

MOCA-HESP: Meta High-dimensional Bayesian Optimization for Combinatorial and Mixed Spaces via Hyper-ellipsoid Partitioning

Lam Ngo^{a,*}, Huong Ha^a, Jeffrey Chan^a and Hongyu Zhang^b

^aRMIT University, Australia

^bChongqing University, China

Abstract. High-dimensional Bayesian Optimization (BO) has attracted significant attention in recent research. However, existing methods have mainly focused on optimizing in continuous domains, while combinatorial (ordinal and categorical) and mixed domains still remain challenging. In this paper, we first propose MOCA-HESP, a novel high-dimensional BO method for combinatorial and mixed variables. The key idea is to leverage the hyper-ellipsoid space partitioning (HESP) technique with different categorical encoders to work with high-dimensional, combinatorial and mixed spaces, while adaptively selecting the optimal encoders for HESP using a multi-armed bandit technique. Our method, MOCA-HESP, is designed as a *meta-algorithm* such that it can incorporate other combinatorial and mixed BO optimizers to further enhance the optimizers' performance. Finally, we develop three practical BO methods by integrating MOCA-HESP with state-of-the-art BO optimizers for combinatorial and mixed variables: standard BO, CASMOPOLITAN, and Bounce. Our experimental results on various synthetic and real-world benchmarks show that our methods outperform existing baselines. Our code implementation can be found at <https://github.com/LamNgo1/moca-hesp>.

1 Introduction

In recent years, high-dimensional BO has emerged as a crucial area of research, as many expensive black-box optimization problems can have hundreds of dimensions. Applications of high-dimensional BO include but are not limited to hyperparameter tuning of machine learning models [41, 10, 48], neural architecture search [21, 40], drug discovery [22, 13], supply chain management [19, 49] and optimal system design [9].

Various research works have tackled the high-dimensional BO problem for continuous spaces [10, 34, 42, 50, 30, 31]. However, in many real-world scenarios, objective functions may include combinatorial (ordinal and/or categorical) and mixed variables [19, 32, 9, 22, 35]. This presents significant challenges, as BO methods that are tailored for continuous variables cannot be directly applied to solve problems involving these combinatorial and mixed variables [39, 46, 35, 7]. There are multiple reasons. First, categorical variables lack a natural ordering or rank, preventing standard kernels from capturing the relationships between categorical variables and other variables, thus degrading the surrogate model's predictive

accuracy. Second, since ordinal and categorical spaces are discrete, gradient-based optimizers are not effective for optimizing the acquisition function. Third, conventional approaches such as using one-hot encoding to transform categorical variables into numerical values are not scalable as they can significantly increase the problem dimensionality, affecting the computational costs. Although there exist research works that tackle the problem of BO over high-dimensional combinatorial and mixed search spaces [32, 39, 46, 5, 35], due to the aforementioned challenges, achieving promising performance for these types of inputs is still an open question.

In this paper, we propose a novel high-dimensional BO *meta-algorithm* for combinatorial and mixed variables. To address high dimensionality, we leverage the hyper-ellipsoid space partitioning technique (HESP) [30] to define local regions with a high probability of containing the global optimum, within which BO can be performed. We propose a novel HESP technique designed for combinatorial and mixed problems, which integrates various encoding methods to transform combinatorial variables into continuous representations and employs a multi-armed bandit (MAB) strategy to adaptively select the most effective encoder. Our proposed method, *Meta-Algorithm for High-dimensional Ordinal, Categorical and Mixed Bayesian Optimization via Hyper-Ellipsoid Space Partitioning* (MOCA-HESP), is designed as a *meta-algorithm* [47, 42, 30] that can incorporate various BO optimizers to solve high-dimensional BO problems with combinatorial and mixed variables.

Furthermore, we develop three novel algorithms MOCA-HESP-BO, MOCA-HESP-Casmo and MOCA-HESP-Bounce, corresponding to the cases when we incorporate MOCA-HESP with state-of-the-art BO methods for high-dimensional combinatorial and mixed variables: standard BO, CASMOPOLITAN [46], and Bounce [35], respectively. For MOCA-HESP-BO, we integrate MOCA-HESP with the standard BO method, using it as the optimizer within MOCA-HESP. For MOCA-HESP-Casmo, we propose a novel technique to integrate CASMOPOLITAN's core feature - the local region adaptation mechanism - into MOCA-HESP and define local regions satisfied both the Mahalanobis and the Hamming distance criteria. For MOCA-HESP-Bounce, we propose a novel technique to incorporate Bounce's core feature - the subspace embedding technique - into MOCA-HESP's local regions, allowing low-dimensional optimization. Notably, the MOCA-HESP meta-algorithm can flexibly adapt different settings, e.g., input data encoding, used by various BO optimizers, making it compatible with existing and future BO opti-

* Corresponding Author. Email: s3962378@student.rmit.edu.au

mizers. Our experimental results on various synthetic and real-world combinatorial and mixed benchmark problems show that the MOCA-HESP methods outperform the respective BO optimizers and other baselines, demonstrating the effectiveness and efficiency of MOCA-HESP. We summarize our contributions as follows:

- We propose a novel meta-algorithm, namely MOCA-HESP, for solving BO problems involving high-dimensional combinatorial and mixed variables. We develop a novel HESP strategy that leverages categorical encoders and an adaptive encoder selection mechanism to effectively transform combinatorial variables into numerical representations. To the best of our knowledge, MOCA-HESP is the *first meta-algorithm* for BO in high-dimensional combinatorial and mixed spaces.
- We develop three practical methods, MOCA-HESP-BO, MOCA-HESP-Casmo, MOCA-HESP-Bounce, by proposing novel strategies to incorporate MOCA-HESP with state-of-the-art BO methods for high-dimensional combinatorial and mixed variables.
- We conduct extensive experiments and analysis on synthetic and real-world combinatorial and mixed benchmark problems against the baselines, validating the efficiency of our proposed algorithm.

2 Related Work

High-dimensional BO with Combinatorial and Mixed Variables.

One popular approach is to enhance the capacity of surrogate models in modeling objective functions with these types of variables. For example, TPE [4] and SMAC [20] propose to use different surrogate models such as Tree Parzen Structure and Random Forest, which can directly model combinatorial and mixed variables. Garrido-Merchán and Hernández-Lobato [11] propose to modify the kernel by rounding the kernel input to better capture flat regions in combinatorial and mixed objective functions. COMBO [32] tackles the problem by constructing a combinatorial graph, where each vertex represents a possible combination of categorical variables, and proposing a graph kernel to capture the interactions between categorical variables within this graph. Moss et al. [28] propose the String kernel, which models the categorical variables as strings, hence performing the optimization in a string space. CoCaBO [39] introduces the Overlapped kernel, which combines continuous and categorical kernels via a weighted average of addition and multiplication to capture the interaction between continuous and categorical variables, enhancing the modeling of the objective function for categorical and mixed variables. Based on this, CASMOPOLITAN [46] proposes the Transformed Overlapped kernel, improving the expressiveness of the model by learning different GP lengthscales for categorical kernels. BODi [7] employs a dictionary-based embedding to transform combinatorial variables to continuous values, enabling the use of standard continuous GP kernels for modeling combinatorial and mixed inputs. However, Papenmeier et al. [35] showed that BODi’s embedding tends to favor generating zero-sequence solutions, which may not be robust in all scenarios. Additionally, BODi does not generalize well to mixed-variable problems beyond binary-continuous settings.

In addition to improving surrogate models, various research works aim to optimize acquisition functions for combinatorial and mixed variables more effectively. BOCS [3] employs Simulated Annealing, which performs random walks in the neighborhood of a data point, moving to the next point based on acquisition function improvements and probabilistic factors. COMBO [32] employs a local search strategy while applying multi-starts to escape from local optima. CoCaBO [39] leverages a multi-armed bandit algorithm to optimize the categorical acquisition function. CASMOPOLITAN [46]

and Bounce [35] define local regions to restrict the search space of the acquisition function and then apply a local search in the categorical and mixed space. Bounce [35] further incorporates nested sub-space embedding to handle combinatorial and mixed spaces more effectively. Daulton et al. [5] propose a probabilistic reparameterization to relax the discrete acquisition function optimization into continuous domains, allowing the use of gradient-based optimizers.

The HESP Technique. The HESP technique [30] is developed from a well-known black-box optimization algorithm in Evolutionary Algorithms (EA), called Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [17], which aims to address high-dimensional black-box optimization problems. However, CMA-ES is inherently limited to continuous variables. Although several discrete variants have been proposed [15, 14], none of the CMA-ES based methods are capable of addressing optimization problems involving categorical variables.

Meta-algorithms for BO. In BO, several meta-algorithms have been developed to incorporate different BO optimizers, enhancing the the optimizers’ performance. LA-MCTS [47] learns regions with non-linear boundaries via K-mean classification. MCTS-VS [42] leverages the Monte Carlo Tree Search to perform dimensionality reduction and select a subset of important variables. The work in [30] defines systematic hyper-ellipsoid local regions using the CMA strategy. However, to the best of our knowledge, all these meta-algorithms only target continuous domains, hence cannot be directly applied to combinatorial and mixed problems.

3 Background

3.1 Combinatorial and Mixed Bayesian Optimization

Let us consider a minimization problem of an *expensive black-box* objective function $f : \mathcal{Z} \rightarrow \mathbb{R}$, where $\mathcal{Z} = \mathcal{X} \times \mathcal{H}$ is a d -dimensional mixed domain constructed from a d_x -dimensional continuous space $\mathcal{X} \subset \mathbb{R}^{d_x}$, a d_h -dimensional combinatorial space \mathcal{H} , $d = d_x + d_h$. By definition, $\mathcal{H} = \mathcal{W} \times \mathcal{Q}$ where \mathcal{W} is a d_w -dimensional ordinal space $\mathcal{W} \subset \mathbb{Z}^{d_w}$ and \mathcal{Q} is a d_q -dimensional categorical space \mathcal{Q} , $d_h = d_w + d_q$. Our goal is to find the global minimum $\mathbf{z}^* = [\mathbf{x}^*, \mathbf{h}^*]$ of the function f using *the least number of function evaluations*,

$$\mathbf{z}^* \in \arg \min_{\mathbf{z} \in \mathcal{Z}} f(\mathbf{z}), \quad (1)$$

where $\mathbf{z} = [\mathbf{x}, \mathbf{h}]$ is the input vector, $\mathbf{x} \in \mathcal{X}$ is the continuous input vector, $\mathbf{h} = [\mathbf{w}, \mathbf{q}] \in \mathcal{H}$ is the combinatorial input vector, $\mathbf{w} \in \mathcal{W}$ is the ordinal input vector, and $\mathbf{q} = [q_1, \dots, q_{d_q}] \in \mathcal{Q}$ is the categorical input vector, with q_i ($i \in [1, \dots, d_q]$) being a categorical variable having c_i unique categories. Note that in this work, we treat ordinal variables as categorical variables as in various works [39, 46, 35, 7].

In BO, the objective function is assumed to be *expensive*, i.e., time-consuming and/or costly to evaluate, and *black-box*, i.e., the analytical form and other information such as the gradient or Hessian are unavailable. Therefore, the primary goal of BO is to find the best solution with the least number of function evaluations. Details on BO including common surrogate models and acquisition functions for combinatorial and mixed BO are in the Appendix A.1 and A.2.

3.2 The Hyper-ellipsoid Space Partitioning Strategy

The HESP technique is a search space partitioning method designed to solve *high-dimensional* BO problems for continuous variables [30]. The key idea is to maintain a *multivariate normal search*

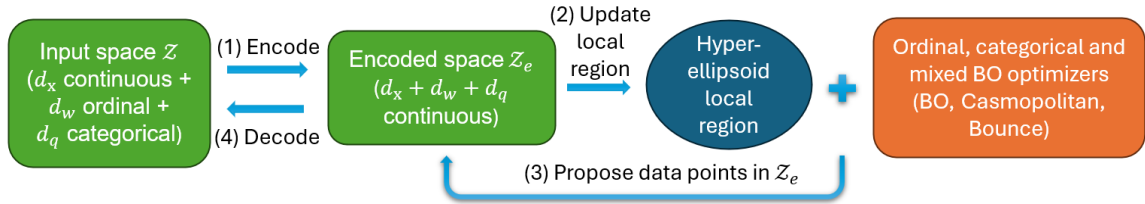


Figure 1. Illustration of MOCA-HESP meta-algorithm. (1) The mixed input data is encoded by an *adaptively chosen encoder*. (2) The hyper-ellipsoid local region, which is derived from the *base local region* (Eq. (3)) and the *distinct features* of the BO optimizers, is constructed/updated, (3) From the hyper-ellipsoid’s local region, the BO process (based on the chosen BO optimizes) proposes the next observations in the encoded spaces. (4) A decoder is used to transform the encoded data for function evaluations.

distribution $\mathcal{N}(\mathbf{m}, \Sigma)$ over the input search space, where \mathbf{m} is the mean vector and Σ is the covariance matrix of the search distribution. A local region is then defined as the α -confidence hyper-ellipsoid of $\mathcal{N}(\mathbf{m}, \Sigma)$. Within the local region, a BO process can be used to sequentially suggest $\lambda > 1$ data points for function evaluation. These λ observations are subsequently used to update the search distribution $\mathcal{N}(\mathbf{m}, \Sigma)$ and the local region in the next iteration via the mean update and covariance matrix adaptation formulas [17]. The local regions defined by HESP are shown to have a high probability of containing the global optimum of the objective function. The HESP strategy has been shown to not only perform well in finding global optima of high-dimensional optimization problems but also incurs minimal computation overhead [30]. However, despite its effectiveness, HESP can only work for continuous variables.

4 Proposed Methods

In this section, we first develop a HESP strategy tailored for combinatorial and mixed variables (Sec. 4.1). Then, we present our proposed meta-algorithm, MOCA-HESP, for BO in high-dimensional combinatorial and mixed spaces (Sec. 4.2). Finally, we develop three algorithms: MOCA-HESP-BO, MOCA-HESP-Casmo and MOCA-HESP-Bounce, obtained when incorporating MOCA-HESP with the state-of-the-art high-dimensional BO methods for these mixed spaces: standard BO, CASMOPOLITAN [46], and Bounce [35] (Sec. 4.3). Incorporating MOCA-HESP with CASMOPOLITAN and Bounce is especially challenging due to their distinct core features, such as different local region strategies or nested subspace embedding mechanisms, as well as their different requirements such as input data encoding. For an overview of these optimizers, see the Appendix A.4 and A.5.

4.1 The Proposed HESP Strategy for Combinatorial and Mixed Spaces

To develop HESP for combinatorial and mixed variables, we propose to use categorical encoders [36] to transform combinatorial (ordinal and categorical) variables into continuous values and apply HESP to these transformed variables. In addition, we also suggest a decoder to transform the encoded data back to the original domain for objective function evaluations. We further propose to adaptively select the optimal encoder for HESP using a multi-armed bandit (MAB) algorithm. Below, we describe these steps in detail.

Combinatorial and Mixed Variables Encoding and Decoding for HESP. From the mixed space $\mathcal{Z} = \mathcal{X} \times \mathcal{H}$, we convert combinatorial variables to continuous values by using a 1-to-1 mapping categorical encoders $g : \mathcal{H} \rightarrow \mathcal{H}_e$ where \mathcal{H} is a d_h -dimensional combinatorial space and $\mathcal{H}_e \subset \mathbb{R}^{d_h}$. In this work, we propose to use either

an *ordinal* or a *target* encoder [26] as these encoders are commonly used, effective, efficient in runtime, and do not require as much training data as a VAE [23]. Ordinal encoders map the inputs to numerical values based on existing or assumed relationships on their orders and rankings. Target encoders map the inputs to numerical values based on their target values, e.g., the average function values. More details on these two encoders are in the Appendix A.3. After training the encoder $g(\cdot)$, we apply the HESP technique to the encoded mixed spaces $\mathcal{Z}_e = \mathcal{X} \times \mathcal{H}_e$. Note that, in order to perform objective function evaluation on \mathcal{Z} , we need a decoder $\tilde{g} : \mathcal{H}_e \rightarrow \mathcal{H}$ to transform the encoded continuous values in \mathcal{H}_e back to the original combinatorial domain \mathcal{H} . In particular, to decode a numerical value $x \in \mathcal{H}_e$ to a c -choice combinatorial variable $h \in \{u_1, \dots, u_c\}$, we find a category u_i with the encoded value closest to the numerical value x , or formally $\tilde{g}(x) = u_i$ where $i = \arg \min_{i=1, \dots, c} |g(u_i) - x|$.

Adaptive Encoder Selection. In practice, as no encoder is guaranteed to be the best in all cases, to select the optimal encoders for HESP, we propose to use a multi-armed bandit (MAB) approach. We consider each encoder as an MAB action, and the goal is to select the best action (i.e., the best encoder), based on the optimization data. We propose to use EXP3 [2] as the MAB approach due to its general applicability, robustness to adversarial environments, and strong theoretical regret guarantees. Our proposed adaptive encoder selection method is outlined in the pseudo-code Alg. 1. Each encoder corresponds to an action, and is associated with a weight and a selection probability, which determines the likelihood of that action being chosen. Given K types of encoders and their associated weights $w_k(t)$ for $k = 1, 2, \dots, K$ at iteration t , the selection probability for each action k (line 3) is:

$$p_k(t) = (1 - \eta) \frac{w_k(t)}{\sum_{j=1}^K w_j(t)} + \frac{\eta}{K}, \quad (2)$$

where $\eta \in [0, 1]$ is a parameter that governs how much we should sample the actions uniformly at random (explore unknown actions) rather than focus on good performing actions (exploit potential actions) [2]. In each iteration, to update the weights $w_k(t)$, we compute the estimated reward $\hat{r}(t)$ from the λ observations in an iteration of MOCA-HESP. We first normalize the observed function values $y_i|_{i=1}^t$ to $\hat{y}_i|_{i=1}^t$ such that $\hat{y}_i = (y_i - \max \mathbf{y}) / (\min \mathbf{y} - \max \mathbf{y})$ (line 4). This normalization ensures that a high reward $\hat{r}(t)$ is given only when the reward \hat{y}_i is similarly good or better, increasing the action’s weight and selection probability [1]. Otherwise, the probability remains unchanged, allowing EXP3 to explore other actions. The estimated reward is then calculated as $\hat{r}(t) = r(t) / p_{k_t}(t)$, where $r(t) = \min \hat{y}_i|_{i=k_t-\lambda+1}^t$ is the minimum normalized function value (lines 5-6). Subsequently, we update the current selected action’s weight $w_{k_t}(t+1) = w_{k_t}(t) \exp(\eta \hat{r}(t) / K)$, and preserve other ac-

tions' weights, $w_j(t+1) = w_j(t)$, $j \neq k_t$ (line 7). Finally, a new selection probability is computed (line 8) to sample a new encoder type k_{t+1} for the next iteration (line 9).

Algorithm 1 The Adaptive Encoder Selection.

- 1: **Input:** The observed function values $y_j|_{j=1}^t$ up to iteration t , the number of categorical encoder types K , the weight vector $w_k(t)|_{k=1}^K$, the current encoder type k_t , the parameters η and λ
 - 2: **Output:** The next encoder type k_{t+1} .
 - 3: Compute the probability $p_k(t)$, $k = 1, \dots, K$ ▷ Eq. (2)
 - 4: Normalize $y_i|_{i=1}^n$ to $\hat{y}_i|_{i=1}^t$, where $\hat{y}_i \in [0, 1]$
 - 5: Compute the reward $r(t) = \min \hat{y}_i|_{k=t-\lambda+1}^t$.
 - 6: Compute the estimated reward $\hat{r}(t) = r(t)/p_{k_t}(t)$
 - 7: Update the weights $w_k(t+1)$, $k = 1, \dots, K$
 - 8: Update the selection probability $p_k(t+1)$, $k = 1, \dots, K$
 - 9: Sample the next encoder type k_{t+1} following $p_k(t+1)$
 - 10: Return the encoder type k_{t+1}
-

4.2 The Proposed MOCA-HESP Meta-Algorithm

In this section, we present our proposed meta-algorithm MOCA-HESP for BO in high-dimensional combinatorial and mixed spaces.

Overall Process. The overall process of MOCA-HESP is shown in Alg. 2 and illustrated in Fig. 1. Initially, all K encoder weights are set as 1 (line 6). Based on the initial observed dataset $\mathcal{D}_0 \in \mathcal{Z}$ (line 7), we construct an initial hyper-ellipsoid local region $\mathcal{E}_{\text{bo_opt}}^{(0)}$ in the encoded space $\mathcal{Z}_e \subseteq \mathbb{R}^d$ (line 8). Subsequently, within the local region $\mathcal{E}_{\text{bo_opt}}^{(0)}$, the BO optimizer `bo_opt` suggests λ data points $\mathcal{D}_\lambda \in \mathcal{Z}$ (line 11). Then, the encoded λ data points $g_k^{(0)}(\mathcal{D}_\lambda)$ are used to update the local region $\mathcal{E}_{\text{bo_opt}}^{(1)}$ (line 12). Furthermore, a new categorical encoder $g_k^{(1)}(\cdot)$ is also suggested for the next iteration based on Alg. 1 (line 14). Restart conditions [16, 30] are verified every iteration to prevent the algorithm from being trapped at a local optimum (line 15). This process repeats until the evaluation budget

Algorithm 2 The MOCA-HESP Meta-algorithm.

- 1: **Input:** The objective function $f(\cdot)$, the mixed search domain \mathcal{Z} , the evaluation budget N , the number of initial points n_0 , the BO optimizer `bo_opt`, the number of categorical encoder types K
 - 2: **Output:** The optimum \mathbf{z}^*
 - 3: Set $\lambda, t \leftarrow 0, T \leftarrow \lfloor (N - n_0)/\lambda \rfloor$, the dataset $\mathcal{D} \leftarrow \emptyset$
 - 4: **while** $t \leq T$ **do**
 - 5: Set the encoder weights $w_k = 1$, for $i = 1 \dots K$
 - 6: Choose and train the categorical encoder $g_k^{(t)}(\cdot)$ ▷ Eq. (2)
 - 7: Sample a set of n_0 initial data points \mathcal{D}_0
 - 8: Set the initial HE local region $\mathcal{E}_{\text{bo_opt}}^{(0)}$ based on $g_k^{(0)}(\mathcal{D}_0)$
 - 9: Set $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_0$
 - 10: **while** $t \leq T$ **do**
 - 11: Use `bo_opt` to propose a set of λ data points $\mathcal{D}_\lambda \in \mathcal{Z}$.
 - 12: Update $\mathcal{E}_{\text{bo_opt}}^{(t+1)} \leftarrow \text{HESP}(g_k^{(t)}(\mathcal{D}_\lambda))$
 - 13: Update $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\lambda, t \leftarrow t + 1$
 - 14: Select and train a new encoder $g_k^{(t+1)}(\cdot)$ ▷ Alg. 1
 - 15: Break loop **if** the stopping criteria are satisfied
 - 16: **end while**
 - 17: **end while**
 - 18: Return $\mathbf{z}^* = \arg \min_{\mathbf{z}_i \in \mathcal{D}} \{y_i\}_{i=1}^N$ from the dataset \mathcal{D}
-

is depleted, returning the best data point in dataset \mathcal{D} as the optimum \mathbf{z}^* . In the below we describe in detail each component of our proposed MOCA-HESP meta-algorithm.

Local Region Formulation. We provide a detailed formulation of the local regions for HESP in combinatorial and mixed domains given the aforementioned categorical encoders. We first define a *base local region* \mathcal{E}_b based on a *multivariate normal search distribution* $\mathcal{N}_{\mathcal{Z}_e} = \mathcal{N}_{\mathcal{Z}_e}(\mathbf{m}_{\mathcal{Z}_e}, \Sigma_{\mathcal{Z}_e})$ in the encoded space \mathcal{Z}_e , where $\mathbf{m}_{\mathcal{Z}_e}$ and $\Sigma_{\mathcal{Z}_e}$ are the encoded mean vector and covariance matrix of the search distribution. Specifically, \mathcal{E}_b is defined as the α -confidence hyper-ellipsoid of the search distribution, i.e.,

$$\mathcal{E}_b = \{\mathbf{z}_e \mid \mathbb{M}(\mathbf{z}_e, \mathcal{N}_{\mathcal{Z}_e}) \leq \chi_{1-\alpha}^2(d_{\mathcal{Z}_e})\}, \quad (3)$$

where $\mathbb{M}(\mathbf{x}, \mathcal{N}) = \sqrt{(\mathbf{x} - \mathbf{m})^\top \Sigma (\mathbf{x} - \mathbf{m})}$ is the Mahalanobis distance from a continuous vector \mathbf{x} to a multivariate normal distribution $\mathcal{N} = \mathcal{N}(\mathbf{m}, \Sigma)$ and $\chi_{1-\beta}^2(\gamma)$ denotes a Chi-squared β critical value with γ degree of freedom. From the base local region \mathcal{E}_b , additional requirements from the respective BO optimizers will be incorporated to create the local region $\mathcal{E}_{\text{bo_opt}}$ in later sections.

Local Optimization. In each iteration, we apply the BO optimizer `bo_opt` within the local region $\mathcal{E}_{\text{bo_opt}}$ to propose λ data points in $\mathcal{z} \in \mathcal{Z}$. When fitting the GP, we preserve all settings from the BO optimizer, e.g., the kernel and input encoding type. To propose the next observed data, we optimize the same acquisition function employed by the BO optimizer. Specifically, we first sample a pool of candidate data points $\mathbf{z}_{e\alpha} \in \mathcal{Z}_e$ following $\mathcal{N}_{\mathcal{Z}_e}(\mathbf{m}_{\mathcal{Z}_e}, \Sigma_{\mathcal{Z}_e})$ and confined within $\mathcal{E}_{\text{bo_opt}}$. These candidate points assist the acquisition function optimization depending on the type of acquisition function employed by the BO optimizer. Furthermore, the next observation data points must satisfy the local region $\mathcal{E}_{\text{bo_opt}}$. Details for each optimizer will be explained in the next sections. Note that if the acquisition functions are defined in the original mixed space \mathcal{Z} , we use a decoder to transform $\mathbf{z}_{e\alpha}$ to $\mathbf{z}_\alpha \in \mathcal{Z}$ to compute the acquisition function values.

4.3 The Proposed MOCA-HESP-BO, MOCA-HESP-Casmo, MOCA-HESP-Bounce

In this section, we develop three algorithms by incorporating our meta-algorithm MOCA-HESP with the state-of-the-art combinatorial and mixed BO optimizers: standard BO, CASMOPOLITAN [46], and Bounce [35]. Incorporating MOCA-HESP with CASMOPOLITAN and Bounce is especially challenging, as these optimizers have *different requirements* for input data encoding, kernel type, and acquisition function, as well as *distinct components* such as local region formulations and subspace embedding techniques. The key ideas of the three proposed methods are summarized in Table 1.

4.3.1 The MOCA-HESP-BO Algorithm

This algorithm is developed by incorporating MOCA-HESP with the standard BO optimizer. MOCA-HESP-BO shares similar components with the meta-algorithm MOCA-HESP, including the formulation of the local region \mathcal{E}_{BO} and the local optimization step.

4.3.2 The MOCA-HESP-Casmo Algorithm

This algorithm is developed by incorporating our meta-algorithm MOCA-HESP with CASMOPOLITAN [46]. The challenge is that we need to incorporate CASMOPOLITAN's local region adaptation

Table 1. Key ideas in the three MOCA-HESP algorithms. MOCA-HESP can robustly incorporate different BO optimizers that (1) require different settings, e.g., data encoding, kernel, acquisition function, and (2) have distinct core features, e.g., local region adaptation mechanisms, subspace embedding techniques.

MOCA-HESP Method	Local Region	Local Optimization
MOCA-HESP-BO	<ul style="list-style-type: none"> • In the encoded space \mathcal{Z}_e. • Local regions \mathcal{E}_{BO} satisfy the Mahalanobis distance criterion. 	<ul style="list-style-type: none"> • Train a GP with ordinal encoded data (Matern 5/2 kernel). • Sample candidates in \mathcal{Z}_e via a scaled search distribution (for Mahalanobis criterion). • Decode candidates back to \mathcal{Z}. • Maximize the acquisition function over candidates.
MOCA-HESP-Casmo	<ul style="list-style-type: none"> • In the encoded space \mathcal{Z}_e. • Local regions $\mathcal{E}_{\text{Casmo}}$ satisfy the Mahalanobis distance criterion <i>and</i> the Hamming distance criterion. 	<ul style="list-style-type: none"> • Train a GP with ordinal encoded data (Transform Overlapped kernel). • Sample candidates in \mathcal{Z}_e via a scaled search distribution (for Mahalanobis criterion). • Decode candidates back to \mathcal{Z}. • Randomly reset several categorical variables to satisfy the Hamming criterion. • Maximize the acquisition function over candidates.
MOCA-HESP-Bounce	<ul style="list-style-type: none"> • In the encoded subspace \mathcal{V}. • Local regions $\mathcal{E}_{\text{Bounce}}$ satisfy the Mahalanobis distance criterion <i>and</i> the Hamming distance criterion. 	<ul style="list-style-type: none"> • Train a GP with one-hot encoded data (Matern 5/2 kernel). • Sample candidates in \mathcal{V} via a scaled search distribution (for Mahalanobis criterion). • Decode candidates back to the subspace \mathcal{A}. • Randomly reset several categorical variables to satisfy the Hamming criterion. • Maximize the acquisition function in \mathcal{A} and project the best candidate to \mathcal{Z}: <ul style="list-style-type: none"> – Categorical problems: Local search over candidates. – Mixed problems: Interleave search the categorical and continuous variables separately. While optimizing categorical variables, fix the candidates’ continuous variables to the current best solution, and vice versa.

mechanism, which maintains two separate local regions for continuous and categorical variables. To overcome this challenge, first, we incorporate the continuous local region (governed by a scaling factor L_x) by scaling the base local region in Eq. (3) by L_x . In particular, we scale the covariance matrix $\Sigma_{\mathcal{Z}_e}$ by a $d_{\mathcal{Z}_e}$ -dimensional scaling vector \mathbf{L} , where $L_i = L_x^2$ if the i -th dimension is associated with a continuous variable, and $L_i = 1$ otherwise. This results in a scaled distribution $\mathcal{N}_{\mathcal{Z}_e, \mathbf{L}} = \mathcal{N}_{\mathcal{Z}_e}(\mathbf{m}_{\mathcal{Z}_e}, \psi(\Sigma_{\mathcal{Z}_e}, \mathbf{L}))$, where $\psi(\Sigma, \mathbf{S})$ scales the radii of the α -level confidence hyper-ellipsoid of Σ by the vector \mathbf{S} . Second, to incorporate the categorical local region (governed by a Hamming distance threshold L_h), we impose an additional threshold to the base local region. That is, we employ additional constraint $\mathbb{H}(\tilde{g}_k(\mathbf{z}_e), \tilde{g}_k(\mathbf{m}_{\mathcal{Z}_e})) \leq L_h$, where $\mathbb{H}(\cdot, \cdot)$ is the Hamming distance between categorical data points, and $\tilde{g}_k(\cdot)$ is the categorical decoder from \mathcal{Z}_e to \mathcal{Z} . Overall, the local region $\mathcal{E}_{\text{Casmo}}$ is the combination of two requirements, such that,

$$\mathcal{E}_{\text{Casmo}} = \{\mathbf{z}_e \mid \mathbb{M}(\mathbf{z}_e, \mathcal{N}_{\mathcal{Z}_e, \mathbf{L}}) \leq \chi_{1-\alpha}^2(d_{\mathcal{Z}_e}); \mathbb{H}(\tilde{g}_k(\mathbf{z}_e), \tilde{g}_k(\mathbf{m}_{\mathcal{Z}_e})) \leq L_h\}. \quad (4)$$

Having defined the local region $\mathcal{E}_{\text{Casmo}}$, we carry out the local optimization to select the next observation data point. The choice of the surrogate model and the acquisition function are similar to CASMOPOLITAN, however, acquisition optimization needs to satisfy the local region $\mathcal{E}_{\text{Casmo}}$. Specifically, we sample a pool of candidates \mathbf{z}_e from the scaled search distribution $\mathcal{N}_{\mathcal{Z}_e, \mathbf{L}}$ that satisfy the Mahalanobis distance criterion. Then we decode these candidates back to the mixed spaces \mathcal{Z} via the decoder $\tilde{g}_k(\cdot)$. Then, random selection is carried out to alter several dimensions of $\tilde{g}_k(\mathbf{z}_e)$ until they all satisfy the Hamming distance criterion. New observations are chosen from these candidates by maximizing the acquisition function values.

4.3.3 The MOCA-HESP-Bounce Algorithm

This algorithm is developed by incorporating MOCA-HESP with Bounce [35], a recent state-of-the-art BO method for combinatorial and mixed variables. The challenge is how to incorporate Bounce’s expanding subspace embedding technique, which optimizes a series of mixed low-dimensional subspaces \mathcal{A} (*mixed target subspace*) of increasing dimensions $d_{\mathcal{A}}$. For an overview of subspace embedding in Bounce, refer to the Appendix A.5. We denote the *encoded target subspace* $\mathcal{V} \subset \mathbb{R}^{d_{\mathcal{A}}}$ as a $d_{\mathcal{V}}$ -dimensional encoded space of \mathcal{A} . The relationship between these spaces is given by $d_{\mathcal{A}} = d_{\mathcal{V}} < d_{\mathcal{Z}_e} = d_{\mathcal{Z}}$.

In each iteration of MOCA-HESP-Bounce, we first derive the base local region (Eq. (3)) in the encoded space \mathcal{V} via a projected search distribution $\mathcal{N}_{\mathcal{V}}(\mathbf{m}_{\mathcal{V}}, \Sigma_{\mathcal{V}})$, where $\mathbf{m}_{\mathcal{V}}, \Sigma_{\mathcal{V}}$ are the mean vector and the covariance matrix, respectively. To compute $\mathcal{N}_{\mathcal{V}}(\mathbf{m}_{\mathcal{V}}, \Sigma_{\mathcal{V}})$, we project the search distribution $\mathcal{N}_{\mathcal{Z}_e}(\mathbf{m}_{\mathcal{Z}_e}, \Sigma_{\mathcal{Z}_e})$ onto \mathcal{V} using the embedding matrix $\mathbf{P} : \mathcal{Z}_e \rightarrow \mathcal{V}$, which is the Moore–Penrose inverse of Bounce’s projection matrix $\mathbf{Q} : \mathcal{V} \rightarrow \mathcal{Z}_e$ [12, 30]. Subsequently, as Bounce employs CASMOPOLITAN’s local region adaptation mechanism, we apply the similar technique as in MOCA-HESP-Casmo to define the respective local region in \mathcal{V} - via scaling the base local region by a continuous factor L_x and imposing a Hamming constraint by a categorical factor L_h . Hence, the local region $\mathcal{E}_{\text{Bounce}}$ is,

$$\mathcal{E}_{\text{Bounce}} = \{\mathbf{v} \mid \mathbb{M}(\mathbf{v}, \mathcal{N}_{\mathcal{V}, \mathbf{L}}) \leq \chi_{1-\alpha}^2(d_{\mathcal{V}}); \mathbb{H}(\tilde{h}_k(\mathbf{v}), \tilde{h}_k(\mathbf{m}_{\mathcal{V}})) \leq L_h\}, \quad (5)$$

where $\mathcal{N}_{\mathcal{V}, \mathbf{L}} = \mathcal{N}_{\mathcal{V}, \mathbf{L}}(\mathbf{m}_{\mathcal{V}}, \psi(\Sigma_{\mathcal{V}}, \mathbf{L}))$ and $\tilde{h}_k(\cdot)$ is the categorical decoder. Having defined the local region $\mathcal{E}_{\text{Bounce}}$, we carry out the local optimization to select the next observation data point. The choice of the surrogate model and the acquisition function are similar to Bounce, yet the acquisition optimization needs to satisfy $\mathcal{E}_{\text{Bounce}}$. Specifically, the candidate sampling steps need to satisfy both the Mahalanobis distance and the Hamming distance criterion in the encoded target subspace \mathcal{V} . For categorical problems, we use local search over the generated candidates. For mixed problems, we leverage the interleave search procedure and optimize combinatorial and continuous parts separately. While optimizing combinatorial variables, we fix the continuous values of the candidates, and vice versa. Note that as MOCA-HESP-Bounce suggests data points in \mathcal{V} , we decode the solution to \mathcal{A} , then project it to \mathcal{Z} for function evaluations.

5 Experiments

5.1 Experiment Setup and Baselines

We extensively evaluate our proposed MOCA-HESP meta-algorithm by comparing the three developed algorithms MOCA-HESP-BO, MOCA-HESP-Casmo and MOCA-HESP-Bounce against: (1) their respective BO optimizers: **Standard BO**, **CASMOPOLITAN** [46], **Bounce** [35], (2) other state-of-the-art BO optimizers for high-dimensional combinatorial and mixed variables including **TPE** [4],

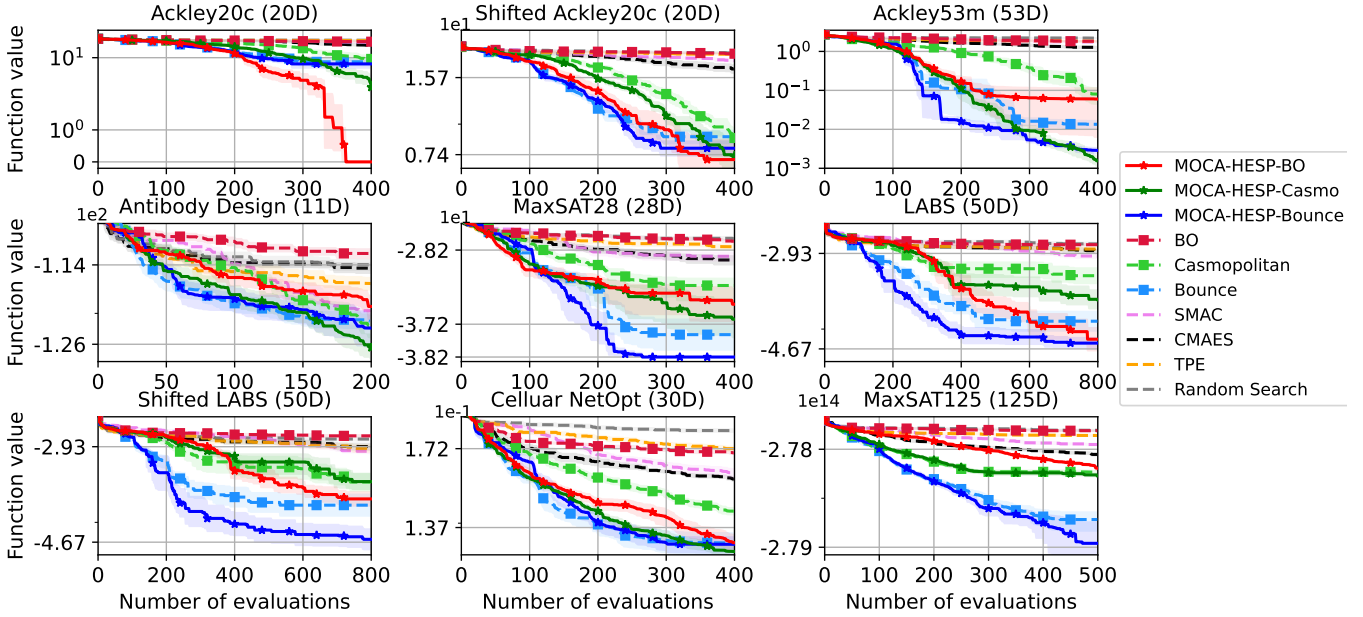


Figure 2. Comparison between the MOCA-HESP methods (MOCA-HESP-BO, MOCA-HESP-Casmo, MOCA-HESP-Bounce) against (1) the original BO optimizers (BO, CASMOPOLITAN, Bounce) respectively and (2) the other baselines (TPE, SMAC, CMA-ES, Random Search). The results show that all the MOCA-HESP methods outperform or are on par with their respective BO optimizers (e.g. MOCA-HESP-BO outperforms BO, MOCA-HESP-Casmo outperforms CASMOPOLITAN, and MOCA-HESP-Bounce outperforms Bounce), and also outperform the other baselines.

SMAC [20], (3) CMA-ES [17], an evolutionary optimization algorithm, in which we used the ordinal encoder to transform combinatorial inputs to numeric values, and (4) Random Search. Note that there are no meta-algorithm baselines as to the best of our knowledge, we are the first to propose a meta-algorithm for high-dimensional BO in combinatorial and mixed spaces. Details of the settings for MOCA-HESP and the baselines are in the Appendix A.6 and Appendix A.7.

5.2 Benchmark Problems

We conduct experiments on 9 benchmark problems: 3 synthetic and 6 real-world problems. The synthetic problems include 2 combinatorial problems: *Ackley20c - 20D* and *Shifted Ackley20c - 20D*, and 1 mixed problem: *Ackley53m - 53D* [39, 46, 35, 7]. The real-world problems include 5 combinatorial ones: *Antibody Design - 11D* [22], *LABS - 50D* [33], *MaxSAT28 - 28D*, *MaxSAT125-125D* [18], *Shifted LABS - 50D*, and 1 mixed problem: *Cellular Network - 30D* [9]. The dimensionalities of these problems range from **11 to 125**, and the cardinalities of the combinatorial variables range from **2 to 20**. These problems have been widely used in other works to evaluate high-dimensional BO methods for combinatorial and mixed spaces [32, 39, 46, 5, 7]. Note that we also evaluate our methods using shifted benchmark functions, which are designed to eliminate the special structure, e.g., zero-sequence patterns, of the global optimum in the original test functions [35, 30]. Details of these benchmark problems are in Table 2 and the Appendix A.8.

5.3 Performance Comparison with Baselines

Fig. 2 compares our proposed methods against all baselines. Firstly, we show that *MOCA-HESP-BO*, *MOCA-HESP-Casmo* and *MOCA-HESP-Bounce* noticeably outperform their respective BO optimizers, demonstrating the effectiveness of our proposed MOCA-HESP meta-algorithm. Specifically, *MOCA-HESP-BO* consistently outperforms

Table 2. Details of benchmark problems.

Benchmark problems	Inputs (x is continuous, h is combinatorial)	# evals
Ackley20c (20D)	$h \in [-32.768, \dots, 32.768]^{20}$, each has 11 evenly spaced values	400
Antibody Design (11D)	$h \in AA^{11}$, where AA is the set of 20 amino acids	200
LABS (50D)	$h \in \{-1, +1\}^{50}$	800
MaxSAT28 (28D)	$h \in \{0, 1\}^{28}$	400
Ackley53m (53D)	$x \in [-1, 1]^3$ $h \in \{0, 1\}^{50}$	400
Cellular Network (30D)	$x \in [30, 50]^{15}$ $h \in \{0, 5\}^{15}$	400
MaxSAT125 (125D)	$h \in \{0, 1\}^{25}$	500

standard BO in all 9 benchmark problems by large margins. Notably, on *Ackley20c*, *MOCA-HESP-BO* surpasses all baselines and effectively finds the global optimum within the budget, highlighting the effect of MOCA-HESP in identifying the global optimum. Compared to *CASMOPOLITAN*, *MOCA-HESP-Casmo* shows superior performance on 7 out of the 9 benchmark problems. In these 7 problems, *MOCA-HESP-Casmo* consistently outperforms *CASMOPOLITAN* across all iterations. *MOCA-HESP-Bounce* outperforms *Bounce* on 6 benchmark problems, including *Shifted Ackley20c*, *Ackley53*, *LABS*, *Shifted LABS*, *MaxSAT28* and *MaxSAT125*. On *MaxSAT28*, *LABS* and *Shifted LABS*, *MOCA-HESP-Bounce* maintains more optimal than *Bounce* across all iterations. On *MaxSAT125D*, *MOCA-HESP-Bounce* initially performs similarly to *Bounce* but eventually finds more optimal solutions as the iterations progress.

Finally, all three *MOCA-HESP* methods outperform other baselines, including *TPE*, *SMAC*, *CMA-ES* and *Random Search* on all 9 benchmark problems. These results showcase the effectiveness of

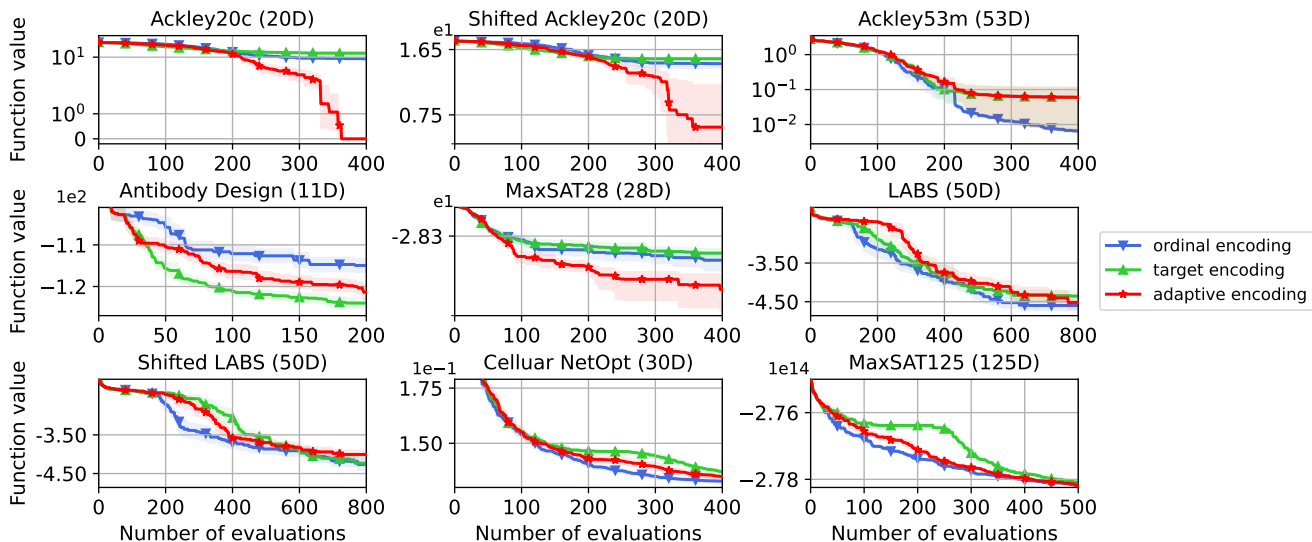


Figure 3. The effectiveness of using the proposed *adaptive* encoder selection mechanism against using only a single type of encoder. The performance of the adaptive encoder version is better or at least on par with the other versions.

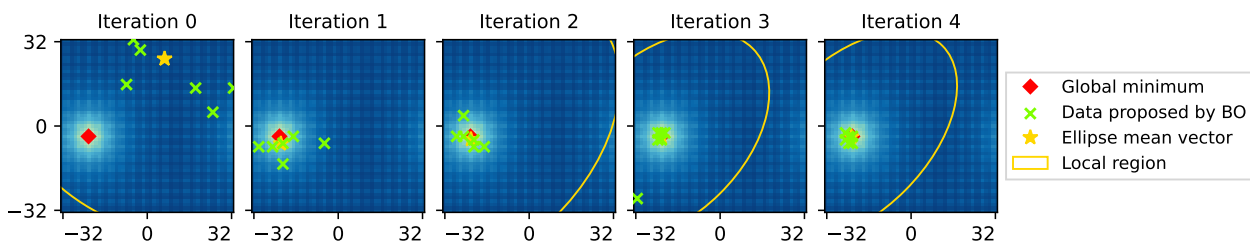


Figure 4. The trajectories of the local regions from MOCA-HESP-BO on the 2D Shifted-Ackley categorical function. The hyper-ellipsoid local region starts at a random location, then evolves to better locations and finally directs the mean vector to the global optimum. The radii are prevented from shrinking too small to maintain a sufficient number of candidate points inside the local regions.

the MOCA-HESP meta-algorithm in enhancing the performance of state-of-the-art BO optimizers.

5.4 Ablation Study and Analysis

The Effectiveness of Adaptive Encoder Selection. We evaluate the effectiveness of our proposed categorical encoder selection in choosing the optimal encoder for the HESP strategy. We compare three variants of MOCA-HESP-BO, each using a different categorical encoder strategy: (1) the ordinal encoder, (2) the target encoder, and (3) our proposed adaptive optimal encoder selection. We perform the analysis on all 9 benchmark problems. In Fig. 3, we can see that the adaptive version has the best performance. The ordinal encoder shows good results on MaxSAT125 and LABS, yet falls behind substantially on Antibody Design. The target encoder is the most optimal on the Antibody Design problem, but degrades substantially on Ackley20c and MaxSAT125. The analysis indicates the robustness of our proposed adaptive encoder selection strategy for MOCA-HESP.

The Trajectories of Local Regions under the MOCA-HESP Meta-algorithm. We visualize the trajectories of the local regions generated by our MOCA-HESP meta-algorithm on two 2D test functions: Ackley and Shifted Ackley. The search domain $[-32.768, \dots, 32.768]$ is discretized into 51 evenly spaced points. Fig. 4 shows the trajectories for the Shifted Ackley function; additional plots for the Ackley function are provided in the Appendix A.9. Note that, we can only plot for MOCA-HESP-BO be-

cause the Hamming distance criteria in the local region of MOCA-HESP-Casmo and MOCA-HESP-Bounce do not work in 2D problems ($L_{h_{\min}} = 1$ while $L_{h_{\text{init}}} = 2$). In Fig. 4, we can see that at the beginning (Iteration 0), the proposed data points are scattered randomly to explore the region, as the prior information of BO is insufficient. However, when moving on to the next iterations, the local region evolves closer to the global optimum, helping BO to propose better solutions. Note that in combinatorial spaces, the search distribution is lower-bounded by a minimum standard deviation threshold in order to maintain a sufficiently large area for sampling [25]. Therefore, the radii of the local region are also retained large, while the mean vector gradually moves closer to the global minimum.

6 Conclusion

In this paper, we propose MOCA-HESP, the *first meta-algorithm* for high-dimensional BO in the combinatorial and mixed domains. MOCA-HESP incorporates categorical encoders, which are optimally selected via MAB, to partition the mixed search space into hyper-ellipsoid local regions for local modeling and optimization. We further derived three practical algorithms, namely MOCA-HESP-BO, MOCA-HESP-Casmo and MOCA-HESP-Bounce, by incorporating MOCA-HESP with the state-of-the-art BO optimizers for combinatorial and mixed variables: BO, CASMOPOLITAN, Bounce, respectively. Our extensive empirical results demonstrate the effectiveness of our proposed MOCA-HESP meta-algorithm.

Acknowledgements

This work is partially supported by Australian Research Council (ARC) Discovery Project DP220103044. The first author (L.N.) would like to thank the School of Computing Technologies, RMIT University, Australia and the Google Cloud Research Credits Program for providing computing resources for this project. Additionally, this project was undertaken with the assistance of computing resources from RMIT AWS Cloud Supercomputing Hub (RACE).

References

- [1] A. Ahmadianchalchi, S. Belakaria, and J. R. Doppa. Pareto front-diverse batch multi-objective bayesian optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.
- [2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.
- [3] R. Baptista and M. Poloczek. Bayesian optimization of combinatorial structures. In *ICML*, pages 462–471. PMLR, 2018.
- [4] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyperparameter optimization. *NeurIPS*, 24, 2011.
- [5] S. Daulton, X. Wan, D. Eriksson, M. Balandat, M. A. Osborne, and E. Bakshy. Bayesian optimization over discrete and mixed spaces via probabilistic reparameterization. *NeurIPS*, 35, 2022.
- [6] A. Deshwal, S. Belakaria, and J. R. Doppa. Bayesian optimization over hybrid spaces. In *ICML*, pages 2632–2643. PMLR, 2021.
- [7] A. Deshwal, S. Ament, M. Balandat, E. Bakshy, J. R. Doppa, and D. Eriksson. Bayesian optimization over high-dimensional combinatorial spaces via dictionary-based embeddings. In *AISTATS*, pages 7021–7039. PMLR, 2023.
- [8] K. Dreczkowski, A. Grosnit, and H. Bou Ammar. Framework and benchmarks for combinatorial and mixed-variable bayesian optimization. *NeurIPS*, 36, 2024.
- [9] R. M. Dreifuerst, S. Daulton, Y. Qian, P. Varkey, M. Balandat, S. Kasturia, A. Tomar, A. Yazdan, V. Ponnampalam, and R. W. Heath. Optimizing coverage and capacity in cellular networks using machine learning. In *ICASSP*, 2021.
- [10] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable Global Optimization via Local Bayesian Optimization. In *NeurIPS*, 2019.
- [11] E. C. Garrido-Merchán and D. Hernández-Lobato. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing*, 380:20–35, 2020.
- [12] G. H. Golub and C. F. Van Loan. *Matrix computations*. JHU press, 2013.
- [13] S. Guan and N. Fu. Class imbalance learning with bayesian optimization applied in drug discovery. *Scientific reports*, 12(1):2069, 2022.
- [14] R. Hamano, S. Saito, M. Nomura, and S. Shirakawa. Cma-es with margin: Lower-bounding marginal probability for mixed-integer black-box optimization. In *Proceedings of GECCO*, 2022.
- [15] N. Hansen. *A CMA-ES for mixed-integer nonlinear optimization*. PhD thesis, INRIA, 2011.
- [16] N. Hansen. The CMA Evolution Strategy: A Tutorial, 2016.
- [17] N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2): 159–195, 2001.
- [18] P. Hansen and B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, 1990.
- [19] Y. Hu, J. Hu, Y. Xu, F. Wang, and R. Z. Cao. Contamination control in food supply chain. In *Proceedings of the 2010 Winter Simulation Conference*. IEEE, 2010.
- [20] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5*, 2011.
- [21] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, and E. P. Xing. Neural Architecture Search with Bayesian Optimisation and Optimal Transport. In *NeurIPS*, pages 2020–2029, 2018.
- [22] A. Khan, A. I. Cowen-Rivers, A. Grosnit, D.-G.-X. Deik, P. A. Robert, V. Greiff, E. Smorodina, P. Rawat, K. Dreczkowski, R. Akbar, et al. Towards real-world automated antibody design with combinatorial bayesian optimisation. *arXiv*, 2022.
- [23] D. P. Kingma, M. Welling, et al. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4), 2019.
- [24] B. Letham, R. Calandra, A. Rai, and E. Bakshy. Re-examining linear embeddings for high-dimensional bayesian optimization. In *NeurIPS*, volume 33, 2020.
- [25] T. Marty, Y. Semet, A. Auger, S. Héron, and N. Hansen. Benchmarking cma-es with basic integer handling on a mixed-integer test problem suite. In *GECCO*, 2023.
- [26] D. Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD*, 2001.
- [27] J. Mockus, V. Tiesis, and A. Zilinskas. The Application of Bayesian Methods for Seeking the Extremum. *Towards Global Optimization*, 2(117-129):2, 1978.
- [28] H. Moss, D. Leslie, D. Beck, J. Gonzalez, and P. Rayson. Boss: Bayesian optimization over string spaces. *NeurIPS*, 33, 2020.
- [29] A. Nayebi, A. Munteanu, and M. Poloczek. A Framework for Bayesian Optimization in Embedded Subspaces. In *ICML*, volume 97, 2019.
- [30] L. Ngo, H. Ha, J. Chan, V. Nguyen, and H. Zhang. High-dimensional bayesian optimization via covariance matrix adaptation strategy. *Transactions on Machine Learning Research (TMLR)*, 2024.
- [31] L. Ngo, H. Ha, J. Chan, and H. Zhang. Boids: High-dimensional bayesian optimization via incumbent-guided direction lines and subspace embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 19659–19667, 2025.
- [32] C. Oh, J. M. Tomczak, E. Gavves, and M. Welling. Combinatorial bayesian optimization using the graph cartesian product. In *NeurIPS*, pages 2910–2920, 2019.
- [33] T. Packebusch and S. Mertens. Low autocorrelation binary sequences. *Journal of Physics A: Mathematical and Theoretical*, 49(16), 2016.
- [34] L. Papenmeier, L. Nardi, and M. Poloczek. Increasing the scope as you learn: Adaptive bayesian optimization in nested subspaces. *NeurIPS*, 2022.
- [35] L. Papenmeier, L. Nardi, and M. Poloczek. Bounce: Reliable high-dimensional bayesian optimization for combinatorial and mixed spaces. *NeurIPS*, 2023.
- [36] K. Potdar, T. S. Pardawala, and C. D. Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International journal of computer applications*, 175(4):7–9, 2017.
- [37] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006. ISBN 026218253X.
- [38] H. E. Robbins. An empirical bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*. Springer, 1992.
- [39] B. X. Ru, A. S. Alvi, V. Nguyen, M. A. Osborne, and S. J. Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *ICML*, volume 119, 2020.
- [40] Y. Shen, Y. Li, J. Zheng, W. Zhang, P. Yao, J. Li, S. Yang, J. Liu, and B. Cui. Proxybo: Accelerating neural architecture search via bayesian optimization with zero-cost proxies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9792–9801, 2023.
- [41] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *NeurIPS*, volume 25. Curran Associates, Inc., 2012.
- [42] L. Song, K. Xue, X. Huang, and C. Qian. Monte carlo tree search based variable selection for high dimensional bayesian optimization. In *NeurIPS*, 2022.
- [43] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th ICML*, pages 1015–1022, 2010.
- [44] A. Thebelt, C. Tsay, R. Lee, N. Sudermann-Merx, D. Walz, B. Shafei, and R. Misener. Tree ensemble kernels for bayesian optimization with known constraints over mixed-feature spaces. *NeurIPS*, 35, 2022.
- [45] W. R. Thompson. On the likelihood that an unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933. ISSN 0006-3444.
- [46] X. Wan, V. Nguyen, H. Ha, B. Ru, C. Lu, and M. A. Osborne. Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. *ICML* 38, 2021.
- [47] L. Wang, R. Fonseca, and Y. Tian. Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search. In *NeurIPS*, volume 33, 2020.
- [48] Z. Wang, G. E. Dahl, K. Swersky, C. Lee, Z. Nado, J. Gilmer, J. Snoek, and Z. Ghahramani. Pre-trained gaussian processes for bayesian optimization. *Journal of Machine Learning Research*, 25(212):1–83, 2024.
- [49] C. Zhang and X. Zhou. Forecasting credit risk of smes in supply chain finance using bayesian optimization and xgboost. *Mathematical Problems in Engineering*, 2023(1):5609996, 2023.
- [50] J. K. Ziomek and H. B. Ammar. Are random decompositions all we need in high dimensional bayesian optimisation? In *ICML*. PMLR, 2023.

A Appendix

A.1 General Process in Bayesian Optimization

BO addresses the optimization problem in an iterative fashion. At iteration t , BO approximates the objective function f using a surrogate model based on the observed data collected so far $\mathcal{D}_t = \{\mathbf{z}_i, y_i\}_{i=0}^t$, where $y_i = f(\mathbf{z}_i) + \varepsilon_i$ and $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ represents noise. Subsequently, an acquisition function $\alpha : \mathcal{Z} \rightarrow \mathbb{R}$ is constructed to assign scores to data points in the domain \mathcal{Z} based on their potential to identify a better solution. The next data point for observation is selected by maximizing the acquisition function $\mathbf{z}_{\text{next}} = \arg \max_{\mathbf{z} \in \mathcal{Z}} \alpha(\mathbf{z})$ and is evaluated with the objective function $y_{\text{next}} = f(\mathbf{z}_{\text{next}}) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. This newly observed data $\{\mathbf{z}_{\text{next}}, y_{\text{next}}\}$ is then aggregated to the current dataset. This process is conducted iteratively until the predefined budget, such as the maximum number of function evaluation, is depleted. Finally the best value found in the observed dataset is returned as the estimate of the global optimum \mathbf{z}^* .

A.2 Surrogate Models and Acquisition Functions for Combinatorial and Mixed BO

Surrogate Models. In BO, the most popular surrogate model is Gaussian Process (GP) [37], which depends on the kernel to capture the relationship between the variables. To capture the relationship between different ordinal/categorical variables and between ordinal/categorical and continuous variables, various GP kernels have been proposed, including Graph Diffusion Kernel [32], Additive Diffusion Kernel [6], Overlapped Kernel [39], Transformed Overlapped Kernel [46], Tree Ensemble Kernel [44], and String Kernel [28]. Beyond GP, there are other surrogate models that can naturally represent the categorical data, such as Tree Parzen Estimator [4] and Random Forest [20].

Acquisition Functions. Acquisition functions for combinatorial and mixed BO methods are similar to those in continuous domains, such as Expected Improvement (EI) [27], Upper Confidence Bound [43] and Thompson Sampling [45]. In order to optimize these acquisition functions for combinatorial variables, many methods have been proposed, such as Simulated Annealing [3], Local Search [32, 46, 35], Gradient-based Search [46, 7, 35]. For mixed variables, Interleaved Search [46, 35, 39] is often used to alter between optimizing ordinal/categorical and continuous acquisition functions.

A.3 Categorical Encoders

Ordinal Encoder. Ordinal encoder is a 1-to-1 mapping $g : \mathcal{H} \rightarrow \mathbb{R}^{d_h}$ that assigns each categorical value with a unique numerical value. Given a c -choice categorical variable $h \in \{u_1, u_2, \dots, u_c\}$, the ordinal encoded value of u_i is $g_{\text{ordinal}}(u_i) = a_i$, where $a_i \in \mathbb{R}$ and $a_i \neq a_j$ if $i \neq j$ for $i, j = 1, \dots, c$. Ordinal encoder is a simple choice to transform ordinal/categorical values to numerical values, based on an existing or assumed relationships on the orders and rankings among ordinal/categorical values.

Target Encoder. Target encoder [26] is a 1-to-1 mapping $g : \mathcal{H} \rightarrow \mathbb{R}^{d_h}$ that maps categorical variables to the statistics of the target variable. In practice, the most common statistics to be used for target encoder is the mean value. Denote a c -choice categorical variable $h \in \{u_1, u_2, \dots, u_c\}$. Let the observed dataset of n samples with respect to the variable h be denoted as $D = \{h_i, y_i\}_{i=1}^n$ where y_i is the objective function value. The vector of target values associated with categorical value u_i is defined as $\mathbf{y}_{u_i} = \{y_k | u_i = h_k, k =$

$1, \dots, n\}$. Denote $n_i = \sum_{k=1}^n \mathbb{I}(u_i, h_k)$ as the frequency that u_i exists in the dataset D , where $\mathbb{I}(\cdot, \cdot)$ is an indicator function. The target encoded value of u_i can be computed via its target mean value $\bar{y}_{u_i} = \sum_{k=1}^n (y_i \mathbb{I}(u_i, h_k)) / n_i$, such that, $g_{\text{target}}(u_i) = \frac{n_i \bar{y}_{u_i} + m \bar{y}}{n_i + m}$, where $\bar{y} = \sum_{i=1}^n y_i / n$ is the overall mean of the dataset, and m is a weight factor.

The main goal of target encoder is to map each the categorical value u_i to its expected target value $\mathbb{E}[y_{u_i} | u_i]$. For large datasets, where categorical value u_i appears frequently, i.e., n_i is large, the encoded value $g_{\text{target}}(u_i)$ can be reasonably approximated by the target mean value \bar{y}_{u_i} . However, when n_i is small, the target mean value can become unreliable due to small sample size [26]. To address this, the overall mean value $\mathbb{E}[y] = \bar{y}$ is incorporated with a tuneable weighting factor m , which controls the extent to which the overall mean value is considered when u_i does not have sufficient data. The target encoder formula is derived from the Empirical Bayes estimation theory [38], which uses the empirical prior calculated from the observed data - the expected overall mean value $\mathbb{E}[y]$ - to adjust the empirical posterior computed from the observed data for each category u_i - the expected target value $\mathbb{E}[y_{u_i} | u_i]$.

A.4 CASMOPOLITAN

CASMOPOLITAN [46] is a state-of-the-art high-dimensional BO method for categorical and mixed spaces. To tackle the high dimensions, it employs a local search strategy to define small local regions, called Trust Region (TR) [10] and performs BO within. CASMOPOLITAN deals with categorical and continuous variables separately by maintaining two respective TRs. The continuous TR is a hyper-rectangle centered at the best solution found so far, with its side lengths determined by the GP length-scales multiplied with a length ratio factor L_x . The categorical TR is constructed as a region centered at the best solution found so far, and contain the data points within a Hamming distance of L_h to the TR center. During optimization, CASMOPOLITAN adaptively expands or shrinks the TRs (increases or decreases both L_x and L_h) depending on the success or failure of the algorithm.

When modelling the GP, CASMOPOLITAN proposes to use the Transformed Overlapped Kernel, which can capture the relationship between categorical variables, and also relationship between categorical and continuous variables. Because this kernel uses Hamming distance, CASMOPOLITAN simply uses ordinal encoding to transform categorical data into numerical data for GP modelling.

A.5 Bounce

Bounce [35] is a state-of-the-art high-dimensional BO method for combinatorial and mixed spaces. To tackle the high dimensionality, Bounce employs a subspace embedding method [29, 24, 34], which leverages the random linear embedding to map the d -dimensional input vectors $\mathbf{z} \in \mathcal{Z}$ to $d_{\mathcal{A}}$ -dimensional vectors in a low-dimensional target space \mathcal{A} , meaning $d_{\mathcal{A}} < d$. This random embedding allows for optimization in a low-dimensional subspace \mathcal{A} . During optimization, Bounce adaptively increases the target space dimensionality $d_{\mathcal{A}}$ to allow greater flexibility compared to the fixed choice of $d_{\mathcal{A}}$ as in previous methods [29, 24]. In ordinal, categorical and mixed settings, Bounce only maps variables of the same type together using the same embedding rules. Bounce also employs the local search strategy and maintains two different TRs (with factors L_x and L_h), similar to CASMOPOLITAN. However, the TR management rule of Bounce is different compared to other TR-based methods [10, 34, 46]. Bounce

instantly changes the TR sizes (both L_h and L_x) after each iteration, while other methods require the algorithms to maintain consecutive successes (or failures) to change the TR sizes.

When modelling the GP, Bounce uses one-hot encoder to transform categorical variables to approximate the GP. Bounce leverages the Overlapped Kernel [39] by combining two separate Matern-5/2 kernels to capture the relationship between categorical and continuous variables.

A.6 Experiment Setup

For GP modelling and acquisition function choice, in all derived methods of MOCA-HESP, we use similar surrogate model and acquisition functions as in the respective BO optimizers.

For the HESP mechanism, we use the code implementation that is made available¹ to compute and update the search distribution. We follow [30, 16] to set the population size λ . We further set the lower bound for the standard deviations of ordinal and categorical dimensions, as suggested in [25]. When initializing the search distribution from the observed dataset \mathcal{D}_{0e} , we set the mean vector $m^{(0)}$ as the minimizer of the dataset, the covariance matrix $\Sigma^{(0)}$ as the identity matrix, and the step-size $\sigma = 0.3[u-l]$, where $[l, u]^d$ is the boundary region.

For the categorical encoders, we use the package `category-encoders`² of `scikit-learn` and keep the factor m as default.

For the adaptive encoder selection, we follow [2] to set $\eta = \min\{1, \sqrt{K \ln K / ((1-e)N)}\}$, where N is the maximum number of iteration budget.

We implement MOCA-HESP in Python (version 3.10). We provide in the code supplementary a `.yaml` file to install the required packages for running our proposed methods.

A.7 Baselines

To evaluate the baseline methods, we use the implementation and hyper-parameter settings provided in their public source code and their respective papers. All the methods are initialized with 20 initial data points and are run for 10 repeats with different random seeds. All experimental results are averaged over these 10 independent runs. We then report the best minimum value found.

Standard BO. This is the standard BO method with the TS acquisition function. We apply an ordinal encoder to transform ordinal and categorical variables into continuous variables, which are then modelled via continuous kernels. Specifically, we use the Matern 5/2 ARD kernel to model the transformed continuous variables.

CASMOPOLITAN [46]. We set all the hyper-parameters of CASMOPOLITAN as suggested in their paper. This includes the upper and lower bound for both TR side length $L_{x_{\max}} = 1.6$, $L_{x_{\min}} = 2^{-7}$, $L_{h_{\max}} = d$, $L_{h_{\min}} = 1$ and the TR adaptation threshold $\tau_{\text{succ}} = 3$, $\tau_{\text{fail}} = 40$ where d is the dimension of the problem. For GP modelling, we use Transformed Overlapped kernel. For the input encoder, we use an ordinal encoder. For the acquisition function, we use TS [45]. We use their implementation that is made available at <https://github.com/xingchenwan/Casmopolitan>.

¹ <https://github.com/CMA-ES/pycma>

² <https://pypi.org/project/category-encoders/>

Bounce [35]. We set all the hyper-parameters of Bounce as suggested in their paper. This includes the upper and lower bound for both TR side length $L_{x_{\max}} = 1.6$, $L_{x_{\min}} = 2^{-7}$, $L_{h_{\max}} = d_A$, $L_{h_{\min}} = 1$ and m_D is set to half of the maximum budget. For GP modelling, we use the weighted combination of Matern 5/2 kernels via addition and multiplication. For input encoder, we use one-hot encoder. For acquisition function, we use EI [27] which is optimized via Local Search (or Interleaved Search for mixed cases). We use their implementation that is made available at <https://github.com/LeoIV/bounce>.

CMAES [17]. We use the default settings as suggested in the paper. This includes the population size $\lambda = 4 + \lfloor 3 + \ln d \rfloor$ where d is the problem dimension, the random initial mean vector $m^{(0)}$ selected from the minimum of the 20 random initial points, the identity initial covariance matrix $C^{(0)} = I_d$ and the initial step-size $\sigma^{(0)} = 0.3(u-l)$ where the domain is scaled to uniform bound of $[l, u]^d$. We also activate the restart mechanism of CMA-ES so that the algorithm can restart when it converges to a local minimum. For the input encoder, we use an ordinal encoder. To handle ordinal and categorical variables, we follow Marty et al. and set the lower bound of the standard deviations of the ordinal and categorical dimensions $\sigma_{\text{LB}} = 0.1$. We use their implementation that is made available at <https://github.com/CMA-ES/pycma>.

TPE [4]. We use the implementation and default settings from `hyperopt` Python package. The code is available at <http://hyperopt.github.io/hyperopt/>.

SMAC [20]. We use the implementation and default settings from <https://github.com/automl/SMAC3>.

A.8 Experiments Details

We additionally provide more information on the test problems below. See a summary in Table 2.

Synthetic Benchmark Functions. The problem Ackley20c is the common Ackley function³ with 20 dimensions, in which each dimension is evenly discretized into 11 values within the interval $[-32.768, 32.768]$. This implementation is from [8]. The problem Ackley53m is also an Ackley function with 53 dimensions, which consists of 50 binary variables $[0, 1]^{50}$ mapped to the boundary $[0, 1]^{50}$, and 3 continuous variables $[0, 1]^3$. This implementation has been used in [46, 7, 35].

Real-world Benchmark Problems. The problem Antibody Design finds a string sequence representing the antibody to optimize a binding energy between the antibody and an antigen. The string sequence is represented as a 11D vector of categorical variables, each has 20 different choices. We use the implementation in [8]. The problem LABS (Low-Autocorrelation Binary Sequences) finds a binary sequence to maximize a merit factor. The sequence is represented as a 50D vector of binary variables. We use the implementation in [7, 35]. The problems MaxSAT28 and MaxSAT125 (maximum satisfiability problem) [18] find a 28-dimensional and 125-dimensional binary vector to maximize the combined weights of satisfied clauses. We use the implementation in [32, 46, 7, 35]. The problem Cellular Network has 30 dimensions mixed between 15 continuous values to represent the transmission power and 15 ordinal variables (each with 6 values) to represent the tilting angle of the antennas. The goal is to maximize the coverage quality of the antenna system. We use the implementation in [5].

³ <https://www.sfu.ca/~ssurjano/ackley.html>

For the shifted functions, we follow Papenmeier et al. [35] to permute the categories of the categorical variables by a uniformly random vector $\delta = [\delta_1, \dots, \delta_{d_{\mathcal{H}}}]$ and $\delta_i \sim \mathcal{U}(1, \dots, c_i) \in \mathbb{N}$ where c_i is the number of choices for categorical variables h_i . Therefore, the shifted functions are defined as $f_{\text{shifted}}(\mathbf{h}) = f_{\text{original}}((\mathbf{h} + \delta) \% \mathbf{c})$, where $\mathbf{c} = [c_1, \dots, c_{d_{\mathcal{H}}}]$ and $\%$ is the modulo operator. The motivation for these shifted functions are to remove special structure of the global optimum [35, 30].

A.9 Additional Trajectory Plot

We present the trajectory plot for Ackley-2D function in Fig. 5.

A.10 Computing Infrastructure

We run experiments on a server with a CPU of type AMD EPYC 7R32 (clock speed 3.3GHz). Each experiments are allocated 8 CPUs and 32GB Memory. The server is installed with Operating System Ubuntu 20.04.3 LTS. We use Miniconda (version 23.3.1) to install Python packages.

A.11 Running Time

We report the running time per iteration (in seconds) of our proposed methods and baselines in Table 3. We also report in Table 4 the running time per iteration (in seconds) of MOCA-HESP-BO when using adaptive encoders and fixed encoders (ordinal, target). The results in Table 4 confirm that adaptive selection incurs minimal computational time.

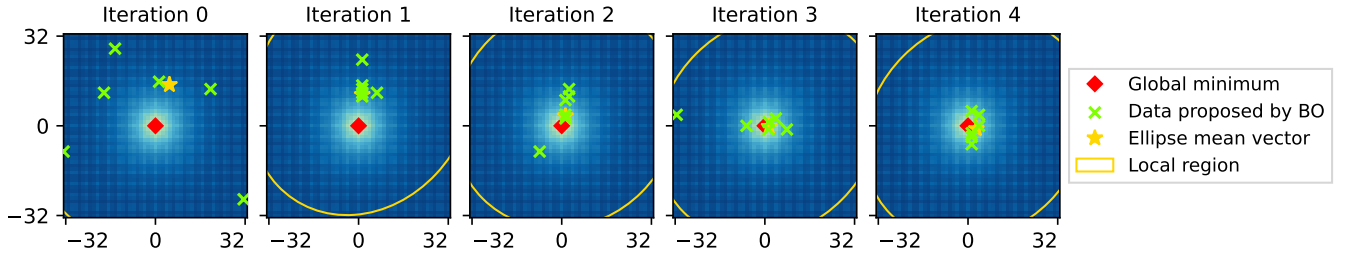


Figure 5. The trajectory of the local regions from MOCA-HESP-BO on 2D Ackley categorical function.

Table 3. Average running time per iteration (in seconds).

	TPE	SMAC	BO	MOCA-HESP-BO	CASMO-POLITAN	MOCA-HESP-Casmo	Bounce	MOCA-HESP-Bounce
Ackley20c	0.03	0.27	0.64	0.80	4.82	6.15	1.23	2.73
Antibody Design	0.01	0.23	0.64	3.23	6.33	10.82	1.75	3.68
LABS	0.06	0.41	2.90	2.46	8.90	9.48	2.19	3.46
MaxSAT28	0.03	0.08	0.63	0.86	5.49	4.06	0.72	1.28
MaxSAT125	0.41	1.00	2.73	3.83	11.78	19.65	3.71	4.44
Shifted Ackley20c	0.03	0.27	0.70	0.79	5.04	6.04	1.25	2.70
Shifted LABS	0.06	0.41	2.80	2.35	9.32	9.53	2.50	3.44
Ackley53m	0.05	0.31	1.74	2.23	18.07	18.17	3.75	15.87
Cellular Network	0.02	0.30	1.20	1.10	7.58	8.40	8.45	17.98

Table 4. Average running time per iteration (in seconds) of MOCA-HESP-BO when using adaptive encoders and fixed encoders (ordinal, target). The results confirm that adaptive selection incurs minimal computational time.

	Adaptive	Ordinal	Target
Ackley20c	0.80	0.62	0.66
Antibody Design	3.23	1.99	1.81
LABS	2.46	2.07	2.20
MaxSAT125	3.83	3.92	3.49
Shifted Ackley20c	0.79	0.64	0.65
Shifted LABS	2.35	1.99	1.80
Ackley53m	2.23	1.84	1.92
Cellular Network	1.10	1.13	0.97