is either \( r = 1 \) (win) or \( r = 0 \) (loss). Drawing a sample as the action \( a \) from a beta distribution, this action is then treated as the success probability. The parent beta distribution parameters \( \alpha_s, \beta_s \) are updated to \( r + \alpha_s, 1 - r + \beta_s \) respectively. Tesauro et al. (2012) suggest to propagate posteriors up the tree along the path \( s \) either numerically or by a moment-matched Gaussian distribution approximating the extreme distributions among all possible nodes.

A.1 Conjugate prior: Dirichlet Update

One of the most important differences between game theoretic MCTS and surrogate-based optimization MCTS is the reward function at leaf nodes. Gopakumar et al. (2018) and Ru et al. (2020) target at the maximum and adopt the adversarial bandit setting implicitly by using EXP3 search policy, but both suffer if there are a lot of combinations of categorical values. Neither of these approaches model the dependence between categories in a principled way. Rakotoarison et al. (2019) treat the empirical mean of the black-box function as the reward function and build stochastic surrogates at the leaves of MCTS, but it is nullifying the usual assumption of a deterministic reward function in MCTS. While this practice may converge to the actual function with sufficient sequential samples, it casts doubt on the finite-time behavior.

Consider a vector \( \mathbf{r} \) of rewards for an action \( a \) in a multiarmed bandit, we could compare the reward \( r(a) \) with the rewards of the other possible but unplayed actions. Taking all these possible rewards together, we can form a vector of reward for the update strategy. The deterministic update strategy can be considered as a special case of \( \mathbf{r} = (0, \cdots, \xi(a), \cdots, 0) \) with only the reward for played action \( a \) being updated.

Most differently from section 2.1, our search policy below is a random policy. Suppose that there are \( K (K \geq 1) \) different actions at a certain node\(^9\), then a natural generalization to the above model is the Dirichlet-multinomial conjugate model with the base measure

\(^9\)N.B. The number \( K \) can depend on the number of layers \( L \) or its parent node or even both. This \( K \) should be considered as a hyper-parameter. For simplicity, we assume this to be a fixed constant, resulting a \( K \)-ary hierarchical tree structure. We use \( K \geq 2 \) example to illustrate hereafter.
\( \alpha_s = (\alpha_s(1), \cdots, \alpha_s(K)) \in \mathbb{R}^{K}_+ \). For the choice of base measures in the Dirichlet prior distribution, unless otherwise is known, we recommend a non-informative prior with \( \alpha \propto \epsilon 1 \) with a small \( \epsilon > 0 \). This recommendation follows Sethuraman and Tiwari (1982)'s interpretation that the base measure \( \alpha(\cdot) \) is the prior observed sample size and forcing \( \alpha \) tend to zero means no prior information. We do not attempt to discuss the choice of base measure in our context, but it is possible to model the dependence using base measure (Görür and Rasmussen 2010).

The playout strategy \( a \) is a probability vector \( a = (a_1, \cdots, a_K) \in [0, 1] \) putting on \( K \) different playout actions.

\[
\begin{align*}
\mathbb{P}(a \mid s) &\propto a_1^{\alpha_s(1)-1} \cdots a_K^{\alpha_s(K)-1} \sim \text{Dir}(\alpha_s) \\
\mathbb{P}(r \mid a, s) &\propto a_1^{r_1} \cdots a_K^{r_K} \sim \text{Multi}(a) \\
\mathbb{P}(s \mid r, a) &\propto \mathbb{P}(r \mid a, s) \mathbb{P}(a \mid s) \\
&\propto a_1^{r_1+\alpha_s(1)-1} \cdots a_K^{r_K+\alpha_s(K)-1} \\
&\sim \text{Dir}(\alpha_s + r)
\end{align*}
\]

Note that the reward vector \( r = (r_1, \cdots, r_K) \in \{0, 1\} \) for the playout \( a \) is also in form of a binary vector in \( \mathbb{R}^K_+ \), which shall be interpreted as one outcome for this playout \( a \). A naive Dirichlet strategy is composing a vector \( r = e_k \) if the \( k \)-th node is selected, which cannot guarantee convergence. A maximum-preferred Dirichlet strategy is composing a vector \( e_k^* \) if \( k^* = \arg \max_{k=1, \cdots, K} \mu_k(s a_k) \), the \( k^* \)-th surrogate has the largest sample maximum (or posterior mean). Although we write down the likelihood for one path, the path search at time \( t \) would depends only the searched paths, \( \mathbb{P}(s(T) \mid t < T) = \prod_{t \leq T} \mathbb{P}(s(t) \mid s(t'), r(t'), a(t'), t' < t) \). This update strategy is similar to the Simultaneous Optimistic Optimization (SOO) algorithm in Munos (2014), but it only searches one node at a time.

In our Dirichlet update, both leaf and interior nodes are selected probabilistically and not dependent on the \( \mu, \sigma^2 \). Therefore, one playout does not only update the reward distribution at a leaf node but also update the Dirichlet distribution of its parent node.
through Bayes’ theorem. The random policy and Bayesian update can be described as follows:

1. Sample a $K$-vector $\eta$ from the Dirichlet distribution $\text{Dir}(\alpha_s)$ at the parent node.

2. Sample a $K$-vector $v$ from the multinomial distribution $\text{Multi}(\eta)$.

3. Obtain a reward $r(a)$ from the playout leaf and update the reward distribution.

4. Convert the reward $r(a)$ into an update vector $r$ and update the Dirichlet distribution into $\text{Dir}(\alpha_s + r)$.

For sufficiently large $n(s)$ our maximum-preferred Dirichlet update is asymptotic optimal. To see this clearly, we first assume that the reward distributions at leaf nodes are constants that are not subject to updates.

**Proposition 1.** (Dirichlet update convergence) Suppose that the reward function is fixed and the Dirichlet distribution $\text{Dir}(\alpha_s)$, $\alpha_s(\cdot) > 0$ is supported on $\{1, \cdots, K\}$, $K < \infty$ with the update vector $e_{k^*}$, $k^* = \arg\max_{k=1,\cdots,K} \mu_k$, i.e., a vector with all zero entries except that the entry corresponding to the $\mu^* = \max_{k=1,\cdots,K} \mu_k$ is one. Then the maximum preferred Dirichlet update would choose $a_{k^*}$ with probability 1 as the number of visits along $s$, $n(s)$ tends to infinity.

**Proof.** Along the path $s$ we suppose that it has been visited and updated $n(s)$ times, each leaf node $a_k$, $k = 1, \cdots, K$ has been visited and updated $n(a_k)$ times. With this update strategy, the underlying multinomial distribution would have a probability mass function defined on $\{1, \cdots, K\}$ with probability 1 on $k^*$ where $\mu_{k^*} = \mu^* = \max_{k=1,\cdots,K} \mu_k$ and 0 on other values. By the consistency result on Dirichlet-multinomial conjugacy (Diaconis and Freedman, 1990; Pitman, 1996) and the assumption of strictly positive base measure $\alpha_s(\cdot) > 0$, we can assert that the posterior Dirichlet would have limiting base measure concentrated on $k^*$, leading to $\lim_{n(s)\to\infty} \frac{n_{a_k}}{n_s} = \{k = k^*\}$ almost surely.
For the \((n(s) + 1)\)-st visit of the Dirichlet update with probability
\[
P(v = e_k) = \mathbb{E}(1 \{v = e_k\}) = \mathbb{E}_{\text{Dir}(\alpha_s)}(\mathbb{E}_{\text{Multi}(\eta)}(1 \{v = e_k\} | \eta))
\]
\[
= \mathbb{E}_{\text{Dir}(\alpha_s)}(\eta_k) = \frac{n(a_k) + \alpha_s(k)}{n(s) + \sum_{k=1}^{K} \alpha_s(k)} = \frac{n(s) \cdot \frac{n(a_k)}{n(s)} + \left(\sum_{k=1}^{K} \alpha_s(k)\right) \cdot \frac{\alpha_s(k)}{\sum_{k=1}^{K} \alpha_s(k)}}{n(s) + \sum_{k=1}^{K} \alpha_s(k)},
\]
we decide to play out at the leaf node \(a_k\). Therefore, the leaf node \(a_k^*\) would be chosen with probability \(\xi_s = \frac{n(a_k^*) + \alpha_s(k^*)}{n(s) + \sum_{k=1}^{K} \alpha_s(k)}\) and the rest nodes would be chosen with probability \(1 - \xi_s\). From assumption, \(\lim_{n(s) \to \infty} \xi_s = \lim_{n(s) \to \infty} \frac{n(a_k^*) + \alpha_s(k^*)}{n(s) + \sum_{k=1}^{K} \alpha_s(k)} = O\left(\frac{n(a_k^*)}{n(s)}\right)\).

On-policy convergence like Theorem 2 in Tesauro et al. (2012) can be elicited by using this proposition in place of Lemma 1 in the same paper.

Although both \(\varepsilon\)-greedy randomized policy and maximum-preferred Dirichlet update would converge to a distribution that searches \(a_k^*\) with probability 1, these two sequences of probabilities approach 1 differently.

### A.2 Non-conjugate prior: Aitchson Update

From the discussion above, we can see that it is possible to introduce more complex structures than trees as we studied above, which allows us to characterize the dependence between categories. To see clearly how dependence may be modeled between nodes, we digress and introduce another update strategy. Unlike the dependence between continuous variables handled smoothly through the choice of covariance kernels in GP surrogates, categorical variables cannot be modeled easily since both ordering and magnitude information for categorical variables are inaccessible (See section (1.1)). However, the following update strategy allows us to introduce dependence between node search probabilities via a positive definite matrix \(\Sigma\).

This introduces another major challenge in Taking Dirichlet update in the previous section, we are actually introducing negative correlations between the leaf nodes and hence the combination of categorical variables associated. Aitchison (1982) generalized Dirichlet distribution to Aitchison distribution class (a.k.a. logistic-normal class). Aitchison distri-
bution class contains distributions than Dirichlet with richer dependent structure between
the probabilities for a multinomial distribution.

We provide a formulation of Aitchison update (with prior only on mean, assuming $\Sigma$
known for simplicity) below but point out that this introduces far more parameters than
the base measures in the Dirichlet update; and the convergence of Aitchison update likely
uses more samples compared to the simple Dirichlet update.

$$
\mathbb{P}(\mathbf{a} | \mathbf{s}) \propto \exp \left( - (\mathbf{a} - \mathbf{\mu}(\mathbf{s}))^T \Sigma (\mathbf{a} - \mathbf{\mu}(\mathbf{s})) \right) \sim N_K(\mathbf{a} | \mathbf{\mu}(\mathbf{s}))
$$

$$
\mathbf{a} = (a_1, \cdots, a_{K-1}, a_K),
$$

$$
T(\mathbf{a}) = \left( \frac{\exp(a_1)}{1 + \sum_{k=1}^{K-1} \exp(a_k)}, \cdots, \frac{\exp(a_{K-1})}{1 + \sum_{k=1}^{K-1} \exp(a_k)}, \frac{1}{1 + \sum_{k=1}^{K-1} \exp(a_k)} \right)
$$

(19)

$$
\mathbb{P}(\mathbf{r} | \mathbf{a}, \mathbf{s}) = \mathbb{P}(\mathbf{r} | \mathbf{a}) \propto T(\mathbf{a})_1 \cdots T(\mathbf{a})_K \sim \text{Multi}(T(\mathbf{a}))
$$

$$
\mathbb{P}(\mathbf{s} | \mathbf{r}, \mathbf{a}) \propto \mathbb{P}(\mathbf{r} | \mathbf{a}, \mathbf{s}) \mathbb{P}(\mathbf{a} | \mathbf{s})
$$

Unlike the Dirichlet update, we do not have Dirichlet-multinomial conjugacy to preserve
the form of posterior. To update the parent parameters $\mathbf{\mu}_s$ we need either an approximation
to the posterior integral or a Metropolis-Hasting sampling step. With the Aitchison update,
each leaf node closely relates to a computationally efficient multinomial logistic (MLN)
model derived by Silverman et al. (2019). It would be of interest to understand if the
specific empirical Bayes prior in this model (Lwin and Maritz, 1989) can be leveraged
further.

For one path $\mathbf{s}$ along the tree, if the parameter $\mathbf{\mu}(\mathbf{s}) = (\mu(a_1), \cdots, \mu(a_K))$ is the poste-
rior mean maxima of each leaf node GP surrogate, then $T(\mathbf{a})$ is a probability vector that
determines the chance of selecting each leaf. If we only choose the leaf node with MAP as
our playout, then this is equivalent to the deterministic policy $\pi$ used in (4) of Rakotoarison
et al. (2019) Silver et al. (2017).
Figure 10: Schema for node pruning (left) and layer collapse (right) based on the tree displayed in Figure 2.

B Layer Operations

Below, we briefly discuss how the hybrid model admits the layer operations, to organize the structure of categorical parameter space and reflect the constraint information better. Note that the layer operations we discuss below are static, that is, the structure of the tree-like structure is fixed before sequential sampling. Dynamic operations during sequential sampling are relevant but not treated in this paper.

Node pruning removes the infeasible combinations (e.g., due to the constraint of the problem or apriori knowledge that certain configurations cannot behave well) of categorical variables so that we do not have to search these combinations. When node pruning is performed, the number of layers usually remains the same, but the number of leaf nodes decreases.

Layer collapse operation shrinks the configuration search space by combining two layers into one layer in the tree-like structure. When a layer collapse is performed, the number of layers decreases, but the number of leaf nodes may increase. The main purpose of layer collapse is to adjust sequential ordering of categorical variables when sampling.

For categorical variables, 1-layer multiarmed bandits or 1-layer trees do not distinguish the sequential order between categorical variables but consider different combinations of categorical variables to be arms. When we collapse two layers, the search along the tree is shortened, since we only have one layer. When all the layers are combined into one layer, we would have a 1-layer tree. If we still treat the leaf nodes of this 1-layer tree as arms of
the bandit and conduct the search policy, we would have a 1-layer bandit. The following Table 2 shows how hybrid models can include some of the aforementioned surrogate models for mixed variables to be its special cases.

There may exist a *sequential ordering* between categorical variables, which is different from the ordering between the values of categorical variables. Tree structures express sequential ordering in terms of layer orders, and the tree structure can also be used to impose constraints on categorical variables. For example, if $x_1 = 1, x_2 = 2$ is not a feasible combination of categorical variables, instead of modifying the acquisition functions or encoding, we simply drop the corresponding leaf to prevent searching that combination of categorical variables.

At the first few layers, the selection of each branch would be important since it directly decides the later search path; at deeper layers, as the number of branches increases, we need to sample efficiently to avoid bad combinations of categorical variables and backpropagate to parents. When there is a sequential ordering between categorical variables (e.g., $x_1$ determines types of algorithms first, $x_2$ determines the algorithm-specific hyper-parameters, then we have to choose $x_1$ first, and then $x_2$), the sequential ordering should be reflected by the tree structure. Applications of sequential ordering between categorical variables include neural network tuning (Mittal et al., 2020; Rakotoarison et al., 2019), chess games (Gelly and Silver, 2011) and (randomized) algorithm tuning.

When there is no sequential ordering between categorical variables (e.g., $x_1$ represents gender, $x_2$ represents eye color, so we can choose $x_1$ value first or choose $x_2$ value first), then we can collapse the layers. Actually, collapsing is recommended whenever possible, since different orderings of layers may actually affect the search performance. Applications without sequential ordering between categorical variables include general black-box optimization (Ru et al., 2020) and architecture search (Gopakumar et al., 2018; White et al., 2020). It is also possible that some categorical variables have sequential ordering but the rest do not. In this case, partial collapse is desired. For instance, we need to choose which type of neural network to fit as the first categorical value; and then we need to choose both the number of layers and the types of activation functions. There is no sequential order
between these last two categorical variables.

C Selection Algorithm for the Covariance Kernel

We present the algorithm for the kernel selection using our novel criteria described in section 2.2 and Figure 1. When we predict based on the fitted MCTS-GP hybrid surrogate model,

The MCTS handles the heuristic search of the categorical part $X_{categorical}^{\ast}$ for the $X^{\ast}$ (or the categorical part of the $X_{new}$). The layer operations on the tree structure used by the MCTS not only allow us to incorporate apriori information into the tree, but also guide the heuristic search. The GP handles the heuristic search of the continuous part $X_{continuous}^{\ast}$ for the $X^{\ast}$. (or the categorical part of the $X_{new}$). The conditional acquisition function (conditioned on categorical part) would only search the continuous variable part, where the gradient information is available. The GP with the surrogate (and prediction) of all parts of $X^{\ast}$ (or $X_{new}$) but its kernel is dynamically selected.

The core algorithm is listed below, while our implementation also supports randomized search and fully Bayesian MCTS as detailed in appendix A.
Data: \( X_{n_0,d} \) (data matrix consisting of pilot samples in \( H^d \))

Input: \( \text{maxSampleSize} \) (the maximal number of evaluations of \( f \)), \( \text{batchSize} \) (the number of samples that can be observed from each leaf visit), \( L_k \) the list of covariance kernels we want to consider.

Result: One tree structure with searching information, one GP surrogate model with trained covariance kernel.

\( \text{EI} (x, g) \) is an acquisition function based on the surrogate model \( g \).

\begin{verbatim}
1 begin
2     Initialize \( X = X_{n_0,d} \)
3     Train the MCTS structure with the categorical part of pilot samples \( X_{n_0,d} \)
4     Train the GP surrogate with the full pilot samples \( X_{n_0,d} \)
5     while \( i \leq \text{maxSampleSize}/\text{batchSize} \) do
6         /* This outer loop uses MCTS to observe further samples for the categorical part of the next location \( X_{\text{categorical}}^* \). */
7         Find the best leaf and extract its corresponding combination of categorical variables as the proposed categorical part \( X_{\text{categorical}}^* \) through the MCTS (with prescribed search policy.)
8         while \( j \leq \text{batchSize} \) do
9             /* This inner loop uses GP to observe further samples for the continuous part of the next location \( X_{\text{continuous}}^* \). */
10            for \( k \) in \( L_k \) do
11                Train the GP surrogate \( g_k \) with data \( X, Y \) (including both continuous and categorical part of the \( X \)) with the covariance kernel \( k \).
12            Compute the model selection criteria for this GP model \( g_k \).
13        Find the kernel \( k^* \) that maximizes the model selection criteria.
14        /* The best kernel \( k^* \) may be different for each iteration. */
15        \( X_{\text{continuous}}^* = \arg \max_{z=(X_{\text{categorical}}^*, \cdot) \in H^d} \text{EI}(z, g_k^*(z)) \)
16        /* Fixing the categorical part of the variable as \( X_{\text{categorical}}^* \)
17        search the continuous part using \( z=(X_{\text{categorical}}^*, \cdot) \). */
18        \( X_{\text{next}} = (X_{\text{categorical}}^*, X_{\text{continuous}}^*) \)
19        Append \( X_{\text{next}} \) and its function values \( f(X_{\text{next}}) \) to the \( X \) and \( Y \) respectively.
20        Append \( X_{\text{next}} \) and its function values \( f(X_{\text{next}}) \) to the leaf history respectively.
21    end
22    Back-propagate from the best leaf through all its parents.
23 end
\end{verbatim}

Algorithm 1: Hybrid MCTS-GP model algorithm with model(kernel) selection
Figure 11: Averaged comparison between different methods on a scaled version of the function \(\text{func3C}\) in (Ru et al., 2020) over 20 batches. In the box plot, the cross indicates the mean of optima; and the tick indicates median of optima; in the violin plot, the color shades indicate estimated density of optima.

D Boxplot and Violin Plot Summaries

In each of these Figures 11 to 15, we summarize the optimal black-box function values using the boxplot (Tukey et al., 1977) and the violin plot (Hintze and Nelson, 1998). The box plot emphasizes the mean, the median and the quantiles with outlier values; while the violin plot shows the distributional properties of the optimal values for different methods.

E Effect of Model Selection Criteria

In this section, we provide experiments for the hybrid model under different model selection criteria, including AIC (equivalent to Mallow’s Cp), BIC, and log likelihood alone. Based on different criteria, we also exhibit how frequently each of the candidate kernels mentioned in section 2 are chosen. This set of experiments show that novel model selection criteria are needed.

Then, we exhibit the performance of GP surrogate for each of the candidate kernels without model selection. This set of experiments show that different kernels dynamically
Figure 12: Averaged comparison between different methods on scaled version of the 7-dimensional discrete rosenbrock function in Malkomes et al. (2016) over 20 batches. In the box plot, the cross indicates the mean of optima; and the tick indicates median of optima; in the violin plot, the color shades indicate estimated density of optima.

Figure 13: Averaged comparison between different methods on the function (17) over 20 batches. In the box plot, the cross indicates the mean of optima; and the tick indicates median of optima; in the violin plot, the color shades indicate estimated density of optima.
Figure 14: Averaged comparison between different methods on the regression neural network for Boston Housing dataset over 10 batches. In the box plot, the cross indicates the mean of optima; and the tick indicates median of optima; in the violin plot, the color shades indicate estimated density of optima.

Figure 15: Averaged comparison between different methods on the STRUMPACK for Poisson3d with grid size 100 over 10 batches. In the box plot, the cross indicates the mean of optima; and the tick indicates median of optima; in the violin plot, the color shades indicate estimated density of optima.
selected in the sequential sampling do bring us performance improvement, echoing the findings in Malkomes et al. (2016).

E.1 Classic Model Selection Criteria

In Figure (16), we show how different selection criteria may affect the performance of the GP surrogate models. Here, we compare the performance of GP surrogate with dynamic kernels selected by BIC, AIC, and log likelihoods for each surrogate model. In addition, we provide the performance of another selection criterion, Hannan–Quinn information criterion (Hannan and Quinn, 1979),

\[
\text{HQC}_k = 2 \log P_k(y) - 2n_k \log (\log n)
\]  

(20)

whose behavior is similar to using the acquisition function (i.e., expected improvement function in this paper) as a selection behavior. Both HQC and acquisition function, when serving as a model selection criterion, show better performance compared to BIC and AIC. However, their convergence rate is clearly not as good as our custom selection criterion \( R_{1/2} \).

We suspected that such a behavior is caused by a better balance between log likelihood and the sample size \( n \). The balance between log likelihood and acquisition function is discussed in the derivation of \( R_{a,i} \) in 9 as well.

Compared to the results in Figure 5, we can conclude that a model selection criterion in an online setting seems to be a deciding factor for the sequential sampling, hence the result of the black-box optimization.

Model selection criteria have little influence on the convergence rate (a dynamic selection seems to speed up the convergence) but affects the quality of optima.

E.2 Fixed Kernel Experiments

We can compare GP surrogate model with and without kernel selection in each iteration. In this set of experiments, we fix the covariance kernel throughout the sequential sampling scheme. This result echoes the ones shown in Malkomes et al. (2016) and indicates that
Figure 16: Comparison of performance between hybrid methods with different selection criteria on the function (17) over 10 batches. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. The actual maximum is 30.

while different selection criteria lead to different optima as in Figure 16, the fixed kernels are different in terms of convergence rates.

Dynamic selection of kernels has an influence both on the convergence rate (a dynamic selection seems to speed up the convergence) and the quality of optima. This is most obvious as shown by the benchmarking functions of the second and the third kind (Figure 19 and Figure 18), yet not obvious in the benchmarking functions of the first kind (). This is due to the fact that the second kind of benchmarking is not really providing discontinuities as the other two kinds. For the benchmarking function of the first kind, it is not surprising for this specific example (Figure 17) that the improvement of kernel selection is not obvious since the Matern 5/2 kernel on the categorical part can describe the correlation between categories well so that the hybrid method is not far away from the best kernels ($k_{\text{continuous}} = \text{MLP} + \text{Matern52}$, $k_{\text{categorical}} = \text{Matern52}$).
Figure 17: Comparison of performance between hybrid method with the selection criterion and fixed kernel GP surrogates on a scaled version of the function func3C in [Ru et al., 2020] over 10 batches. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. The actual maximum is 55.

Figure 18: Comparison of performance between hybrid method with the selection criterion and fixed kernel GP surrogates on a scaled version of the discrete rosenbrock function of 7 dimensions (4 continuous variables in \([-5, 5]\); 3 categorical variables \([-5, -4, \ldots, 4, 5]\)) in Malkomes et al. (2016) over 20 batches. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. The actual maximum is 0.
Figure 19: Comparison of performance between different but fixed kernels of GP surrogate with the selection criterion 8 and fixed kernel GP surrogates on the function (17) over 10 batches. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. The actual maximum is 30.

### F Benchmarking on Additional Functions

#### F.1 2-Category Function

The following scaled version of func2C has one fewer category compared to the func3C and relatively few combinations of categories. In this scaled version of the function, we can expect (since only two categories exist) and observe that the SMAC and GP with one-hot encoding outperforms the rest method. Our hybrid method comes the third and shows quicker convergence than SMAC. From Figures 20 and 21 the hybrid model is slightly worse than the one-hot encoded GP in this simple example due to the fact that one-hot encoding GP is essentially fitting different surrogates for two categories. The rest methods do not enjoy the fast convergence exhibited by the hybrid method.

When compared to the Figure 3 and 11 this simpler function indicates that not all advanced methods can outperform the simpler method (i.e., GP with one-hot encoding) when applied onto the benchmarking function of the first kind. The scale of the function seems important in this scaled version of func2C since some kernels are optimized for
Figure 20: Comparison of performance between different methods on a scaled version of the function func2C in (Ru et al., 2020) over 20 batches. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. The actual maximum is 25.

small scales. In addition, the EXPBO and CoCaBO need caution when there are a lot of categorical encoding, since when the categorical variable has a high absolute value, the exponential update strategy may cause numerical overflow. Most methods do not enjoy the fast convergence exhibited by the hybrid method.

F.2 Batch Size Effect

Unlike the usual practice in the mixed-variable model, we found that the batch strategy is not attractive when model selection is in play. In fact, under a limit budget, when we have to handle a high-dimensional space, larger batches may deteriorate the performance as shown in Figure 22 (The label hybridM2 corresponds to batch size 2, 50 searches over the tree; the label hybridM4 corresponds to batch size 4, 25 searches over the tree). Considering that there are usually a large number of combinations of categories, we would recommend batch size 1 by default.

While it is observed in CoCaBO (Ru et al., 2020) that batch size could have little effect on the averaged performance, its supporting experimental evidence is limited to the first kind of benchmarking functions. In general, we would not recommend any large batch size unless the sampling budget is relatively large and parallelization of the algorithm is desired.
Figure 21: Averaged comparison between different methods on a scaled version of the function func2C in (Ru et al., 2020) over 20 batches. In the box plot, the cross indicates the mean of optima; and the tick indicates median of optima; in the violin plot, the color shades indicate estimated density of optima.

Figure 22: Comparison of performance between hybrid methods with different batch sizes on the function (17) over 20 batches. In the line plot, the mean of the optima is computed for each method; the standard deviation is indicated by the vertical segments. The actual maximum is 30.