DT-PBO: an Interpretable Tree-based Surrogate Model for Preferential Bayesian Optimization

Nick Leenders[1,2,3*] **Thomas Quadt**[1,3*] **Boris Čule**[3] **Roy Lindelauf**[1,3]
Herman Monsuur[1] **Joost van Oijen**[2] **Mark Voskuijl**[1]

[1] Data Science Center of Excellence, Faculty of Military Sciences, Breda, NL
[2] NLR - Royal Netherlands Aerospace Centre, Amsterdam, NL
[3] Department of Intelligent Systems, Tilburg University, Tilburg, NL

Abstract

Preferential Bayesian Optimization (PBO) aims to find a decision-maker's most preferred solution in as few pairwise comparisons as possible. Most existing approaches use surrogates based on kernel-based methods like Gaussian Processes, which provide strong performance but limited interpretability. This limits real-world usability in high-stakes domains, such as healthcare, where interpretability and trust are essential. We propose DT-PBO, a novel tree-based surrogate model for PBO that is inherently interpretable while capturing preference uncertainty. Specifically, we introduce a novel splitting heuristic that constructs interpretable shallow decision trees directly from pairwise comparison data, and use Laplace approximation to obtain probabilistic estimates for each leaf. This enables efficient preference modeling without sacrificing interpretability. Across eight benchmark functions, our method achieves competitive convergence compared to kernel-based PBO, particularly on functions with rugged optimization landscapes. Additional experiments show robustness against noise and a fast computational running time. Experiments on real-world datasets further demonstrate that our model provides interpretable insights into decision-maker preferences that would remain opaque under kernel-based approaches.

1 Introduction

Preference learning is commonly used in decision-making support, recommender systems, and LLM-tuning to model human preferences. Most methods assume that preferences can be modeled via a latent utility function. This function cannot be accessed directly as it is difficult for a Decision-Maker (DM) to assign numerical ratings to instances [61]. For example, why rate the taste of a slice of cake as 7? Why not 8? Expressing preferences through pairwise comparisons (PCs), like preferring chocolate cake over carrot cake, is cognitively much easier [30, 23]. Preference learning aims to construct a complete model of the preferences. However, often one only wants to find the most preferred solution (e.g. in decision-support). In that case, one can use Preferential Bayesian Optimization (PBO), a subfield within preference learning. PBO is a sample-efficient approach to finding the maximum of an unknown function (i.e. latent utility function) from PCs [22]. A literature review situating PBO within preference learning is provided in Appendix A.

PBO's optimization loop, as shown in Figure 1, constructs a probabilistic surrogate model of the utility function which is used by an acquisition function to identify the most promising next PC to evaluate. First proposed by Chu & Ghahramani [15], Gaussian Processes (GPs), which use a kernel function to measure the similarity between points, have become the standard surrogate model.

*Equal contribution. Corresponding authors: nick.leenders@nlr.nl t.j.quadt@tilburguniveristy.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

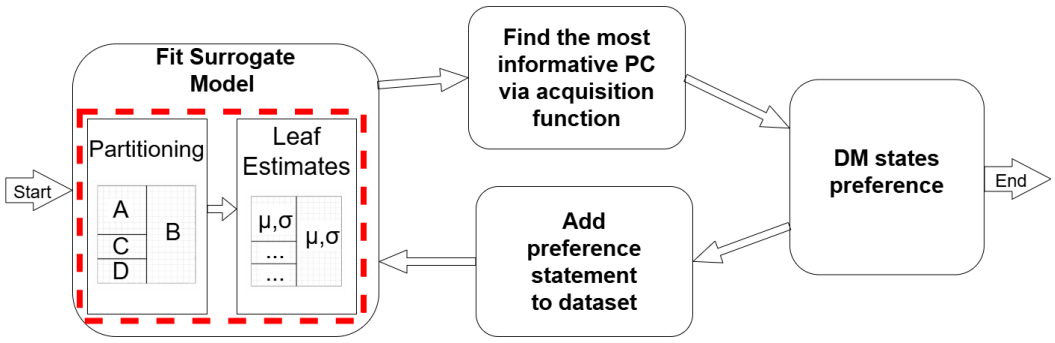

Figure 1: A schematic overview of the PBO loop with the tree-based surrogate model being depicted in the red rectangle.

Besides computational efficiency and needing specialized kernels for handling categorical data (see Appendix A), they suffer from a key drawback. Unless the kernel function is constrained to be additive over the input dimensions [41], they behave as black box models, making it hard for DMs to understand why certain solutions are preferred. In high-stakes domains such as healthcare and defense, decisions must not only be accurate but also interpretable. For example, NATO's strategy for responsible AI places a strong emphasis on explainability [17]. Similarly, the EU's GDPR requires that individuals receive meaningful explanations for automated decisions that significantly affect them [1]. Beyond societal and regulatory considerations, interpretability also plays a crucial role in the performance of a decision support system. By providing insight into model behavior, explanations help DMs better articulate and refine their preferences. Empirical evidence supports this: Chakraborty et al. [11] show that users with access to explanations converge more quickly to their most preferred solutions than those without. Similar findings are reported in Gutierrez et al. [24] and Rodemann et al. [45].

Existing work on explainability for PBO [11, 24, 45] focused on creating post-hoc explanations for GPs. However, post-hoc methods approximate the learned preference function and therefore cannot guarantee faithfulness to the underlying model, potentially leading to incomplete, unstable, or misleading explanations [47]. In preference learning, this is particularly problematic, as DMs rely on these explanations not only to justify model outputs but also to form and refine their own preferences [11]. Inherently interpretable models, on the other hand, are transparent by design, ensuring that the contribution of each input dimension to the learned preference is directly observable and faithful to the model itself. This enables DMs to reason more effectively about trade-offs, adjust their preferences with greater confidence, and interact with the learning system in a more informed manner. Motivated by this, we propose DT-PBO, a Decision Tree (DT) based surrogate model, which, to the best of our knowledge, is the first non-linear interpretable surrogate model for PBO. Besides its inherent interpretability, it is computationally efficient and can naturally handle both quantitative and qualitative data. Our work advances the field of Preferential Bayesian Optimization and Preference Learning through the following key contributions:

1. **Tree-based Preference Learning:** Using a novel splitting heuristic and leaf estimation approach, we develop an algorithm to learn interpretable decision trees *directly* on PC data. The tree has a deterministic structure to maximize *interpretability* with probabilistic leaves that allow for uncertainty quantification and *active learning* (Section 3).

2. **Numerical Evaluation:** We empirically verify the convergence of our model and compare it to kernel-based models. Results indicate competitive convergence, especially on rugged optimization functions, with much lower running times (Section 4).

3. **Real-life Case Studies:** We apply DT-PBO to multiple real-life datasets to illustrate its interpretability (Section 4.2).

Our code is available at `https://github.com/Thomasq99/DT-PBO-NeurIPS`.

2 Background

2.1 Preferential Bayesian Optimization

In PBO, a DM's preferences are modeled by a latent utility function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} = \mathcal{X}_{cont} \times \mathcal{X}_{cat}$, with $\mathcal{X}_{cont} \subseteq \mathbb{R}^{d_c}$, and $\mathcal{X}_{cat} = C_1 \times \ldots \times C_{d_k}$, where each C_i is a finite set of category labels. f measures the degree of preference of a solution $\mathbf{x}$, where if $\mathbf{x}$ is preferred to $\mathbf{x}'$, $f(\mathbf{x}) > f(\mathbf{x}')$. The DM's most preferred solution $\mathbf{x}_{MPS}$ can then be found by solving the global optimization problem $\mathbf{x}_{MPS} = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$. In PBO f is not observed directly. Instead, f is observed via PCs $(\mathbf{x}, \mathbf{x}') \in \mathcal{X} \times \mathcal{X}$ from which we observe preference relations $\mathbf{x} \succ \mathbf{x}'$ indicating that $\mathbf{x}$ is preferred to $\mathbf{x}'$. The goal is to find $\mathbf{x}_{MPS}$ in as few PCs as possible.

To this end, as shown in Figure 1, PBO iteratively selects new pairs to query, updates a probabilistic surrogate model over f based on the observed preferences, and uses this model to guide future comparisons. Specifically, PBO starts with specifying a prior belief over the unknown function f. Then, after observing t preference relations, $\mathcal{D}_t = \{\mathbf{x}_i \succ \mathbf{x}'_i\}_{i=1}^t$, the likelihood $p(\mathcal{D}_t|f)$ and prior $p(f)$ are used to find the posterior $p(f|\mathcal{D}_t)$ via Bayes' rule. This posterior is then used to select the next pair $(\mathbf{x}_{t+1}, \mathbf{x}'_{t+1})$ by maximizing an acquisition function $a(\mathbf{x}, \mathbf{x}' \mid \mathcal{D}_t)$. The DM evaluates the new pair, and the observed preference statement is added to the data. This loop of querying, updating, and selecting new pairs continues until a stopping criterion is met.

2.2 Preference Learning with Gaussian Processes

Let $\mathcal{D}_n = \{\mathbf{x}_i \succ \mathbf{x}'_i\}_{i=1}^n$ be a set of n preference relations constructed from m unique instances $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathcal{X}$. The main idea is to define a GP prior over the latent function f, use a pairwise likelihood function, and find the posterior using Bayes' rule.

GP Prior: A Gaussian Process [57] defines a distribution over functions $f \sim \mathcal{GP}(m(\mathbf{x}), \mathcal{K}_\theta(\mathbf{x}, \mathbf{x}'))$, where $m(\mathbf{x})$ is the mean function and $\mathcal{K}_\theta(\mathbf{x}, \mathbf{x}')$ is a parametrized kernel function encoding assumptions about the smoothness or form of the function f. In preference learning, zero-mean GP priors are used, which model the latent function values $f(x_i)$ as a multivariate joint Gaussian

$$p(\mathbf{f}) = \frac{1}{(2\pi)^{\frac{m}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\mathbf{f}^\top \Sigma^{-1} \mathbf{f}\right), \tag{1}$$

where Σ is an $m \times m$ covariance matrix with $\Sigma_{ij} = \mathcal{K}_\theta(\mathbf{x}_i, \mathbf{x}_j)$, and $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m)]^\top$.

Pairwise Likelihood: Assuming that observations are influenced by homoskedastic Gaussian noise and assuming independence of preference statements, following Benavoli & Azzimonti [6] the likelihood of observing n preference relations $\mathcal{D}_n$ given $\mathbf{f}$ is

$$p(\mathcal{D}_n|\mathbf{f}) = \prod_{i=1}^{n} \Phi\left(\frac{f(\mathbf{x}_i) - f(\mathbf{x}'_i)}{\sqrt{2}\sigma_{noise}}\right), \tag{2}$$

where $\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(t; 0, 1)dt$ is the cumulative standard normal distribution.

Posterior: The posterior can be found using Bayes' rule $p(\mathbf{f}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{f})p(\mathbf{f})}{p(\mathcal{D})}$, where the prior $p(\mathbf{f})$ is defined in Eq. 1, the likelihood $p(\mathcal{D}|\mathbf{f})$ is defined in Eq. 2, and the normalization factor $p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{f})p(\mathbf{f})d\mathbf{f}$. The latter is only needed for model selection. This posterior is a computationally intractable skew GP [7]. To deal with this, one can approximate the posterior with a Gaussian using, for example, Laplace Approximation (LA) [15], Expectation Propagation [36], or use Markov Chain Monte Carlo methods to employ the skew GP directly. In this research, LA is used because of its computational simplicity. The use of different approximation methods is left for future research.

Using LA, the posterior is approximated by a Gaussian centered at the mode (i.e., maximum a posteriori (MAP) estimate) of the posterior [15]. In particular, $\mathbf{f}|\mathcal{D} \sim \mathcal{N}(\mathbf{f}_{MAP}, \Sigma_{post})$ where $\mathbf{f}_{MAP} = \arg\max_{\mathbf{f}} p(\mathbf{f}|D)$, which can be found as minimizer of the following functional

$$L(\mathbf{f}) = -\sum_{i=1}^{n} \ln \Phi(\frac{f(\mathbf{x}_i) - f(\mathbf{x}'_i)}{\sqrt{2}\sigma_{noise}}) + \frac{1}{2}\mathbf{f}^\top \Sigma^{-1}\mathbf{f}. \tag{3}$$

The posterior covariance Σ_{post} is the inverse of the Hessian of $L(\mathbf{f})$ evaluated at the MAP estimate

$$\Sigma_{post} = H(L(\mathbf{f}))^{-1}|_{\mathbf{f}=\mathbf{f}_{MAP}} = (\Sigma^{-1} + \Lambda_{MAP})^{-1}, \tag{4}$$

where Λ is an $m \times m$ matrix where the ijth element is $\frac{\partial^2 \sum_{k=1}^{n} - \ln \Phi(z_k)}{\partial f(x_i) \partial f(x_j)}$, $z_k = \frac{f_{MAP}(\mathbf{x}_k) - f_{MAP}(\mathbf{x}'_k)}{\sqrt{2}\sigma_{noise}}$.

2.3 Extending Decision Trees to Pairwise Comparison data

A simplistic approach to training a DT with PCs is to frame it as a classification problem, where the tree is trained on *(Item A, Item B, User prefers A)* as done by Qomariyah et al. [42]. However, this leads to an asymmetrical model which could be solved by training on both *(Item A, Item B, User prefers A)* and *(Item B, Item A, User prefers A)* for every single comparison. This doubles the dataset and forces the tree to learn two separate rules for the same underlying preference, which is highly inefficient. Alternatively, as done by Shavarani et al. [49], one could train directly on the difference in feature vectors *(Item A - Item B, User prefers A)*. This leads to a loss of context of the absolute feature values. Most importantly, all these approaches provide only local interpretability: one can determine when one item is preferred over another but global interpretability is not possible (one cannot easily see which item is overall the best or second-best). Finally, neither of these methods provide a probabilistic estimate and they are therefore not suitable for active learning. Shavarani et al. [48] do extend their method by using Random Forests which allow for active learning but greatly reduces the interpretability.

Somewhat similar to learning from PCs is learning from ranking data, i.e., data where items are ordered by preference. Cheng et al. [12] propose a DT that uses a ranking probability model (Mallows model) as splitting heuristic. This method is suitable for active learning, although computationally expensive. However, it is made for label preferences, where the training data consists of PCs among labels associated with the objects, whereas in our research we focus on object preferences, where the training data consists of PCs among the characteristics or attributes associated with the objects [6]. Similarly, Rebelo & Soares [44] build a tree directly from ranking data but their method is correlation-based instead of probabilistic, making it unsuitable for active learning. Finally, Liu & Shih [33] transform the PCs into a scoring vector using a scoring system and reframe the preference learning problem as a multi-response regression task where the higher the score, the more preferred the object is. This method also lacks a probabilistic estimate and is therefore not suitable to active learning. Existing tree-based solutions to ranking or PCS are thus not suitable for active learning on object preferences. At the same time, probabilistic extensions of DTs, such as Bayesian DTs (e.g., [13, 39]) exist but are not applicable to PC data. To the best of our knowledge, a globally interpretable DT that can be used for active learning and learns directly on PC data does not yet exist.

3 Method: DT-PBO

We propose DT-PBO, which as shown in Figure 1 follows the same optimization loop as GP-based PBO but uses a tree-based surrogate model instead. This tree is constructed from PCs in two steps. First, the feature space is partitioned into a set of leaves using a novel splitting heuristic. Second, for each leaf a Gaussian distribution is fitted to represent the prediction estimate. The resulting tree has a deterministic structure with probabilistic Gaussian distributions as leaves. Whilst Bayesian tree-structures are possible, they reduce the interpretability of the method and are thus not used in this research. Nevertheless, future work could focus on extending our method to Bayesian tree structures. Using the probabilistic leaves, the Expected Utility of Best Option (qEUBO) [3] acquisition function finds the most informative PC to ask. Pseudocode of DT-PBO is shown in Appendix B.

3.1 Partitioning the Space

First, to learn a single regression tree with probabilistic leaves from PC data, we need a new splitting heuristic. The idea proposed here is the Consistency Score (defined in Eq. 6), a metric designed

to find a split between winners and losers from PCs. Consider a node corresponding to a region $A \subseteq \mathcal{X}$ and let $D_A = \{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^{N_A}$ denote the set of comparisons for which both winner and loser lie in A. A split is defined by a feature index k and, for numerical features, a threshold value t or, for categorical features, a subset of categories B, where $\mathbf{x}[k]$ is the kth feature value. For a numerical feature, candidate thresholds are taken as midpoints between consecutive distinct values: $\mathcal{T}_{A,k}^{\text{num}} = \left\{ \left(k, \frac{u_i + u_{i+1}}{2} \right) : i = 1, \ldots, |U_{A,k}| - 1 \right\}$, where $U_{A,k} = \{u_1 < u_2 ... < u_m\}$ denotes the set of sorted unique values $\mathbf{x}[k]$ for all $\mathbf{x} \in D_A$. For a categorical feature, let $C_{A,k}$ contain the categories observed in D_A. A split (k, B), with $\varnothing \subsetneq B \subsetneq C_{A,k}$, induces the child regions $A_L(B) = \{\mathbf{x} \in A : x[k] \in B\}$, $A_R(B) = \{\mathbf{x} \in A : x[k] \notin B\}$. With $c = |C_{A,k}|$, we enumerate all $2^{c-1} - 1$ distinct ways of dividing the categories into two non-empty groups. These splits form $\mathcal{T}_{A,k}^{\text{cat}}$. For a numerical candidate split $s = (k, t)$, define

$$Z_i(s) = \mathbf{1}\{x_i[k] \geq t,\ x'_i[k] < t\} - \mathbf{1}\{x_i[k] < t,\ x'_i[k] \geq t\}, \tag{5}$$

so that $Z_i(s) \in \{-1, 0, 1\}$: $+1$ has the winner on the right and the loser on the left of the split, -1 the reverse, and 0 has both items on the same side. The empirical consistency score is:

$$\widehat{S}_A(s) = \left| \frac{1}{N_A} \sum_{i=1}^{N_A} Z_i(s) \right|. \tag{6}$$

which rates how well a split aligns with observed preferences. Its population counterpart is

$$S_A(s) = \left| \mathbb{E}[Z(s) \mid X \in A,\ X' \in A] \right|, \tag{7}$$

where $S_A(s) = |p_+ - p_-|$ with $p_\pm = \Pr(Z(s) = \pm 1 \mid X, X' \in A)$. It equals 1 for a split that perfectly separates winners from losers and 0 for a split completely uncorrelated with preferences.

The tree is constructed greedily by selecting $s_A^\star = \arg\max_{s \in \mathcal{T}_A} \widehat{S}_A(s)$, where $\mathcal{T}_A = \mathcal{T}_A^{\text{cat}} \cup \mathcal{T}_A^{\text{num}}$ denotes the finite set of candidate splits considered at node A. This method differs from the approaches mentioned in Section 2.3, which are based on ranking algorithms. Rather than transforming the data, our method trains directly on preference pairs $(\mathbf{x}, \mathbf{x}')$.

Selecting $s_A^\star$ greedily from the empirical score $\widehat{S}_A(s)$ is only meaningful if $\widehat{S}_A(s)$ is a reliable estimate of the population score $S_A(s)$, otherwise the chosen split may reflect noise rather than the underlying preference structure. Assume that for a fixed node A and candidate split $s \in \mathcal{S}_A$, the random variables $Z_1(s), \ldots, Z_{N_A}(s)$ are independent and per definition $Z_i(s) \in [-1, 1]$. The following proposition bounds this estimation error uniformly over the candidate set $\mathcal{T}_A$. This is a concentration result rather than a direct guarantee on split selection: translating it into a correct-selection statement additionally requires a margin between the best and second-best population scores, and this is therefore best read as a heuristic motivation for the consistency score.

Proposition 1 (Error Bound for the Consistency Score)**.** *For any tolerance $\varepsilon > 0$, the probability of the empirical score deviating from the true score for a specific split s is bounded by:*

$$\Pr\left(\left| \widehat{S}_A(s) - S_A(s) \right| \geq \varepsilon \right) \leq 2 \exp\left(-\frac{N_A \varepsilon^2}{2} \right).$$

Consequently, by a union bound over the finite set of candidate splits $\mathcal{T}_A$:

$$\Pr\left(\sup_{s \in \mathcal{T}_A} \left| \widehat{S}_A(s) - S_A(s) \right| \geq \varepsilon \right) \leq 2|\mathcal{T}_A| \exp\left(-\frac{N_A \varepsilon^2}{2} \right).$$

Proof. This result follows from Hoeffding's inequality for bounded independent random variables, followed by a union bound over $|\mathcal{T}_A|$. The complete proof is provided in Appendix C.1. □

Adding features only slowly increases the estimation error of the split scores, as $O(\sqrt{\log d / N_A})$ when the number of candidate splits per feature is bounded. Appendix C.1 discusses the effect of the problem's dimensions on the bound in more detail.

After selecting the best split $s_A^\star = (k^\star, t^\star)$, the two child regions are

$$A_L = \{\mathbf{x} \in A : x[k^\star] < t^\star\}, \qquad A_R = \{\mathbf{x} \in A : x[k^\star] \geq t^\star\}.$$

The child-node datasets used for further recursive split selection are

$$D_{A_L} = \{(\mathbf{x}, \mathbf{x}') \in D_A : \mathbf{x} \in A_L,\ \mathbf{x}' \in A_L\}; D_{A_R} = \{(\mathbf{x}, \mathbf{x}') \in D_A : \mathbf{x} \in A_R,\ \mathbf{x}' \in A_R\}. \quad (8)$$

Pairs for which one item lies in A_L and the other in A_R are called straddlers. These pairs are excluded from descendant split selection but retained for final leaf parameter fitting, where their separated items receive distinct rankings. This exclusion reduces the dataset size internally which self-controls the growth of the DT thereby favoring smaller and more interpretable trees. This exclusion is mathematically justified by two key properties (formal definitions and proofs are in Appendix C):

- **Correct Conditional Objective** (Proposition 3): Descendant nodes must solve a conditional subproblem focused strictly on items within the same region. Retaining straddlers would distort the empirical score, targeting a mixture of within-child and cross-child effects rather than the true conditional estimated value.
- **Lossless Exclusion** (Corollary 1): When the parent split gap dominates the residual spread across sides, the parent split alone determines the preference between straddling items, and straddlers carry no information about the child nodes' internal structure.

Consequently, under these conditions straddlers contain information about the parent ordering only, and excluding them costs no useful information for the lower branches. When the condition fails, exclusion becomes a controlled approximation that trades some descendant-level signal for the regularizing effect of smaller, more interpretable trees.

3.2 Estimating the Leaf Parameters

After observing n preference relations, the latent utility function is represented by m_l leaf values $\mathbf{f} = [f_1, \ldots, f_{m_l}]^T$, where each f_j denotes the mean of the Gaussian distribution of leaf j. To find the probabilistic leaf estimates, we model them as a joint multivariate Gaussian. To do so, we employ a Bayesian approach. As prior, we assume that each leaf f_j follows an independent zero-mean normal distribution with variance σ^2_{prior}. This leads to the following joint prior

$$\mathbf{f} \sim \prod_{j=1}^{m_l} \mathcal{N}(0, \sigma^2_{prior}) = \mathcal{N}(\mathbf{0}, \sigma^2_{prior} I), \text{where } I \text{ is the } m_l\text{-dimensional identity matrix.} \quad (9)$$

We approximate the posterior with a Gaussian centered at the MAP estimate $\mathbf{f}|\mathcal{D} \sim \mathcal{N}(\mathbf{f}_{MAP}, \Sigma_{post})$ via LA. We use the same pairwise likelihood function defined in Eq. 2 as in the GP case but replace the GP prior, with the independent normal prior defined in Eq. 9. Therefore, to find the MAP-estimate $\mathbf{f}_{MAP}$ as the minimizer of the functional $L(\mathbf{f})$, we need only change the prior term in Eq. 3. The new functional $L(\mathbf{f})$ and posterior covariance Σ^{-1}_{post} are then defined as

$$L(\mathbf{f}) = -\sum_{i=1}^{n} \ln \Phi\left(\frac{f(x_i) - f(x_i')}{\sqrt{2}\sigma_{noise}}\right) + \frac{1}{2\sigma^2_{prior}} \mathbf{f}^\top \mathbf{f}.$$

$$\Sigma^{-1}_{post} = H(L(\mathbf{f}))|_{\mathbf{f}=\mathbf{f}_{MAP}} = \frac{1}{\sigma^2_{prior}} I + \Lambda_{MAP}. \quad (10)$$

This method has two hyperparameters σ_{noise} and σ_{prior}. However, by rewriting $L(\mathbf{f})$ it becomes clear that σ_{noise} and σ_{prior} are not identifiable separately (see Appendix D). Instead, the optimization of $L(\mathbf{f})$ depends only on the ratio $\sigma^2_{noise}/\sigma^2_{prior}$. Additionally, when both items of a comparison are in the same leaf, they share the same latent value f_j, resulting in $P(\mathbf{x} \succ \mathbf{x}') = \Phi(0) = 0.5$ for each value f. Such pairs provide no gradient information on $L(\mathbf{f})$ w.r.t. f, do not influence the MAP-estimate and are thus discarded from the computation.

Finally, the pairwise likelihood in Eq. 2 is *translationally invariant* with respect to $\mathbf{f}$. Adding a constant shift c to all components of $\mathbf{f}$ does not influence the pairwise differences and thus leaves the likelihood unchanged. This causes a large irreducible variance term in the constant shift direction for the uncertainty estimate. To remedy this, the model is made identifiable by adding a sum-to-zero

constraint: $\sum_{i=1}^{m} f = \mathbf{1}\mathbf{f} = 0$. This fixes the scale of the model, preventing a constant shift, and thereby removing the unwanted variance (of σ^2_{prior}/m_l for every item of the covariance matrix). This change does not affect the acquisition function (only based on pairwise differences) nor the mean estimates in the leaves. This constraint is incorporated by Eaton's [16] normal conditional distribution result.

Proposition 2 (Conditioned Posterior)**.** *By conditioning the posterior on the sum-to-zero constraint* $\mathbf{1}^\top \mathbf{f} = 0$*, the posterior follows the conditional distribution* $p(\mathbf{f}|\mathbf{1}^\top \mathbf{f} = 0, \mathcal{D})$ *given by:*

$$\mathbf{f} \mid \mathbf{1}^\top \mathbf{f} = 0 \;\sim\; \mathcal{N}\Big(\mathbf{f}_{MAP}, \Sigma_{\text{post}} - \frac{\sigma^2_{\text{prior}}}{m_l}\mathbf{1}\mathbf{1}^\top\Big)$$

Proof. An extensive derivation of this conditional distribution is provided in Appendix E. □

Note that one could also use a Bradley-Terry likelihood model [10] which only affects the Laplace approximation. Appendix F.5, however, shows that the change does not significantly influence the performance of DT-PBO.

3.3 Active Learning

Similar to regression or classification problems, acquisition functions are used for active learning from PCs. However, instead of evaluating the most informative point, the most informative PC has to be chosen. We use qEUBO [3], an acquisition function specialized for PCs that has generally achieved strong results in PBO. In Appendix F.6, we report results using other specialized preference learning acquisition functions. None of the methods clearly outperforms the others for all benchmark problems, however, all of them show significant improvements over random query selection, showing the benefit of active learning. As we only use PCs ($q = 2$), there is an analytical definition of the acquisition function available. For each candidate pair $(\mathbf{x}, \mathbf{x}')$ the leaf-wise Gaussian mean and uncertainty estimates are used to evaluate qEUBO(x, x'), and the next pair is selected as $((\mathbf{x}_{t+1}, \mathbf{x}'_{t+1}) = \arg\max_{(\mathbf{x},\mathbf{x}')\in\mathcal{X}\times\mathcal{X}} qEUBO(\mathbf{x}, \mathbf{x}'))$.

DT-PBO rebuilds the tree from the updated comparisons at each iteration, so its structure can change. Conditional on the current tree $\mathcal{T}^*$, qEUBO accounts for posterior uncertainty in leaf utilities, but not over alternative tree structures. Appendix F.10 examines whether conditioning on a deterministic tree leads to overly exploitative queries and finds no evidence of this on Michalewicz-2D. Within a leaf, both items share the same latent utility, so qEUBO cannot distinguish pairs to guide structural refinement. Appendix C.3 formalizes this refinement through $\alpha_{\text{DT}} = \text{qEUBO} + \lambda I_{\text{struct}}$, where I_{struct} is the expected absolute change in the within-leaf consistency score. For between-leaf pairs, $I_{\text{struct}} = 0$, preserving standard qEUBO. Our top-leaf-prioritization rule is interpreted as the $\lambda \to \infty$ limit with the structural term restricted to the highest-mean leaf, focusing refinement where the optimum is believed to lie. We use this rule only in discrete item-set experiments shown in Section 4.2. Synthetic benchmarks use qEUBO alone ($\lambda = 0$) without within leaf pairs. Once a leaf pair is selected, items are sampled uniformly within the selected leaves.

4 Experiments

4.1 Comparison to State-Of-The-Art PBO Methods

We compare our method to two GP-based models, SkewGP with Hallucination Believer - Expected Improvement (HB-EI) [53] and LA-based GP with qEUBO [3], and one frequentist approach with a GP-like kernel, POP-BO [58]. Comparing with skewGP-qEUBO is not possible because to be computationally feasible SkewGP needs a hallucination step to condition the skewGP on the hallucination point, creating an ordinary GP, which is computationally much easier to use. POP-BO employs its own UCB-like acquisition function which is used. For the baselines' kernels we use the paper's proposed kernels (RBF for GP-based and Matern(5/2) for POP-BO. Detailed model settings are in Appendix F.1. Below, the performance is compared by simulating a DM's noiseless utility function using eight 2D to 6D benchmark functions. In Appendix F.8, we show results using noisy DMs. In Appendix F.2 we describe each function in detail, along with their ruggedness, a term used to describe the degree of local curvature. Each model tries to find the maximum of the utility function by querying PCs from it. The performance is averaged over 20 runs by varying the initial

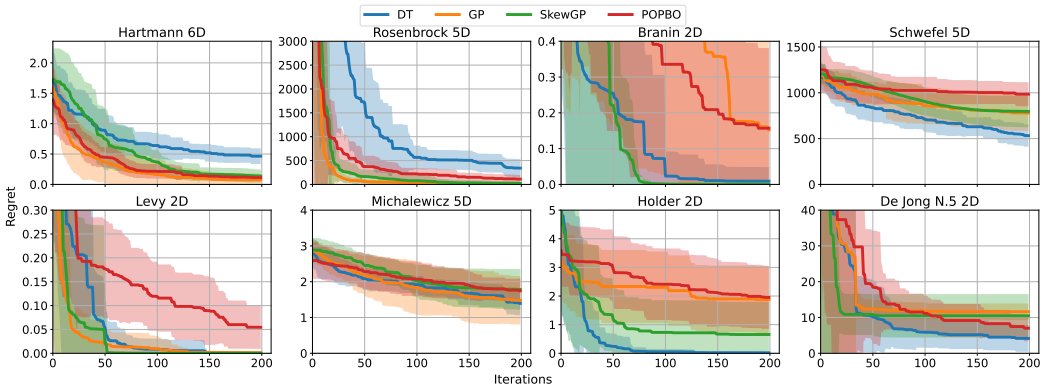

Figure 2: Average Regret over 20 runs for the DT-PBO, GP-qEUBO, SkewGP with HB-EI, and POP-BO models. The methods are tested using eight common benchmark functions as artificial DMs.

questions asked. As a performance metric we use the regret which is defined as $f(x^*) - f(\hat{x}_t)$, where $f(x^*)$ is the optimum (maximum) of function f and $f(\hat{x}_t)$ is the model's recommendation point's mean utility value at timestep t. The recommendation point x_t is the point $x \in \mathcal{D}_t$ with the highest predicted mean utility so far. All models start with 20 initial comparisons sampled via Latin Hypercube Sampling [35] and query 200 PCs from the utility function.

In Figure 2, we present convergence plots across eight benchmark functions. The results indicate that the tree-based model performs slightly worse on Rosenbrock and Hartmann, both non-rugged functions, where its convergence is more than one standard deviation (std.) worse than that of the baselines. For Branin and Levy, both moderately rugged functions and Michalewicz, a rugged function, the DT model performs competitive, remaining within one std. of the best-performing baseline. On the De Jong and Holder functions, both rugged functions and Schwefel, a moderately rugged function, the DT model outperforms the kernel-based models. Overall, these results suggest that the proposed model offers a competitive alternative to state-of-the-art PBO methods, with particularly strong performance on more rugged functions. Additional numerical experiments in Appendix F show that the $ratio$-hyperparameter is robust to small changes, that the running time is faster (with a factor of 10, 400, 1000, for GP, skewGP, POP-BO respectively), that DT-PBO struggles for non-rugged functions in high dimensions, that DT-PBO is robust to noise in preferences, and that shallow trees of depth 4-6 are usually enough to get optimal regret. Moreover, Appendix F.10 investigates exploration-exploitation behavior and Appendix F.11 extends DT-PBO to preference statements consisting of more items and reports results on Sushi dataset A and Michalewicz-5D.

To assess the interpretability of the learned trees, we measure their size: across the eight benchmarks, the final DT-PBO models have mean leaf counts of 12.0–20.8, mean depths of 4.9–7.4, and volume-weighted mean decision-path lengths of 3.7–4.5 conditions (Appendix F.12). To put these sizes in context, Appendix F.13 fits post-hoc regression trees to GP posterior means: these surrogate trees must be much larger (sometimes over 4096 leaves) to reach 95% pairwise fidelity.

4.2 Case Studies: Showing Interpretability in Real-Life Datasets

We report two real-world data studies, each serving a distinct purpose. The sushi dataset is a commonly used real-world benchmark for PBO, introduced by [37], and is one of the few publicly available real-world datasets for this setting. We use it to evaluate whether DT-PBO can recover a user's most preferred sushi from pairwise queries. However, sushi recommendation does not have a high need for interpretability. To illustrate where interpretability matters most, we include a second study on Patient Message Ranking (PMR). PMR is not a PBO problem in the strict sense, but a high-stakes preference-learning task in which the ranking affects which patients receive clinical attention first. In such settings, competitive predictive performance alone is not sufficient: clinicians need to understand which message and Electronic Health Record (EHR) features are driving urgency. Additional case studies can be found in the Appendix. Specifically, in Appendix G.3 we report a case study on the BELA [34] dataset. The dataset contains preference statements of respondents being asked to choose between two breast cancer treatment profiles. The resulting tree found in Figure 27

Table 1: Training and test accuracy for the sushi and PMR dataset.

	Train acc.	Test acc.
Sushi		
DT	0.998	0.888
GP	0.997	0.895
PMR		
Pure LLM	0.703	0.650
LLM + DT	0.840	0.699

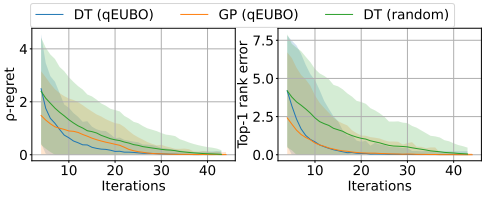

Figure 3: ρ-regret and top 1 error for sushi preference without user features for DT and GP.

shows one person's learned preferences. In Appendix G.4, we show a case study on the presidential candidates dataset [5]. The dataset consists of preferences over presidential candidates. One can find the learned tree in Figure 28

Case Study 1: PBO on Sushi preferences In the sushi dataset [28], users were given 10 sushi and were asked to rank them from 1 to 10. Sushi features contain continuous features, such as oiliness, and categorical features, such as whether the sushi contains seafood, meat, etc. (for details see Appendix G). We converted the ranking to PCs for compatibility with PBO. The goal is to find a specific user's most preferred sushi in as few PCs as possible. To measure performance, a regret function similar to the ρ-regret in Husslage et al. [26] is used, but instead of a top five, a top three is chosen. This regret penalizes the model when items ranked outside the top three appear within the top three. A detailed explanation and example of the ρ-regret are in Appendix G.1. Additionally, we report the top-1 rank error: the number of items that the model ranks above the true best item. A score of 0 indicates that the model correctly identifies the user's favorite, while higher values indicate that inferior items are ranked above it. We compare the performance of our model with qEUBO to our model with randomly selected PCs, and with GP-qEUBO. Additionally, in Appendix G.1 we extend DT-PBO to handle user-features.

As shown in Figure 3, DT-PBO obtains a low ρ regret faster than GP-qEUBO. Furthermore, both GP and DT-PBO, identify the most preferred sushi in less than 9 questions in $\approx 70\%$ of the cases, the number that a bandit-based algorithm would require. Beyond needing less questions, the DT also yields an interpretable preference model that a bandit approach cannot provide: rather than only identifying the best-ranked sushi, the tree exposes which sushi attributes, such as oiliness or seafood content, drive the user's choices, which the user can then inspect to see whether it accurately reflects their taste (see Figure 22 in Appendix G). Because these preferences are expressed over sushi features the tree can also generalize to other unseen sushi. To show this, a train (80%) and test (20%) split is created for each user in the sushi dataset. Both GP and DT get a high test accuracy of 0.9 (see Table 1) which indicates good generalization to unseen sushi.

In Appendix G.1 we also compare our method with the static score-scale method [33]. On all 5,000 users, DT-PBO attains lower mean ρ-regret at all four comparison budgets, with both methods fit to the same randomly sampled PCs for each user. Since the data is identical, this gain comes from the model, as splitting on features lets each comparison inform all sushi that share them.

Case study 2: Patient Message Ranking PMR aims to help clinicians prioritize incoming patient messages according to medical urgency. By optimizing the order of messages in a clinician's inbox, PMR determines which patients receive earlier responses [20]. PMR-Synth, a publicly available PMR dataset, contains pairwise comparisons of synthetic patient portal messages combined with real EHR data, emulating a clinical setting in which urgency is assessed by clinical experts on the basis of both message content and structured patient records. The dataset consists of `INBOX_A`, which is used for training, and `INBOX_B`, which is used for validating performance. These contain 30 unique patients and 434 messages each. Gatto et al. [20] address this task by fine-tuning Large Language Models (LLMs) directly on the pairwise comparison data. Although this approach achieves strong test performance of maximum 0.72 for a non-finetuned LLM and 0.73 for a finetuned LLM, the resulting model remains a black box. In a setting where prioritization decisions may have serious clinical consequences interpretability is essential. Our proposed alternative is to use an LLM as a feature extraction tool instead of as the final decision-maker. Ideally, such features would be specified

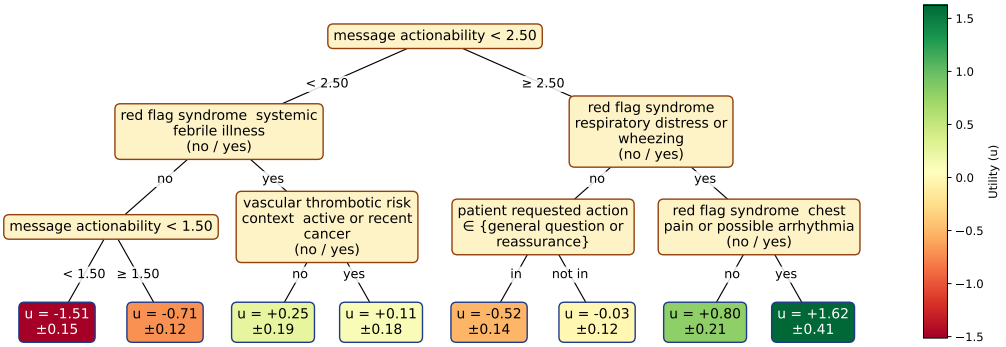

Figure 4: Learned preference tree for PMR.

directly by domain experts. In this study, however, we use an LLM (`Qwen3.5-27B`) to extract and propose candidate features. The resulting feature set and the extraction procedure are described in Appendix G.2. Note that the goal of this case study is not to present a new end-to-end triage system, but rather to show that an interpretable preference model can remain competitive in a high-stakes setting while revealing the factors that drive its decisions. Therefore, we make no claim about the clinical validity of the extracted features or of the learned decision rules; establishing this requires domain experts. DT-PBO makes such an assessment possible: a clinician can inspect the learned tree directly, judge whether its splits are clinically plausible, and flag or correct them, which is not possible with an end-to-end LLM ranker. The resulting tree (see Figure 4) is pruned for readability. The full tree is provided in Figure 26. The training and test accuracies are reported in Table 1, where our pure LLM (using `Qwen3.5-27B`) approach achieves similar results to Gatto et al. [20]. It should be noted that here the pure LLM's train and test score is just it's performance on `INBOX_A` and `INBOX_B` respectively. The results show that our method has a competitive performance compared to the pure LLM approach, while the model itself becomes more interpretable. One limitation of this approach is that the final model quality depends on the reliability of the LLM-based feature extraction step. The tree can be used to identify which combinations of symptoms and EHR-derived features are associated with higher clinical priority. Figure 4 shows that the learned model prioritizes patients with high actionability, respiratory distress or wheezing, and chest pain.

5 Conclusions

We introduced DT-PBO, a novel tree-based surrogate for PBO that provides an interpretable alternative to black-box GP models. A new splitting heuristic learns trees directly from pairwise comparisons, and excluding straddlers during recursion yields shallow trees without needing growth-controlling hyperparameters. Using a Laplace approximation, we obtain a joint Gaussian distribution over leaves, enabling probabilistic estimates for active learning. Across eight benchmark functions, DT-PBO achieves competitive convergence, with particularly strong performance on rugged landscapes. Case studies further demonstrate its interpretability and ability to capture individual preferences. Besides inherent interpretability and competitive performance, DT-PBO is computationally much more efficient and can handle categorical data naturally whereas GPs need specialized kernels. Therefore, we advise readers to use DT-PBO when interpretability matters, when objects of the pairwise comparisons have categorical features or when computational efficiency is important. Limitations include (1) the inability to model indifference statements ($A \sim B$), (2) our model's struggle with non-rugged functions as the number of dimensions increases ($D > 7$), and (3) not being able to account for uncertainty in the tree structure due to the deterministic partitioning step. Limitation 2, however, rarely happens in practice as Kivikangas et al. [29] have shown that pairwise comparisons are only cognitively valid for human decision-makers up to 5-dimensions. Future work can address limitation 1 by allowing the model for including indifference statements by, for example, by including a threshold below which the utility difference of two items renders a preference statement indifferent. Limitation 3 can be addressed, at the cost of reduced interpretability, by using fully Bayesian trees. Finally, our quantitative interpretability evaluation relies on proxies such as tree complexity and fidelity. Future work could include human- and application-grounded evaluation.

Funding Statement

This research is funded by the Chair of Data Science, Safety & Security at Tilburg University, and NLR Royal Netherlands Aerospace Centre.

References

[1] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (text with EEA relevance), May 2016.

[2] Nir Ailon, Zohar Karnin, and Thorsten Joachims. Reducing dueling bandits to cardinal bandits. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 856–864, Beijing, China, 22–24 Jun 2014. PMLR.

[3] Raul Astudillo, Zhiyuan Jerry Lin, Eytan Bakshy, and Peter Frazier. qEUBO: A decision-theoretic acquisition function for preferential Bayesian optimization. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 1093–1114. PMLR, 25–27 Apr 2023.

[4] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo Bayesian optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21524–21538. Curran Associates, Inc., 2020.

[5] Kirk Bansak, Jens Hainmueller, Daniel J Hopkins, and Teppei Yamamoto. The number of choice tasks and survey satisficing in conjoint experiments. *Political Analysis*, 26(1):112–119, 2018.

[6] Alessio Benavoli and Dario Azzimonti. A tutorial on learning from preferences and choices with gaussian processes. *Foundations and Trends® in Machine Learning*, 19:1–120, 04 2026.

[7] Alessio Benavoli, Dario Azzimonti, and Dario Piga. Preferential Bayesian optimisation with skew Gaussian processes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, page 1842–1850, New York, NY, USA, 2021. Association for Computing Machinery.

[8] Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: A survey. *Journal of Machine Learning Research*, 22(1), January 2021.

[9] Viktor Bengs, Aadirupa Saha, and Eyke Hüllermeier. Stochastic contextual dueling bandits under linear stochastic transitivity models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1764–1786. PMLR, 17–23 Jul 2022.

[10] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[11] Tanmay Chakraborty, Christian Wirth, and Christin Seifert. Comparative explanations: Explanation guided decision making for human-in-the-loop preference selection. In Riccardo Guidotti, Ute Schmid, and Luca Longo, editors, *Explainable Artificial Intelligence*, pages 139–161, Cham, 2026. Springer Nature Switzerland.

[12] Weiwei Cheng, Jens Hühn, and Eyke Hüllermeier. Decision tree and instance-based learning for label ranking. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 161–168, Montreal Quebec Canada, June 2009. ACM.

[13] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayesian cart model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

[14] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 844–853. PMLR, 06–11 Aug 2017.

[15] Wei Chu and Zoubin Ghahramani. Preference learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 137–144, New York, NY, USA, 2005. Association for Computing Machinery.

[16] Morris L. Eaton. *Multivariate statistics : a vector space approach*, pages 116–117. Wiley series in probability and mathematical statistics. Wiley, New York, NY, 1983.

[17] Nikolaus Erbach-Fürstenau and Jan Ganschow. Towards responsible military use of artificial intelligence through consistent strategy, national governance, international norms and ethical principles. In *STO-MP-IST-210: Proceedings of the NATO Science and Technology Organization Information Systems Technology Panel*. NATO Science and Technology Organization (STO), 2026. Paper 4.01; Open Access.

[18] Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. Comparing and aggregating rankings with ties. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 47–58, Paris France, June 2004. ACM.

[19] Tristan Fauvel and Matthew Chalk. Efficient exploration in binary and preferential Bayesian optimization. *arXiv preprint arXiv:2110.09361*, 2021.

[20] Joseph Gatto, Parker Seegmiller, Timothy Burdick, Philip Resnik, Roshnik Rahat, Sarah DeLozier, and Sarah M. Preum. Medical triage as pairwise ranking: A benchmark for urgency in patient portal messages, 2026.

[21] Itzhak Gilboa, Stefania Minardi, and Fan Wang. Consumption of values. Hec paris research paper no. eco/scd-2020-1406, HEC Paris, November 2020.

[22] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D. Lawrence. Preferential Bayesian optimization. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1282–1291. PMLR, 06–11 Aug 2017.

[23] Salvatore Greco, Vincent Mousseau, and Roman Słowiński. Ordinal regression revisited: Multiple criteria ranking using a set of additive value functions. *European Journal of Operational Research*, 191(2):416–436, 2008.

[24] Ricardo Luna Gutierrez, Sahand Ghorbanpour, Vineet Gundecha, Rahman Ejaz, Varchas Gopalaswamy, Riccardo Betti, Avisek Naug, Desik Rengarajan, Ashwin Ramesh Babu, Paolo Faraboschi, et al. Explainable meta Bayesian optimization with human feedback for scientific applications like fusion energy. In *NeurIPS 2024 Workshop on Tackling Climate Change with Machine Learning*, 2024.

[25] Eyke Hüllermeier and Roman Słowiński. Preference learning and multiple criteria decision aiding: Differences, commonalities, and synergies–part i. *4OR*, 22(2):179–209, 2024.

[26] Bart Husslage, Peter Borm, Twan Burg, Herbert Hamers, and Roy Lindelauf. Ranking terrorists in networks: A sensitivity analysis of Al Qaeda's 9/11 attack. *Social Networks*, 42:1–7, July 2015.

[27] Maciej Jaworski, Piotr Duda, and Leszek Rutkowski. New splitting criteria for decision trees in stationary data streams. *IEEE transactions on neural networks and learning systems*, 29(6):2516–2529, 2017.

[28] Toshihiro Kamishima. Nantonac collaborative filtering: Recommendation based on order responses. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 583–588, New York, NY, USA, 2003. Association for Computing Machinery.

[29] J. Matias Kivikangas, Eeva Vilkkumaa, Julian Blank, Ville Harjunen, Pekka Malo, Kalyanmoy Deb, Niklas J. Ravaja, and Jyrki Wallenius. Effects of many conflicting objectives on decision-makers' cognitive burden and decision consistency. *European Journal of Operational Research*, 322(1):182–197, 2025.

[30] O. I. Larichev. Cognitive validity in design of decision-aiding techniques. *Journal of Multi-Criteria Decision Analysis*, 1(3):127–138, 1992.

[31] Xuheng Li, Heyang Zhao, and Quanquan Gu. Feel-good Thompson sampling for contextual dueling bandits. *arXiv preprint arXiv:2404.06013*, 2024.

[32] Wei-Ann Lin, Chih-Li Sung, and Ray-Bing Chen. Category tree Gaussian process for computer experiments with many-category qualitative factors and application to cooling system design. *Journal of Quality Technology*, 56(5):391–408, 2024.

[33] Kuang-Hsun Liu and Yu-Shan Shih. Score-scale decision tree for paired comparison data. *Statistica Sinica*, 26(1):429–444, 2016.

[34] Olena Mandrik, Alesya Yaumenenka, Rolando Herrero, and Marcel F. Jonker. Population preferences for breast cancer screening policies: Discrete choice experiment in Belarus. *PLOS ONE*, 14(11):1–17, 11 2019.

[35] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[36] Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[37] Quoc Phong Nguyen, Sebastian Tay, Bryan Kian Hsiang Low, and Patrick Jaillet. Top-k ranking Bayesian optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9135–9143, May 2021.

[38] Jens Brehm Bagger Nielsen, Jakob Nielsen, and Jan Larsen. Perception-based personalization of hearing aids using Gaussian processes and active learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):162–173, 2015.

[39] Giuseppe Nuti, Lluís Antoni Jiménez Rugama, and Andreea-Ingrid Cross. An explainable Bayesian decision tree algorithm. *Frontiers in Applied Mathematics and Statistics*, 7:598833, 2021.

[40] Barna Pásztor, Parnian Kassraie, and Andreas Krause. Bandits with preference feedback: A stackelberg game perspective. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 11997–12034. Curran Associates, Inc., 2024.

[41] Tony A. Plate. Accuracy versus interpretability in flexible modeling: Implementing a tradeoff using Gaussian process models. *Behaviormetrika*, 26(1):29–50, 1999.

[42] Nunung Nurul Qomariyah, Eileen Heriyanni, Ahmad Nurul Fajar, and Dimitar Kazakov. Comparative analysis of decision tree algorithm for learning ordinal data expressed as pairwise comparisons. In *2020 8th International Conference on Information and Communication Technology (ICoICT)*, pages 1–4, Yogyakarta, Indonesia, June 2020. IEEE.

[43] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[44] Carla Rebelo and Carlos Soares. Empirical evaluation of ranking trees on some metalearning problems. In *Proceedings 4th AAAI Multidisciplinary Workshop on Advances in Preference Handling*, 2008.

[45] Julian Rodemann, Federico Croppi, Philipp Arens, Yusuf Sale, Julia Herbinger, Bernd Bischl, Eyke Hüllermeier, Thomas Augustin, Conor J. Walsh, and Giuseppe Casalicchio. Explaining Bayesian optimization by shapley values facilitates human-AI collaboration for exosuit personalization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 525–542. Springer, 2025.

[46] Binxin Ru, Ahsan Alvi, Vu Nguyen, Michael A. Osborne, and Stephen Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8276–8285. PMLR, 13–18 Jul 2020.

[47] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[48] Seyed Mahdi Shavarani, Mahmoud Golabi, and Lhassane Idoumghar. Integrating active learning for improved preference modeling in tree-based interactive evolutionary multi-objective algorithms. In *2025 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2025.

[49] Seyed Mahdi Shavarani, Manuel López-Ibáñez, Richard Allmendinger, and Joshua Knowles. An interactive decision tree-based evolutionary multi-objective algorithm. In *Evolutionary Multi-Criterion Optimization: 12th International Conference, EMO 2023, Leiden, The Netherlands, March 20–24, 2023, Proceedings*, page 620–634, Berlin, Heidelberg, 2023. Springer-Verlag.

[50] Eero Siivola, Akash Kumar Dhaka, Michael Riis Andersen, Javier González, Pablo García Moreno, and Aki Vehtari. Preferential batch Bayesian optimization. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2021.

[51] Yanan Sui, Vincent Zhuang, Joel W Burdick, and Yisong Yue. Multi-dueling bandits with dependent arms. *arXiv preprint arXiv:1705.00253*, 2017.

[52] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved October 17, 2025, from `http://www.sfu.ca/~ssurjano`, 2013.

[53] Shion Takeno, Masahiro Nomura, and Masayuki Karasuyama. Towards practical preferential Bayesian optimization with skew Gaussian processes. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 33516–33533. PMLR, 23–29 Jul 2023.

[54] L. L. Thurstone. A law of comparative judgment. *Psychological Review*, 34(4):273–286, 1927.

[55] Charlie Vanaret, Jean-Baptiste Gotteland, Nicolas Durand, and Jean-Marc Alliot. Certified global minima for a benchmark of difficult optimization problems. *arXiv preprint arXiv:2003.09867*, 2020.

[56] Arun Verma, Zhongxiang Dai, Xiaoqiang Lin, Patrick Jaillet, and Bryan Kian Hsiang Low. Neural dueling bandits: Preference-based optimization with human feedback. *arXiv preprint arXiv:2407.17112*, 2024.

[57] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. In D. Touretzky, M.C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995.

[58] Wenjie Xu, Wenbin Wang, Yuning Jiang, Bratislav Svetozarevic, and Colin N. Jones. Principled preferential bayesian optimization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[59] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012. JCSS Special Issue: Cloud Computing 2011.

[60] Yisong Yue and Thorsten Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1201–1208, New York, NY, USA, 2009. Association for Computing Machinery.

[61] Luisa M. Zintgraf, Diederik M. Roijers, Sjoerd Linders, Catholijn M. Jonker, and Ann Nowé. Ordered preference elicitation strategies for supporting multi-objective decision making. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 1477–1485, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

A Related Work

Preference learning is a machine learning paradigm that learns a model from preference data. Often data comes in the form of pairwise comparisons, as these preference statements are cognitively easy to provide for humans. There are two main approaches to model preferences [25]. First, one can model preferences via a latent *utility function* $f : \mathcal{X} \to \mathcal{U}$, that assigns a utility score $u = f(x)$ to each object x, where a higher utility for x than y $f(x) > f(y)$ means x is preferred over y ($x \succ y$). Alternatively, one can learn preferences by learning *binary preference relations*. Instead of learning a global preference model, one learns a model that takes as input two objects x, y and predicts which one is preferred $x \succ y$.

Additionally, one can distinguish preference learning methods by their goal. One either wants to learn a *complete preference model* or simply find the *most preferred object*. The first setting is most common in applications like recommender systems or learning-to-rank, where accurate predictions across the entire input space are required. In contrast, in many decision-making and optimization scenarios, one usually only wants to efficiently identify the most preferred item(s).

Closely related to this distinction is the way in which preference data is collected. In *offline preference learning*, the learner is given a fixed dataset of preference observations and aims to infer a model from this data. This setting commonly occurs in recommender systems and learning-to-rank. In contrast, *active preference learning* allows the learner to adaptively select which queries to present, for example by choosing pairs of items (x, y) to compare. The goal is to maximize information gain or quickly identify the best item while minimizing the number of required queries. Active approaches are particularly important in settings where preference data is expensive to obtain, such as human-in-the-loop systems.

Although our proposed method could also be used to learn a complete preference model by using a different acquisition function, the main focus of our paper is on finding the most preferred solution using active preference learning. There are two main approaches to solve this: Dueling Bandits and Preferential Bayesian Optimization. Although dueling bandits is often used for online learning (where each query yields a reward which you want to maximize) instead of active learning, it can still be applied to active learning using a strategy that focuses solely on exploration. Therefore, we will give a short overview of the literature on both Dueling Bandits and Preferential Bayesian Optimization. Both use models like Bradley-Terry [10] or Thurstone [54] to find the pairwise likelihood $P(x \succ y)$ from pairwise comparisons but they use this likelihood in different ways.

Dueling Bandits: For a comprehensive review on dueling bandits see Bengs et al. [8]. Classical dueling bandits [59] model preferences through stochastic pairwise comparisons between a finite set of arms $P(x \succ y)$. Each object is treated as an arm, and feedback is obtained in the form of noisy outcomes indicating which of two arms is preferred. However, since they need to model each object as an arm, this model is only applicable to a pre-determined fixed set of objects. An extension to an infinite amount of arms is possible via utility-based dueling bandits [60] and can also exploit an assumed structure in $\mathcal{X}$ (e.g. linear [2]). These models assume a latent utility function exists and use it to find the pairwise probabilities. However, they only maintain pairwise probabilities. They do not model the utility function explicitly and there is no way to interpret why a certain best-solution found is actually the best solution. On the other hand, Sui et al. [51] and Chowdhury & Gopalan [14] explicitly model the preference function via a Gaussian process and learn it using dueling bandits. Although a slightly different problem setting, one where external context (e.g. information about the user) is present, contextual dueling bandits also (usually) explicitly model a latent utility function. Most commonly linear [31, 9] functions have been used but because linearity is too restrictive of an assumption, Neural Networks [56] have also been proposed. In summary, dueling bandits mostly represent preferences via stochastic binary preference relations. This approach only yields the best solution, it does not give any explanation as to why that solution is the best. Whilst utility-based extensions exist, they usually do not explicitly model the latent utility function and the ones that do use simplistic linear models or black-box models like Gaussian Processes and Neural Networks.

Preferential Bayesian Optimization PBO models preferences by constructing a probabilistic surrogate model of the latent utility function via Bayesian optimization. An acquisition function can then use this probabilistic estimate to efficiently sample the next pairwise comparison. First proposed by Chu & Ghahramani [15], Gaussian Processes have become the standard surrogate model. By

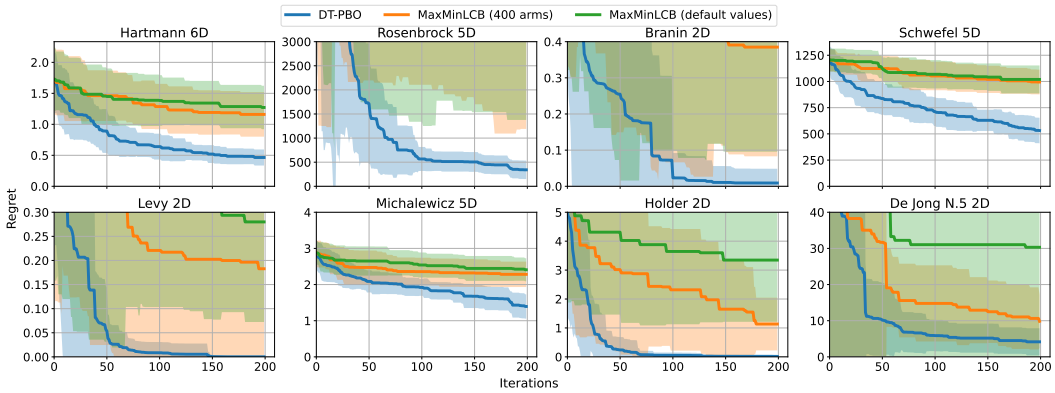

Figure 5: Convergence plots comparing DT-PBO to MaxMinLCB. Results are averaged over 20 runs and error bars show one standard deviation.

using a kernel function that measures the similarity between two points, GPs can efficiently create a model using very few pairwise comparisons. Recent research in PBO has focused on new acquisition functions (e.g., [22, 19, 3]) and new methods to infer the intractable posterior distribution to train GPs on PC data (e.g., [38, 50, 7, 53]). However, alternative surrogate models for PBO have rarely been explored. This is important because GPs suffer from several drawbacks. Most importantly, like dueling bandits, they are not interpretable. They rank each object and have an associated uncertainty score but it is impossible to see why one item is the best. Post-hoc explanation methods have been developed recently (e.g. [11, 24, 45]) but, to the best of our knowledge, an inherently interpretable surrogate model has not yet been used in PBO. Second, GPs struggle with handling qualitative data, especially when a qualitative factor contains many categories [32]. Discrete data namely introduces discontinuities in the utility function, as the function values will now only change at discrete points. The most commonly used Radial Basis Function kernel and Matérn(5/2) kernels encode inherent smoothness assumptions (infinitely differentiable and twice differentiable, respectively) [43] and are therefore incapable of modeling these discontinuities. There exist specialized kernels capable of handling both continuous and categorical data like CoCaBO [46] but they have not yet been applied to PBO. Note that even without discrete data, the utility function can exhibit discontinuities [21]. Finally, GPs are computationally complex, especially when using the full skewGP posterior. In summary, GPs have become the standard surrogate model in PBO. However, they have several drawbacks. They are black-box models, struggle with categorical data, and are computationally complex.

Comparing Dueling Bandits with DT-PBO Recently a novel Dueling Bandits paper, MaxMinLCB [40] has been proposed. Its method comes with theoretical cumulative regret bounds. However, as they focus on minimizing cumulative regret, they are focused on online learning instead of active learning which is our method's focus. For completeness, we present convergence results comparing MaxMinLCB to our method on the same eight benchmarks covered in the main paper. The default settings of MaxMinLCB recommend to use 100 arms for 2D problems and 200 arms for higher dimensional problems. Additionally, we show results with MaxMinLCB with 400 arms. The convergence plots in Figure 5 show that both versions of MaxMinLCB are outperformed by DT-PBO.

B Pseudocode

The pseudocode for DT-PBO can be found in Algorithm 1.

C Consistency score and the discarding of straddlers

To ensure we get shallow interpretable trees, we discard straddlers. This works controls the growth of the decision tree by limiting the amount of data that is propagated to the descendant splits. Whilst we do include hyperparameters that can also control the growth of the tree, they are not used by default as the straddler discarding handles the growth as well. The growth-controlling hyperparameters are as follows. First, the *minimum split score*: a node will only split if the best possible score $S_c(k^*, t^*)$

Algorithm 1: Decision-Tree Preferential Bayesian Optimization (DT-PBO)

Input: search space $\mathcal{X}$, preference function u, initial comparisons N_0, iterations T, hyperparameter ratio $\sigma^2_{noise}/\sigma^2_{prior}$

Generate N_0 initial pairs in $\mathcal{X}$ by Latin hypercube sampling and store the observed winner–loser pairs in dataset $\mathcal{D}$;

for $t = 1$ **to** T **do**

Fit a regression tree to $\mathcal{D}$ by recursively selecting the split $s^\star_A = (k^\star, t^\star)$ at each node A that maximizes the empirical consistency score

$$\widehat{S}_A(s) = \left| \frac{1}{N_A} \sum_{i=1}^{N_A} Z_i(s) \right|,$$

stopping when growth criteria are met; exclude straddling pairs from child datasets D_{A_L} and D_{A_R} for descendant splits;

Retain all pairs to estimate leaf parameters. Let $\mathcal{X}_1, \ldots, \mathcal{X}_m$ denote the regions of the m_l resulting leaves. Compute the MAP estimate $\mathbf{f}_{MAP}$ and the posterior covariance matrix

$$\Sigma^{-1}_{post} = \frac{1}{\sigma^2_{prior}} I + \Lambda_{MAP}$$

Condition the posterior on the sum-to-zero constraint $\mathbf{1}^\top \mathbf{f} = 0$ to resolve translational invariance, yielding the conditional Gaussian $\mathcal{N}(\boldsymbol{\mu}_c, \Sigma_c)$

$$\boldsymbol{\mu}_c = \mathbf{f}_{MAP} - \frac{\mathbf{1}^\top \mathbf{f}_{MAP}}{\mathbf{1}^\top \Sigma_{post}\, \mathbf{1}} \Sigma_{post}\, \mathbf{1}, \quad \Sigma_c = \Sigma_{post} - \frac{1}{\mathbf{1}^\top \Sigma_{post}\, \mathbf{1}} \big(\Sigma_{post}\, \mathbf{1}\big)\big(\Sigma_{post}\, \mathbf{1}\big)^\top$$

Because the surrogate is piecewise constant on the partition $\{\mathcal{X}_\ell\}_{\ell=1}^{m_l}$, qEUBO is constant on each leaf pair and depends only on the predictive means and variances of leaves ℓ, ℓ' obtained from $(\boldsymbol{\mu}_c, \Sigma_c)$. Select the leaf pair that maximizes qEUBO,

$$(\ell^\star, \ell'^\star) = \arg \max_{\ell, \ell' \in \{1, \ldots, m\}} \text{qEUBO}(\ell, \ell'),$$

then sample one query point uniformly at random from each selected leaf,

$$\mathbf{x}_{t+1} \sim \text{Unif}(\mathcal{X}_{\ell^\star}), \qquad \mathbf{x}'_{t+1} \sim \text{Unif}(\mathcal{X}_{\ell'^\star})$$

Query the preference oracle u on $(\mathbf{x}_{t+1}, \mathbf{x}'_{t+1})$ and append the resulting winner–loser pair to $\mathcal{D}$;

Output: final comparison dataset $\mathcal{D}$ and fitted decision tree surrogate model

is greater than this hyperparameter. If not, the node will become a leaf. This prevents the tree from making splits that are not informative enough or are likely based on noise. Secondly, a node is only considered for splitting if the number of comparison pairs it contains, $|D|$, is at least the value of the *minimum samples before split* hyperparameter. This prevents the model from overfitting on very small subsets of data. Finally, the *max depth* hyperparameter imposes a hard limit on the maximum number of levels in the tree. Any node reaching this depth is automatically converted into a leaf, regardless of any other criteria. Below we add theoretical arguments for our splitting heuristic and discarding straddlers. Additionally, we provide Empirical support for discarding straddlers. Finally, we discuss the option for prioritizing within-leaf comparisons (pairwise comparisons where both items fall into the same leaf).

C.1 Theoretical guarantee

Proposition 1 (repetition)

Assume a fixed node A and a finite candidate family $\mathcal{T}_A$ specified independently of the scoring sample, with N_A independent comparisons from the node's conditional distribution. The scope of the bound is discussed in Appendix C.1.

For any tolerance $\varepsilon > 0$, the probability of the empirical score deviating from the true score for a specific split s is bounded by:

$$\Pr\Big(\Big|\widehat{S}_A(s) - S_A(s)\Big| \geq \varepsilon\Big) \leq 2\exp\Big(-\frac{N_A\varepsilon^2}{2}\Big).$$

Consequently, by a union bound over the finite set of candidate splits $\mathcal{T}_A$:

$$\Pr\Big(\sup_{s\in\mathcal{T}_A}\Big|\widehat{S}_A(s) - S_A(s)\Big| \geq \varepsilon\Big) \leq 2|\mathcal{T}_A|\exp\Big(-\frac{N_A\varepsilon^2}{2}\Big).$$

Proof of Proposition 1

For a fixed split s, define

$$\hat{\mu}_A(s) = \frac{1}{N_A}\sum_{i=1}^{N_A} Z_i(s), \qquad \mu_A(s) = \mathbb{E}[Z(s) \mid X \in A,\ X' \in A].$$

Then $\widehat{S}_A(s) = |\hat{\mu}_A(s)|$ and $S_A(s) = |\mu_A(s)|$.

We first use the reverse triangle inequality:

$$\big||u| - |v|\big| \leq |u - v| \qquad \text{for all } u, v \in \mathbb{R}.$$

Applying this with $u = \hat{\mu}_A(s)$ and $v = \mu_A(s)$ gives

$$|\widehat{S}_A(s) - S_A(s)| = \big||\hat{\mu}_A(s)| - |\mu_A(s)|\big| \leq |\hat{\mu}_A(s) - \mu_A(s)|.$$

Therefore

$$\Pr\Big(|\widehat{S}_A(s) - S_A(s)| \geq \varepsilon\Big) \leq \Pr(|\hat{\mu}_A(s) - \mu_A(s)| \geq \varepsilon).$$

Since $Z_i(s) \in [-1, 1]$, Hoeffding's inequality yields

$$\Pr(|\hat{\mu}_A(s) - \mu_A(s)| \geq \varepsilon) \leq 2\exp\left(-\frac{2N_A^2\varepsilon^2}{\sum_{i=1}^{N_A}(1-(-1))^2}\right) = 2\exp\Big(-\frac{N_A\varepsilon^2}{2}\Big).$$

Combining the two displays proves the first claim.

For the second claim, define

$$E_s = \Big\{|\widehat{S}_A(s) - S_A(s)| \geq \varepsilon\Big\}.$$

Then

$$\Big\{\sup_{s\in\mathcal{T}_A}|\widehat{S}_A(s) - S_A(s)| \geq \varepsilon\Big\} = \bigcup_{s\in\mathcal{T}_A} E_s.$$

Applying the union bound gives

$$\Pr\Big(\sup_{s\in\mathcal{T}_A}|\widehat{S}_A(s) - S_A(s)| \geq \varepsilon\Big) \leq \sum_{s\in\mathcal{T}_A}\Pr(E_s) \leq 2|\mathcal{T}_A|\exp\Big(-\frac{N_A\varepsilon^2}{2}\Big).$$

The derivation draws loosely on ideas from [27].

Dimension dependence and scope of the bound

For $N_A > 0$ and a nonempty fixed candidate family, Proposition 1 gives, with probability at least $1 - \delta$ for $\delta \in (0, 1)$,

$$\sup_{s\in\mathcal{T}_A}|\widehat{S}_A(s) - S_A(s)| \leq \sqrt{\frac{2}{N_A}\log\frac{2|\mathcal{T}_A|}{\delta}}. \tag{11}$$

With at most M candidates per feature, $|\mathcal{T}_A| \leq dM$, so the radius grows as $\sqrt{\log d}$ when M, N_A, and δ are held fixed. This assumes a fixed node and candidate family independent of the scoring sample, with independent comparisons as in the proposition; it does not directly guarantee concentration for the algorithm's adaptive search.

This mild dimension dependence does not prevent optimization performance from degrading in higher dimensions. Strong feature interactions can require many more leaves for an axis-aligned, piecewise-constant tree to approximate the utility function. Correct split selection also requires a margin between the best and second-best population scores, which may shrink as additional features introduce competing splits. Finally, N_A decreases during recursion, since only comparisons with both items in a child are retained and straddlers are excluded. The bound therefore weakens where deeper refinement may be needed, and does not control these approximation or split-selection limitations.

Proposition 3 (Conditional correctness of removing straddlers)**.** *For a child node C, the recursive descendant split problem is defined on the conditional event that* both *items of the comparison lie in C. Hence, its natural population target is*

$$S_C(r) = |\mathbb{E}[Z(r) \mid X \in C,\ X' \in C]|\,.$$

Removing straddlers is the operation that makes the child-level empirical objective match this child-level conditional population objective.

Proof. By construction, the dataset D_C consists exactly of the observed comparisons satisfying the conditioning event where both items lie in C. Therefore, the empirical score computed from D_C is the empirical analogue of $S_C(r)$.

If one did not discard straddling pairs when optimizing descendant splits, then the corresponding empirical score would no longer estimate $S_C(r)$. Instead, it would estimate a different mixture quantity combining within-child and cross-child comparisons. Thus, removing straddlers is not an ad hoc data deletion step for recursive split selection; it is a necessary structural alignment. □

Corollary 1 (Exclusion of straddlers under ancestor separation)**.** *Let $x \in A_R$ and $x' \in A_L$. Under the defined tree split s_A and utility score $f(x)$, it holds that $f(x) - f(x') > 0$. Hence, every straddler is ordered solely by the parent split.*

Proof. Recall the following definitions:

- $f(x)$ is the utility score of an item.
- $s_A(x)$ represents the tree split itself. It assigns -1 if the item falls into the left child node (A_L) and $+1$ if it falls into the right child node (A_R). γ_A is the "bonus" or "penalty" an item gets just for landing on the right or left side of the split. The total gap created by this split between the two sides is exactly $2\gamma_A$.
- $h_A(x)$ is the residual or leftover utility. It represents the minor, local variations in an item's score that are not explained by the main split.

Using these definitions, we can expand the difference in utility scores:

$$f(x) - f(x') = \gamma_A s_A(x) + h_A(x) - \gamma_A s_A(x') - h_A(x') = 2\gamma_A + h_A(x) - h_A(x').$$

Since the residual utility difference is bounded by the total gap:

$$|h_A(x) - h_A(x')| \leq \sup_{u \in A_R,\, v \in A_L} |h_A(u) - h_A(v)| < 2\gamma_A,$$

we obtain $f(x) - f(x') > 0$. Therefore, such pairs provide information about the already determined parent-level ordering only, and not about the choice among descendant splits within A_L or A_R. □

C.2 Empirical support for discarding straddlers

As an experimental variant of the procedure described above, we also implemented a scheme in which straddlers are retained rather than discarded. For a parent split $s_A^\star = (k^\star, t^\star)$ with child regions A_L and A_R, each straddling pair is reassigned to the child whose region contains its winner. The augmented child datasets are

$$\begin{aligned} \tilde{D}_{A_L} &= D_{A_L} \cup \{(\mathbf{x}, \mathbf{x}') \in D_A : \mathbf{x} \in A_L,\ \mathbf{x}' \in A_R\}, \\ \tilde{D}_{A_R} &= D_{A_R} \cup \{(\mathbf{x}, \mathbf{x}') \in D_A : \mathbf{x} \in A_R,\ \mathbf{x}' \in A_L\}, \end{aligned} \tag{12}$$

Table 2: Effect of keeping straddlers on train and test score for BELA.

	Training acc.	Test acc.
Discarding straddlers	0.97 ± 0.06	0.63 ± 0.28
Keeping straddlers	1.0 ± 0.01	0.60 ± 0.32

so that every comparison previously dropped now contributes to the descendant containing the winner.

Without further restriction this scheme allows the same pair to keep driving splits on the parent feature: by construction the two items differ on dimension $k^\star$, so any descendant candidate split on $k^\star$ that lies between their two values produces another non-zero Z_i, and the recursion need not terminate. To rule this out, each pair is associated with a set of masked dimensions $\mathcal{K}_i^A \subseteq \{1, \ldots, d\}$ recording the features along which it has already been retained on the path from the root to A. Whenever a pair is retained across a split on dimension $k^\star$, $k^\star$ is added to $\mathcal{K}_i$ in the corresponding child. Which results in descendant splits not being able to use these dimensions (features) again. For a candidate split $s = (k, t)$ at node A, the indicator from Eq. 5 is replaced by

$$\tilde{Z}_i(s) = Z_i(s)\, \mathbf{1}\{k \notin \mathcal{K}_i^A\}, \tag{13}$$

and the empirical consistency score becomes

$$\widehat{\tilde{S}}_A(s) = \left| \frac{1}{N_A} \sum_{i=1}^{N_A} \tilde{Z}_i(s) \right|. \tag{14}$$

A retained pair therefore contributes signal only on features that have not yet been used to promote it, ensuring that no single comparison can drive more than one split on the same feature along any root-to-leaf path.

The tree is now built again for the BELA dataset while retaining straddlers, leading to the tree in Figure 7. The tree that is found while discarding the straddlers is shown in Figure 6. The difference in train and test score is shown in Table 2. Clearly discarding straddlers yields a much more interpretable tree whilst retaining nearly all of its performance.

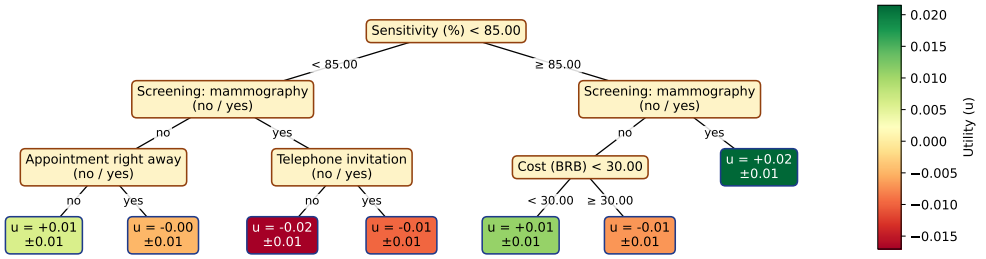

Figure 6: Trained BELA tree while discarding straddlers.

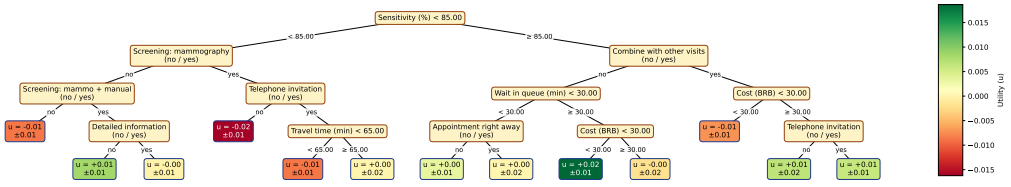

Figure 7: Trained BELA tree without discarding the straddlers and setting the max depth to 4 nodes.

C.3 Within-leaves prioritization

The heuristic described in Section 3.3 can be formalized by augmenting qEUBO with a structural-refinement term. Let

$$\alpha_{\mathrm{DT}}(\mathbf{x}, \mathbf{x}') = \mathrm{qEUBO}(\mathbf{x}, \mathbf{x}') + \lambda \cdot I_{\mathrm{struct}}(\mathbf{x}, \mathbf{x}'),$$

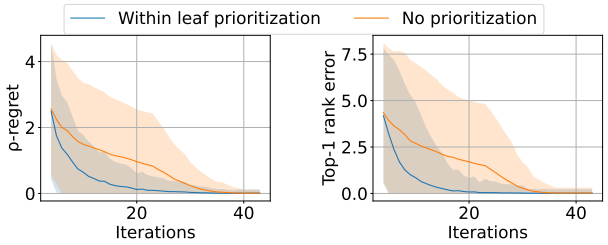

Figure 8: The effect of with- and without within-leaf prioritization for PBO on the sushi dataset.

where $\lambda \geq 0$ controls how strongly the acquisition prioritizes structural refinement relative to utility-based exploration and I_{struct} represents the expected change in the consistency score within the leaf containing the pair. For a candidate pair $(\mathbf{x}, \mathbf{x}')$ falling in leaves A_i and A_j, we define:

$$I_{\text{struct}}(\mathbf{x}, \mathbf{x}') = \mathbb{E}_Y \left[\left| \widehat{S}_A^{(+Y)}(s_A^\star) - \widehat{S}_A(s_A^\star) \right| \right] \cdot \mathbf{1}\{A_i = A_j = A\}$$

where $Y \in \{\mathbf{x} \succ \mathbf{x}', \mathbf{x}' \succ \mathbf{x}\}$ is the unobserved outcome, $\widehat{S}_A^{(+Y)}$ is the consistency score recomputed after appending the comparison with outcome Y to D_A, and the expectation is taken under the current Bernoulli predictive. Within a leaf, the predictive is symmetric ($P(\mathbf{x} \succ \mathbf{x}') = \Phi(0) = 0.5$), so the expectation reduces to the average over the two outcomes.

The indicator $\mathbf{1}\{A_i = A_j\}$ encodes the fact that between-leaf comparisons cannot affect either child's consistency score: if $A_i \neq A_j$, the pair is a straddler with respect to the parent split that separated them, is excluded from both D_{A_i} and D_{A_j} by Eq. 8, and therefore enters neither $\widehat{S}_{A_i}$ nor $\widehat{S}_{A_j}$. Hence, $I_{\text{struct}} = 0$ on between-leaf pairs, and α_{DT} gracefully reduces to standard qEUBO for any λ. The structural term is active only within a leaf, exactly where qEUBO collapses.

Crucially, the practical within-leaf-prioritization heuristic used in our experiments corresponds exactly to evaluating α_{DT} in the limit $\lambda \to \infty$, restricted strictly to the top leaf. Restricting to the top leaf is motivated by the most-preferred-solution objective. Structural refinement matters most where the optimum is plausibly located and it fundamentally avoids the dominant computational cost of evaluating I_{struct} in sub-optimal leaves. Figure 8 compares this rule against plain qEUBO ($\lambda = 0$) on the sushi dataset. Within leaf-prioritization shows much better performance than without prioritization.

Structural-weight ablation. We extend the heuristic above with $\lambda \in \{0, 0.1, 1, 10, \infty\}$ and a separate run of the within-leaf-prioritization heuristic. We use the first 20 users of SUSHI-A, pairing configurations by user, comparison shuffle, and initial design. Each user's ranking supplies 45 noiseless comparisons: four are selected for initialization by item coverage, and the remaining 41 are queried without replacement. The tree is refitted before every query, with maximum depth 15 and noise-to-prior variance ratio 0.1.

The implementation evaluates structural gain as the expected increase in the best *non-normalized* consistency score, recomputing the best split after each hypothetical outcome. This is an operational variant of the fixed-split normalized score change defined above. All remaining pairs are eligible in this sweep, including within-leaf pairs outside the highest-mean leaf. At $\lambda = \infty$, structural gain is maximized first and qEUBO breaks ties. Remaining ties follow the fixed candidate order. The separately evaluated heuristic first restricts candidates to within-leaf pairs whenever any remain, then maximizes qEUBO, selecting the highest-mean leaf with an available pair. It falls back to between-leaf comparisons when no within-leaf pair remains.

Table 3 reports each metric averaged over the 41 pre-query evaluations (comparison budgets 4–43), followed by averaging across users. The uncertainty intervals are approximate 95% confidence intervals, computed as 1.96 times the standard error across the 20 user-level averages. Figure 9 shows the corresponding learning curves.

Larger λ leads to more within-leaf queries and lower ρ-regret. From $\lambda = 10$ onward, the acquisition behaves like the prioritization heuristic: in this experiment, $\lambda = 10$, $\lambda = \infty$ and the heuristic give identical results, and the limiting rule and the heuristic chose the same pair at every one of the 800

Table 3: Structural-weight ablation on SUSHI-A. Lower ρ-regret and top-1 rank error are better. Values are each metric averaged over all comparison budgets (4–43) within a user's run, i.e. the mean height of the learning curve, then averaged across 20 paired users ($\pm$ 95% confidence intervals). The last column is the fraction of acquired comparisons whose items share a leaf.

Acquisition rule	ρ-regret	Top-1 rank error	Within-leaf picks
$\lambda = 0$	0.635 ± 0.124	0.830 ± 0.305	7.1%
$\lambda = 0.1$	0.577 ± 0.096	0.326 ± 0.145	16.2%
$\lambda = 1$	0.512 ± 0.084	0.300 ± 0.155	27.2%
$\lambda = 10$	0.343 ± 0.074	0.341 ± 0.140	44.6%
$\lambda = \infty$	0.343 ± 0.074	0.341 ± 0.140	44.6%
Prioritization heuristic	0.343 ± 0.074	0.341 ± 0.140	44.6%

checked steps. Top-1 error improves over $\lambda = 0$ but does not decrease steadily with λ, as it is lowest at $\lambda = 1$. Finally, even $\lambda = 0$ (plain qEUBO) picks some within-leaf pairs, because the pool of remaining candidates is finite and shrinks as pairs are queried.

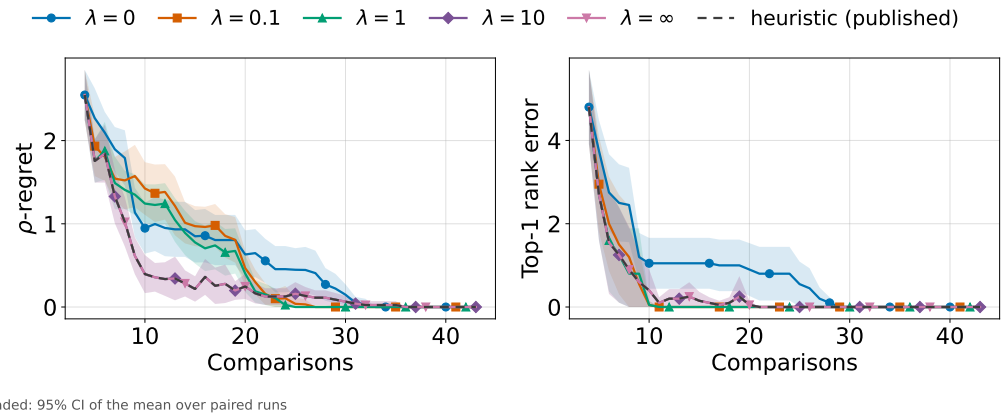

Figure 9: SUSHI-A learning curves for the structural-weight ablation. The horizontal axis counts observed comparisons, including the four initial comparisons. Curves show means across 20 paired users; shaded regions are approximate 95% confidence intervals.

Leaf eligibility and pair selection. We additionally try every combination of three eligible-leaf sets (top 1, top 3, and all leaves) and three within-leaf pair-selection rules (uniform, closest, and farthest). We use the same 20 users, initial comparisons, model settings, and 40-query budget as the structural-weight sweep. Leaves containing items are ranked by posterior mean before checking whether their internal comparison pools are exhausted. We then sample uniformly among eligible leaves with an available pair. This makes the broader eligible sets operative: maximizing qEUBO over within-leaf pairs would simply select the highest-mean eligible leaf again.

Within the selected leaf, the uniform rule samples an unqueried pair uniformly. The closest and farthest rules minimize and maximize, respectively, the mean absolute feature difference normalized by the feature ranges; the nominal minor-group feature contributes a binary mismatch instead of an ordinal distance. Ties are broken randomly, independently of the true ranking. If no eligible within-leaf pair remains, we select a between-leaf pair using qEUBO. If that pool is also exhausted, we use qEUBO on the remaining pairs to complete the budget; this final fallback occurs in 3.9–5.4% of queries for the restricted policies and never for the all-leaf policies. The configurations share one policy seed per user, and confidence intervals summarize variation across users.

As shown in Table 4, expanding the eligible set increases within-leaf querying from approximately 12% (top 1) to 26–28% (top 3) and 49–51% (all leaves). Top-1 restriction yields the lowest mean top-1 error for each pair-selection rule, whereas top-3 restriction yields the lowest mean ρ-regret. No pair-selection rule is uniformly best across both metrics and all eligible sets. Widening the eligible set

Table 4: Leaf-eligibility and pair-selection ablation on SUSHI-A. Metrics are averaged over comparison budgets 4–43, then across 20 paired users (mean ± approximate 95% confidence interval). Lower regret and top-1 error are better. Within-leaf picks include the final fallback when both restricted pools are exhausted.

Eligible leaves	Pair rule	ρ-regret	Top-1 error	Within-leaf picks
Top 1	Uniform	0.501 ± 0.122	0.460 ± 0.203	11.6%
Top 1	Closest	0.581 ± 0.107	0.367 ± 0.161	11.6%
Top 1	Farthest	0.512 ± 0.091	0.426 ± 0.201	11.6%
Top 3	Uniform	0.451 ± 0.091	0.696 ± 0.185	26.7%
Top 3	Closest	0.460 ± 0.111	0.626 ± 0.199	27.9%
Top 3	Farthest	0.463 ± 0.092	0.745 ± 0.242	26.3%
All	Uniform	0.501 ± 0.089	0.804 ± 0.161	48.9%
All	Closest	0.544 ± 0.106	0.731 ± 0.186	48.5%
All	Farthest	0.513 ± 0.090	1.030 ± 0.301	50.6%

consistently increases top-1 error, while ρ-regret is lowest for the intermediate top-3 set and returns to roughly the top-1 level when all leaves are eligible. Concentrating within-leaf queries on the currently preferred leaf thus favors identifying the best item, whereas spreading them over a few high-ranked leaves helps resolve the broader ranking. Spreading them over all leaves gives no further ranking benefit. The ρ-regret differences are small relative to the confidence intervals, so these results indicate a trade-off rather than a universally optimal policy.

D Why only the ratio $\sigma^2_{noise}/\sigma^2_{prior}$ matters

In this section we show that both the Laplace approximation and the EUBO acquisition function depend only on the ratio $\sigma^2_{noise}/\sigma^2_{prior}$, and not on the individual values. Since this is the case, we only need to specify the ratio as a hyperparameter.

D.1 Laplace approximation

When doing Laplace approximation, we seek the minimizer of the following objective function:

$$L(f) = -\sum_{i=1}^{n} \ln \Phi\left(\frac{f(x_i) - f(x_i')}{\sqrt{2}\,\sigma_{noise}}\right) + \frac{1}{2\sigma^2_{prior}} f^\top f.$$

Introduce the rescaled variable

$$\tilde{f} = \frac{f}{\sigma_{noise}}.$$

Substituting $f = \sigma_{noise}\tilde{f}$ into $L(f)$ yields

$$\begin{aligned} L(\tilde{f}) &= -\sum_{i=1}^{n} \ln \Phi\left(\frac{\sigma_{noise}(\tilde{f}_i - \tilde{f}_i')}{\sqrt{2}\,\sigma_{noise}}\right) + \frac{1}{2\sigma^2_{prior}} (\sigma_{noise}\tilde{f})^\top (\sigma_{noise}\tilde{f}) \\ &= -\sum_{i=1}^{n} \ln \Phi\left(\frac{\tilde{f}_i - \tilde{f}_i'}{\sqrt{2}}\right) + \frac{\sigma^2_{noise}}{2\sigma^2_{prior}} \tilde{f}^\top \tilde{f}. \end{aligned}$$

We observe that the likelihood term no longer depends on σ_{noise}, while the prior term depends only on the ratio

$$\lambda = \frac{\sigma^2_{noise}}{\sigma^2_{prior}}.$$

Therefore, the objective can be rewritten as

$$L(\tilde{f}) = -\sum_{i=1}^{n} \ln \Phi\left(\frac{\tilde{f}_i - \tilde{f}_i'}{\sqrt{2}}\right) + \frac{\lambda}{2} \tilde{f}^\top \tilde{f},$$

which shows that the optimization problem in the rescaled variable $\tilde{f}$ depends only on λ. Consequently, only the ratio $\sigma^2_{noise}/\sigma^2_{prior}$ needs to be specified.

D.2 Acquisition function

Suppose the posterior over latent utilities is approximated by

$$f \mid \mathcal{D} \approx \mathcal{N}(\mu, \Sigma),$$

where μ is the posterior mean and Σ the posterior covariance. For a candidate pair (i, j), define

$$\mu_{ij} = \mu_i - \mu_j, \qquad \sigma^2_{ij} = \Sigma_{ii} + \Sigma_{jj} - 2\Sigma_{ij}.$$

The EUBO acquisition score is

$$\alpha(i,j) = \mu_{ij}\,\Phi\left(\frac{\mu_{ij}}{\sigma_{ij}}\right) + \sigma_{ij}\,\phi\left(\frac{\mu_{ij}}{\sigma_{ij}}\right) + m_j.$$

Now consider two parameterizations with the same ratio

$$\frac{\sigma^2_{\text{noise}}}{\sigma^2_{\text{prior}}} = \frac{\hat{\sigma}^2_{\text{noise}}}{\hat{\sigma}^2_{\text{prior}}}.$$

From the reparameterization above, keeping this ratio fixed leaves the inference problem unchanged in the rescaled variable $\tilde{f} = f/\sigma_{\text{noise}}$. Therefore, changing $(\sigma_{\text{noise}}, \sigma_{\text{prior}})$ while preserving this ratio only induces a global rescaling of the latent function, i.e., there exists a constant $c > 0$ such that

$$\hat{f} = cf.$$

Under this rescaling, the Laplace posterior approximation transforms accordingly:

$$\hat{\mu} = c\,\mu, \qquad \hat{\Sigma} = c^2\Sigma.$$

Hence

$$\hat{\mu}_{ij} = c\,\mu_{ij}, \qquad \hat{\sigma}_{ij} = c\,\sigma_{ij},$$

and therefore the standardized quantity is invariant:

$$\frac{\hat{\mu}_{ij}}{\hat{\sigma}_{ij}} = \frac{c\mu_{ij}}{c\sigma_{ij}} = \frac{\mu_{ij}}{\sigma_{ij}}.$$

Substituting into the acquisition function gives

$$\begin{aligned}
\hat{\alpha}(i,j) &= \hat{\mu}_{ij}\,\Phi\left(\frac{\hat{\mu}_{ij}}{\hat{\sigma}_{ij}}\right) + \hat{\sigma}_{ij}\,\phi\left(\frac{\hat{\mu}_{ij}}{\hat{\sigma}_{ij}}\right) + \hat{\mu}_j \\
&= c\,\mu_{ij}\,\Phi\left(\frac{\mu_{ij}}{\sigma_{ij}}\right) + c\,\sigma_{ij}\,\phi\left(\frac{\mu_{ij}}{\sigma_{ij}}\right) + c\,\mu_j \\
&= c\,\alpha(i,j).
\end{aligned}$$

Thus, all EUBO scores are multiplied by the same positive constant c, and so their ordering is unchanged:

$$\arg\max_{(i,j)} \hat{\alpha}(i,j) = \arg\max_{(i,j)} \alpha(i,j).$$

Therefore, although the absolute values of the EUBO scores depend on the overall scale of the latent utilities, the selected comparison pair depends only on the ratio $\sigma^2_{\text{noise}}/\sigma^2_{\text{prior}}$, not on σ_{noise} and σ_{prior} separately.

E Removing Translational Degeneracy

The pairwise likelihood defined in Eq. 2 is *translationally invariant* with respect to $\mathbf{f}$. That is, adding a constant shift c to all components of $\mathbf{f}$ does not influence the pairwise differences and thus leaves the likelihood unchanged. Intuitively, this invariance is expected: because the latent function f is never observed directly, its absolute offset is unidentifiable. However, this invariance has a direct consequence for the Hessian of $L(\mathbf{f})$ and therefore for the posterior covariance defined in Eq. 10. In particular, the likelihood Hessian term Λ_{MAP} is *degenerate* in the constant shift direction: $\Lambda_{\text{MAP}}\mathbf{1} = \mathbf{0}$, where $\mathbf{1}$ is the m_l-dimensional all-ones vector and $\mathbf{0}$ the m_l-dimensional all-zeroes vector.

As a result, the curvature of the negative log-posterior in the all-ones direction, $\mathbf{1}^\top H(L(\mathbf{f}))\mathbf{1}$, is governed entirely by the prior:

$$\mathbf{1}^\top H(L(\mathbf{f}))\mathbf{1} = \mathbf{1}^\top \left(\frac{1}{\sigma^2_{prior}} I + \Lambda_{\text{MAP}} \right) \mathbf{1} = \frac{m_l}{\sigma^2_{prior}}.$$

Consequently, the posterior covariance in this direction is:

$$\mathbf{1}^\top \Sigma_{\text{post}} \mathbf{1} = \mathbf{1}^\top H(L(\mathbf{f}))^{-1} \mathbf{1} = m_l \sigma^2_{prior}. \tag{15}$$

which follows from solving the linear system $H(L(\mathbf{f}))\mathbf{v} = \mathbf{1}$, which implies $\mathbf{v} = H(L(\mathbf{f}))^{-1}\mathbf{1}$. Since $H(L(\mathbf{f}))\mathbf{1} = \frac{1}{\sigma^2_{prior}}\mathbf{1}$, it follows that $\mathbf{v} = \sigma^2_{prior}\mathbf{1}$.

This invariance causes a large irreducible variance term $m_l \sigma^2_{prior}$, especially when σ_{prior} is large.

To remove the irreducible variance, the model is made identifiable by adding a sum-to-zero constraint: $\sum_{i=1}^{m_l} f = \mathbf{1}^\top \mathbf{f} = 0$. Essentially, this fixes the scale of the model, preventing a shift in the all-ones-direction, and thereby removing any variance coming from this direction. This constraint is incorporated by conditioning the posterior on $\mathbf{1}^\top \mathbf{f} = 0$. The posterior then follows a conditional distribution $p(\mathbf{f} | \mathbf{1}^\top \mathbf{f} = 0, \mathcal{D})$. Note that since $p(\mathbf{f}|\mathcal{D})$ follows a normal distribution after Laplace approximation, $p(\mathbf{1}^\top \mathbf{f})$ also follows a normal distribution (as a sum of normal distributions), and thus by using Eaton's [16] normal conditional distribution result $p(\mathbf{f} | \mathbf{1}^\top \mathbf{f} = 0, \mathcal{D})$ follows a conditional normal distribution (for the full derivation see below).

Derivation of the Conditional Distribution We start from Eaton's general result (Eaton, 1983, pp. 116–117) for the conditional distribution of a joint normal vector

$$\begin{pmatrix} X \\ S \end{pmatrix} \sim \mathcal{N}\Big(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_S \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \Big),$$

where $X \in \mathbb{R}^n$, $S \in \mathbb{R}$,

then $X \mid (S = s)$ follows a multivariate normal

$$X \mid (S = s) \;\sim\; \mathcal{N}\Big(\boldsymbol{\mu} + \Sigma_{12}\, \Sigma_{22}^{-}\, (s - \mu_S) \;,\; \Sigma_{11} - \Sigma_{12}\, \Sigma_{22}^{-}\, \Sigma_{21} \Big).$$

Specializing to the Sum Constraint

Let

$$S = \mathbf{1}^\top X = \sum_{i=1}^{n} X_i, \quad \mu_S = \mathbf{1}^\top \mu = \sum_{i=1}^{n} \mu_i., \quad s = 0$$

Note that S is normally distributed as it is the sum of normally distributed random variables. Also note that $(X_1, ..., X_N, S)$ is jointly normal as an affine transform of a jointly normal distribution. Therefore, we can use Eaton's result with the following covariance blocks:

(i) Σ_{12}**.**

$$\Sigma_{12} = \operatorname{Cov}\big(X,\ \mathbf{1}^\top X\big) = \operatorname{Cov}\big(X,\ \sum_{i=1}^{n} \mathbf{1}_i X_i\big) \overset{\text{bilinearity}}{=} \sum_{i=1}^{n} \mathbf{1}_i \operatorname{Cov}\big(X,\ X_i\big) = \sum_{i=1}^{n} \mathbf{1}_i \Sigma_{:,i} = \Sigma\, \mathbf{1}.$$

(ii) Σ_{21}**.**

$$\Sigma_{21} = \mathrm{Cov}\big(\mathbf{1}^\top X,\ X\big) = \mathbf{1}^\top\ \mathrm{Cov}(X, X) = \mathbf{1}^\top\,\Sigma_{\cdot} = (\Sigma\mathbf{1})^\top$$

(iii) Σ_{22}**.**

$$\Sigma_{22} = \mathrm{Var}\big(\mathbf{1}^\top X\big) = \mathbf{1}^\top\,\Sigma\,\mathbf{1},$$

which is a scalar. Its inverse is thus simply $\Sigma_{22}^{-} = \frac{1}{\mathbf{1}^\top\Sigma\,\mathbf{1}}$.

Plugging into the Block-Formula

Using these identifications in the conditional-normal formula, gives

$$X \mid \sum_{i=1}^{n} X_i = 0\ \sim\ \mathcal{N}\Big(\mu\ -\ \frac{\mathbf{1}^\top\mu}{\mathbf{1}^\top\Sigma\,\mathbf{1}}\,\Sigma\,\mathbf{1}\ ,\ \Sigma\ -\ \frac{(\boldsymbol{\Sigma}\,\mathbf{1})\,(\boldsymbol{\Sigma}\,\mathbf{1})^\top}{\mathbf{1}^\top\Sigma\,\mathbf{1}}\Big)\,.$$

However, we can simplify the result. Under the Laplace approximation we have $\mu = \mathbf{f}_{\mathrm{MAP}}$.Because the likelihood is translationally invariant, its gradient has no component in the all-ones direction, i.e. $\mathbf{1}^\top\nabla\ell(\mathbf{f}) = 0$. The MAP condition for the negative log-posterior therefore gives

$$\frac{1}{\sigma^2_{\mathrm{prior}}}\mathbf{1}^\top\mathbf{f}_{\mathrm{MAP}} = 0,$$

and hence $\mathbf{1}^\top\mathbf{f}_{\mathrm{MAP}} = 0$. Consequently, the correction term in the conditional Gaussian mean,

$$\frac{\mathbf{1}^\top\mu}{\mathbf{1}^\top\Sigma\mathbf{1}}\Sigma\mathbf{1},$$

vanishes. Thus the sum-to-zero conditioning removes posterior variance in the constant-shift direction, but leaves the MAP mean unchanged.

Additionally, by using Eq. 15, we can simplify the conditional covariance to

$$\Sigma_c = \Sigma_{\mathrm{post}} - \frac{\sigma^2_{\mathrm{prior}}}{m_l}\mathbf{1}\mathbf{1}^\top.$$

Leading to the following conditional Gaussian:

$$\boxed{p(\mathbf{f} \mid \mathbf{1}^\top\mathbf{f} = 0, \mathcal{D}) \approx \mathcal{N}\left(\mathbf{f}_{\mathrm{MAP}}, \Sigma_{\mathrm{post}} - \frac{\sigma^2_{\mathrm{prior}}}{m_l}\mathbf{1}\mathbf{1}^\top\right)}$$

F Detailed Numerical Experiments

In this Section, we first give a detailed description of each model's settings. Next, we describe all of the benchmark functions, their ruggedness and each model's learned representation. Subsequently, we perform an ablation study of our model's hyperparameter. Next, we dive into the scalability of our model in terms of running time and optimization function dimension. We look into our model's capability to handle noisy preferences. Finally, we vary the maximum depth of the tree to validate the model's interpretability.

F.1 Model Settings

For SkewGP with HB-EI, the implementation of Takeno et al. [53] is used and for GP the implementation in Botorch [4] is used. To find the MAP-estimates of our decision tree scipy's trust-exact solver with a maximum of 100 iterations is used. For GP the lengthscales are selected by marginal maximum likelihood estimation after each iteration, whereas for SkewGP this is done every 10 iterations due to the high computational complexity. For POP-BO, the implementation of Xu et al. [58] is used with the defaults its code applies to functions without hand-tuned hyperparameters: an ARD Matérn(5/2) kernel with variance 1 and lengthscales equal to half the average domain width, norm bound $B = 6$,

$\beta_t = \sqrt{t+1}$, latent utilities bounded to $[-3, 3]$, and Ipopt to optimize its acquisition function. For DT-PBO, the $\sigma^2_{noise}/\sigma^2_{prior}$ ratio is set to $1/4$ and a hyperparameter sensitivity study shown in Appendix F.4 shows that the hyperparameter is insensitive. The qEUBO acquisition function is used to identify the best between-leaf comparisons. For the synthetic benchmark experiments, within-leaf comparisons are only used when there are too few between-leaf comparisons

F.2 Benchmark Optimization Functions and Learned Representations

To compare our model's performance to kernel-based alternatives (GP, SkewGP, POP-BO) eight benchmark functions are used. In Figure 10, surface plots of these eight functions are shown. For Rosenbrock, Michalewicz, and Schwefel, 2D variants of the functions have been plotted. For Hartmann, an arbitrarily chosen 2D slice ($z = 0.7$) of the 3D variant has been shown. The surface plots show the ruggedness of the 2D variants of the functions. Additionally, we sampled 256 random pairwise comparisons for each benchmark function and trained each model on the same set of comparisons. The learned representations are shown in the columns. It shows that the kernel-based methods learn a smooth representation of the function whereas the DT learns a discrete representation. All representations generally come close to the ground-truth

Ruggedness is quantified using the normalized directional second differences on the original N-D functions. After mapping each benchmark function's domain to the unit hypercube and standardizing function values, we estimate the average absolute second-order finite difference (a measure for curvature in a certain direction) over randomly sampled points and directions, aggregated across multiple step sizes. Larger values indicate stronger local curvature, oscillation, or sharp basin structure. The results can be found in Table 5.

Both the table and surface plots show that Hartmann, Rosenbrock, and Branin are not rugged, whereas De Jong, Holder, and Michalewicz exhibit highly rugged landscapes, with Levy and Schwefel displaying moderate ruggedness.

Further details of the optimization functions can be found in Surjanovic & Bingham [52]. Note that the original functions are minimization functions. However, since our model tries to find the maximum, the functions are transformed by multiplying them by -1.

Table 5: Ruggedness metrics for the benchmark functions. We report the mean and 95th percentile (p95) of the normalized directional second-difference measure.

Function	**Mean Ruggedness**	**p95 Ruggedness**
DeJong	1162.28	8611.07
Holder	703.40	2355.97
Michalewicz	488.89	1842.49
Levy	259.45	913.48
Schwefel	230.98	495.63
Branin	24.45	84.38
Rosenbrock	11.37	31.57
Hartmann	7.76	30.19

F.3 Varying Kernels

In Figure 11 and Figure 12, we present convergence plots for GP and SkewGP models using the RBF, Matérn(5/2), Matérn(3/2), and Matérn(1/2) kernels, ordered by decreasing smoothness assumptions. The results show that, regardless of the chosen kernel, DT consistently outperforms the kernel-based models on the Schwefel, Hölder, and De Jong functions. Furthermore, no single kernel performs best across all functions. These findings indicate that the conclusions in the main paper are robust to the choice of kernel.

Figure 13 shows the same comparison for POP-BO, whose default kernel is Matérn(5/2); all other POP-BO settings are unchanged. Because POP-BO optimizes its acquisition function with the gradient-based solver Ipopt, the scaled distance d in the Matérn(3/2) and Matérn(1/2) kernels is replaced by $\sqrt{d^2 + 10^{-16}}$, so that derivatives remain finite at coincident points. For POP-BO,

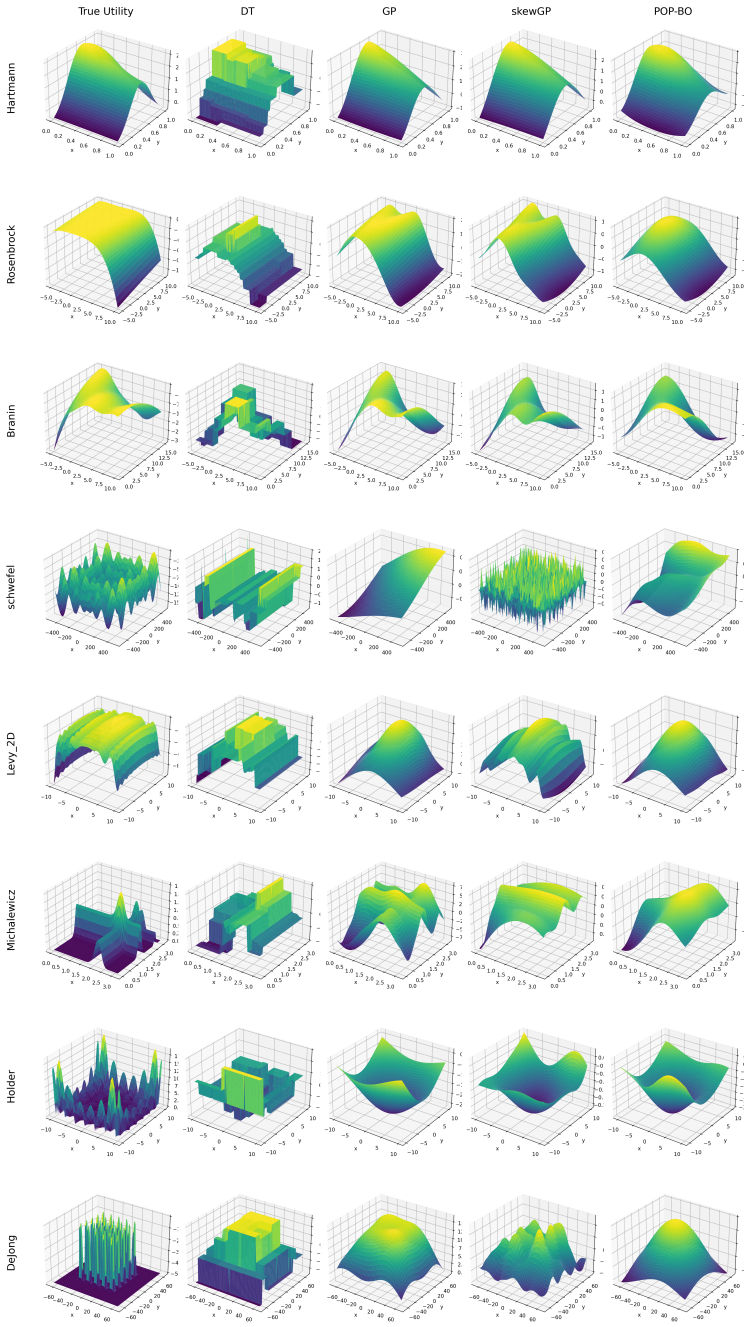

Figure 10: Surface plots and learned representations of each optimization function for all baselines and DT-PBO.

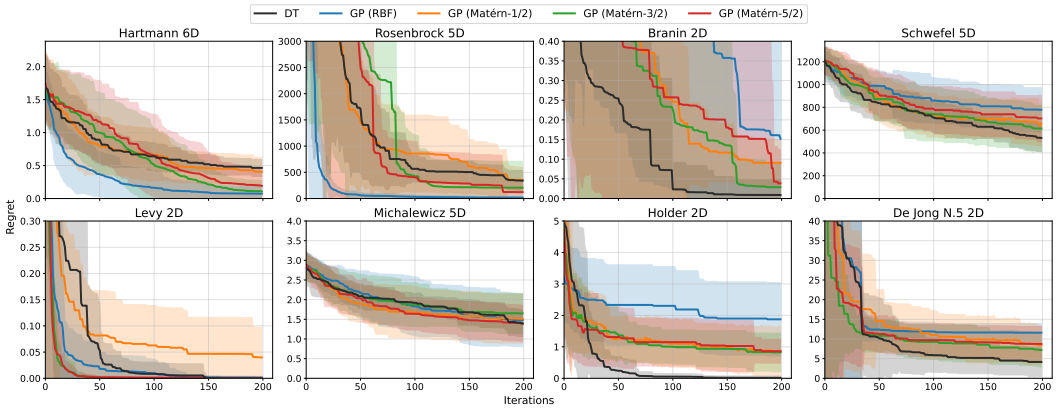

Figure 11: Convergence plots of GP with a varying kernel and our DT-PBO model. Results are averaged over 20 runs. The error bars represent one standard deviation.

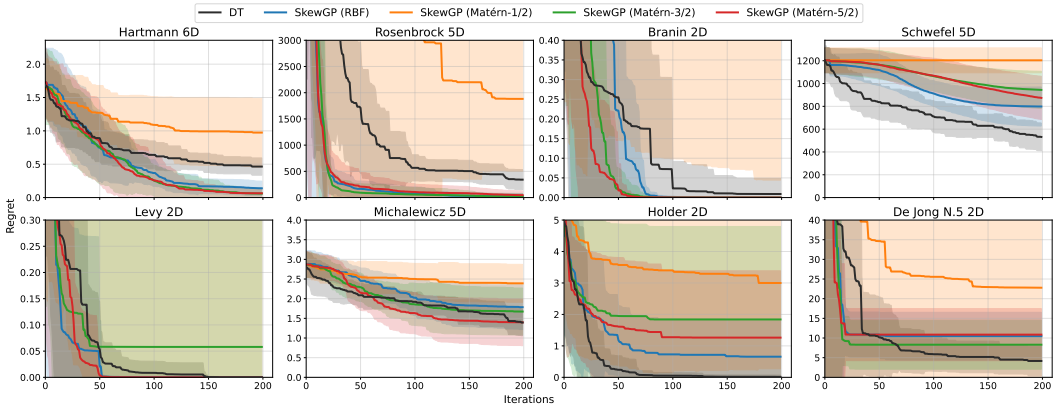

Figure 12: Convergence plots of SkewGP with a varying kernel and our DT-PBO model. Results are averaged over 20 runs. The error bars represent one standard deviation.

Matérn(1/2) gives the lowest final regret on every function except De Jong. Even so, DT-PBO has a lower mean final regret than POP-BO with any kernel on Schwefel, Hölder, and De Jong.

F.4 Ratio Ablation Study

In Figure 14, we show the hyperparameter sensitivity of the ratio hyperparameter. The differences in performance across hyperparameter settings are generally smaller than the associated uncertainty, with substantial overlap in error bars (standard deviation across runs). This indicates that the method is largely insensitive to this hyperparameter over the explored range, suggesting that extensive tuning is not required in practice. We only observe minor degradation at extreme values of the ratio.

F.5 Bradley-Terry likelihood

The current model assumes additive Gaussian observation noise, which induces the probit pairwise likelihood

$$\Pr(x_i \succ x_i' \mid \boldsymbol{f}) = \Phi\left(\frac{f(x_i) - f(x_i')}{\sqrt{2}\,\sigma_{\text{noise}}}\right).$$

This likelihood can be replaced by a Bradley–Terry logistic likelihood without changing the tree-construction procedure, the sum-to-zero identifiability correction, or the qEUBO acquisition function. The required modification only affects the likelihood term used to estimate the leaf utilities and to the corresponding gradient and Hessian in the Laplace approximation.

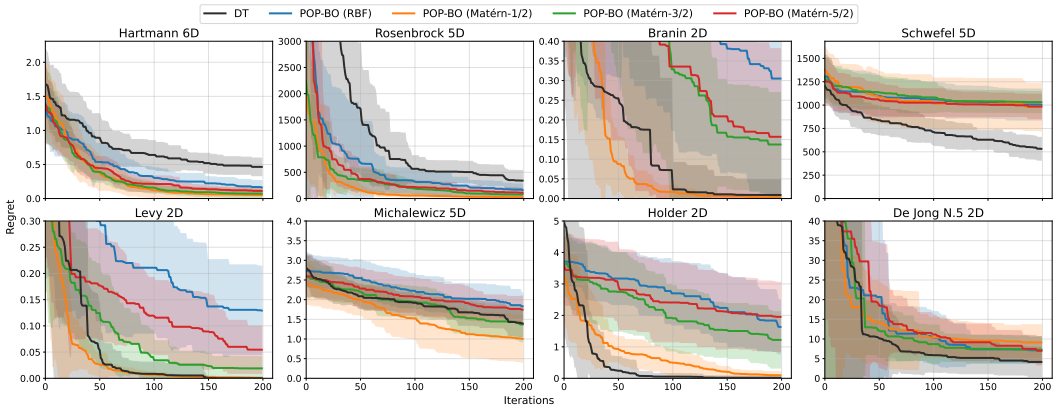

Figure 13: Convergence plots of POP-BO with a varying kernel and our DT-PBO model. Results are averaged over 20 runs. The error bars represent one standard deviation.

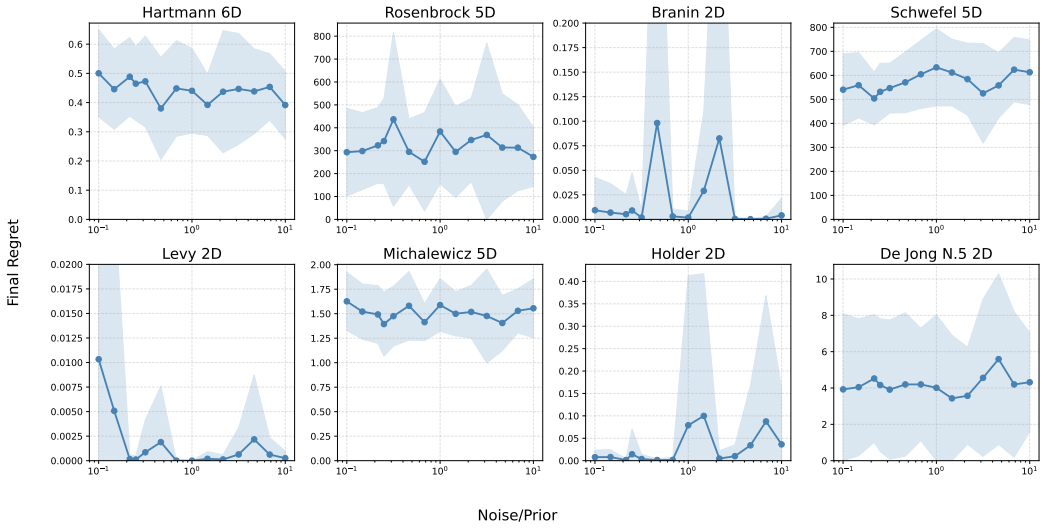

Figure 14: Ablation study of $\sigma^2_{noise}/\sigma^2_{prior}$ for DT-PBO. The y-axis is the regret after 200 iterations and the x-axis represents the different ratios. Results are averaged over 20 runs. The error bars represent one standard deviation.

Let the fitted tree contain m_ℓ leaves and let

$$\boldsymbol{f} = (f_1, \ldots, f_{m_\ell})^\top$$

denote their latent utilities. For comparison i, let w_i and ℓ_i denote the leaves containing the observed winner and loser, respectively, and define

$$\boldsymbol{a}_i = \boldsymbol{e}_{w_i} - \boldsymbol{e}_{\ell_i},$$

so that

$$\boldsymbol{a}_i^\top \boldsymbol{f} = f_{w_i} - f_{\ell_i}.$$

Under a Bradley–Terry model,

$$\Pr(x_i \succ x_i' \mid \boldsymbol{f}) = \mathrm{sigmoid}\left(\frac{\boldsymbol{a}_i^\top \boldsymbol{f}}{\tau}\right), \qquad \mathrm{sigmoid}(z) = \frac{1}{1+e^{-z}},$$

where $\tau > 0$ is a temperature parameter controlling the comparison noise.

Retaining the independent Gaussian prior used in the manuscript,

$$\boldsymbol{f} \sim \mathcal{N}\big(\mathbf{0}, \sigma^2_{\mathrm{prior}} I\big),$$

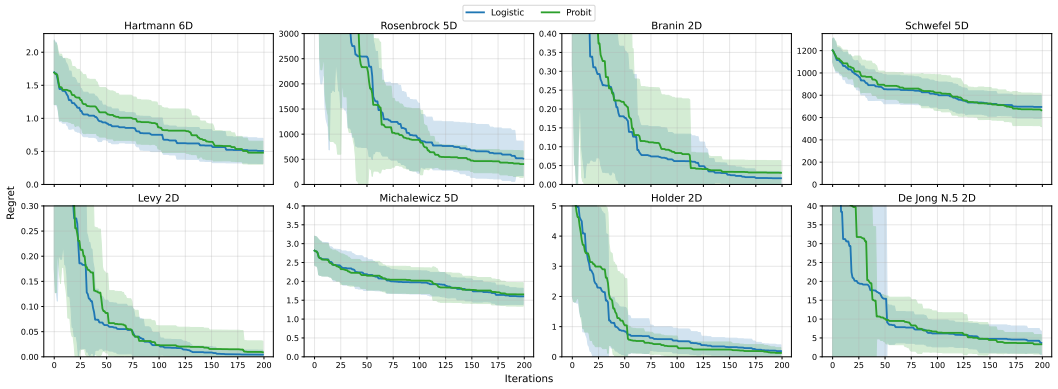

Figure 15: Average Regret over 20 runs for DT-PBO with a Thurstone likelihood vs a Bradley–Terry likelihood. The methods are tested using eight benchmarks. Error bars show one standard deviation.

the negative log-posterior, up to an additive constant, becomes

$$L(\boldsymbol{f}) = \sum_{i=1}^{n} \log\left[1 + \exp\left(-\frac{\boldsymbol{a}_i^\top \boldsymbol{f}}{\tau}\right)\right] + \frac{1}{2\sigma_{\text{prior}}^2} \boldsymbol{f}^\top \boldsymbol{f}. \tag{16}$$

Equivalently,

$$L(\boldsymbol{f}) = \sum_{i=1}^{n} \text{softplus}\left(-\frac{\boldsymbol{a}_i^\top \boldsymbol{f}}{\tau}\right) + \frac{1}{2\sigma_{\text{prior}}^2} \boldsymbol{f}^\top \boldsymbol{f}.$$

The MAP estimate is therefore

$$\boldsymbol{f}_{\text{MAP}} = \arg\min_{\boldsymbol{f}} L(\boldsymbol{f}).$$

Writing

$$p_i = \text{sigmoid}\left(\frac{\boldsymbol{a}_i^\top \boldsymbol{f}}{\tau}\right),$$

the gradient of Equation 16 is

$$\nabla L(\boldsymbol{f}) = -\frac{1}{\tau} \sum_{i=1}^{n} (1 - p_i)\boldsymbol{a}_i + \frac{1}{\sigma_{\text{prior}}^2} \boldsymbol{f}, \tag{17}$$

and the Hessian is

$$\nabla^2 L(\boldsymbol{f}) = \frac{1}{\tau^2} \sum_{i=1}^{n} p_i(1 - p_i)\boldsymbol{a}_i \boldsymbol{a}_i^\top + \frac{1}{\sigma_{\text{prior}}^2} I. \tag{18}$$

Hence, the Laplace approximation remains Gaussian,

$$p(\boldsymbol{f} \mid \mathcal{D}) \approx \mathcal{N}(\boldsymbol{f}_{\text{MAP}}, \Sigma_{\text{post}}),$$

with

$$\Sigma_{\text{post}} = \left[\frac{1}{\tau^2} \sum_{i=1}^{n} p_i^{\text{MAP}}(1 - p_i^{\text{MAP}})\boldsymbol{a}_i \boldsymbol{a}_i^\top + \frac{1}{\sigma_{\text{prior}}^2} I\right]^{-1},$$

where

$$p_i^{\text{MAP}} = \text{sigmoid}\left(\frac{\boldsymbol{a}_i^\top \boldsymbol{f}_{\text{MAP}}}{\tau}\right).$$

Thus, relative to the current formulation, one only replaces the probit negative log-likelihood and its derivatives by Equation 16– Equation 18.

In Figure 15, we show the performance difference of using the standard Thurstone likelihood model vs the Bradley–Terry likelihood model. Results show that performance of the two falls within one standard deviation of one another.

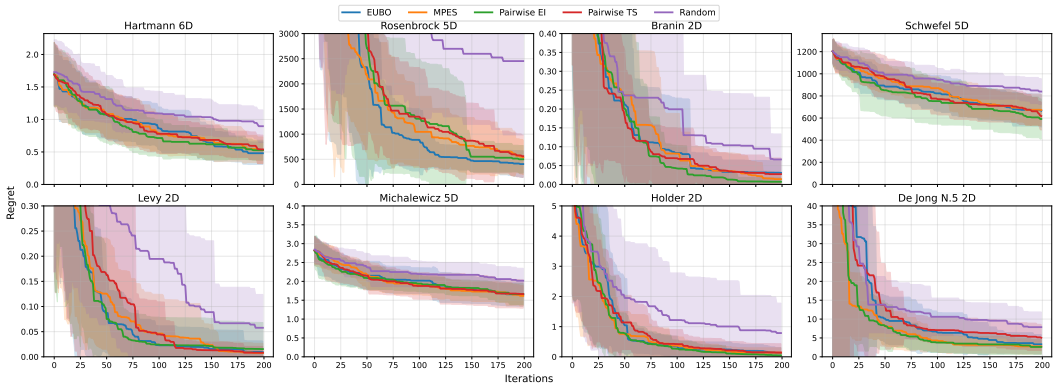

Figure 16: Average Regret over 20 runs for DT-PBO with varying acquisition functions. The methods are tested using eight benchmarks. Error bars show one standard deviation.

F.6 DT-PBO's Acquisition Function Ablation

To test the performance of various acquisition functions, we compare four acquisition functions specialized for preference learning: EUBO [3], pairwiseEI [50], pairwiseTS [50], and MPES [37] and random query selection. Figure 16 show convergence plots of DT-PBO combined with the aforementioned acquisition functions. The plot shows that there is not a single acquisition function that always performs the best. However, all methods clearly outperform random query selection showing the benefit of DT-PBO's active learning.

F.7 DT-PBO's Scalability

Running time Figure 17 shows the cumulative running time on needed for 200 iterations for each benchmark optimization functions. After each iteration, the model is fitted, the most informative PC is queried, and its result is added to the dataset. It can be seen that all kernel-based methods scale much worse on sample size. In particular, skewGP and POP-BO take much longer than the alternatives. Comparing the running time after 200 iterations (see Table 7), our proposed method takes approximately 12s on both the 2D and 5D/6D functions. GP takes around 200s on 2D functions and 300-400s on 5D/6D functions. SkewGP takes much longer and can run for around 1200s for the 2D function and around 2000-4000s on the 5D/6D functions. POP-BO takes around 6000-12000 seconds for varying functions. These results show that DT-PBO scales much better in sample size. Most of the running time benefits come from the fact that DT-PBO does not require hyperparameter optimization and that it reduces the Hessian matrix from the number of unique items m to the number of leaves m_l which means that the following matrix inversion takes $\mathcal{O}(m_l^3)$ instead of $\mathcal{O}(m^3)$, and usually $m_l << m$.

`Qwen3.5-27B` is run as a vLLM in Python. The running times for one feature extraction per patient and one feature suggestion is shown in Table 6

Table 6: Running time for the LLM in the PMR case study.

	Feature extraction	**Feature suggestion**
Running time (s)	0.323 ± 0.05	32.15 ± 0.17

Compute resources. Most experiments, including the case studies, were run on a 32 core AMD EPYC 7502P CPU and NVIDIA A100 80GB GPU (only used for Qwen), with 256 GB RAM memory, using Python 3.10 on Red Hat Enterprise Linux 8.10 (Ootpa). The exception is the synthetic benchmark studies which report the running times (i.e. Figure 17 and Table 7). These were run on Windows 11 with a 10 core Intel i7-1265U CPU and 16GB RAM (GPU was not used) using Python 3.10 on Windows 11 (POP-BO on python 3.9).

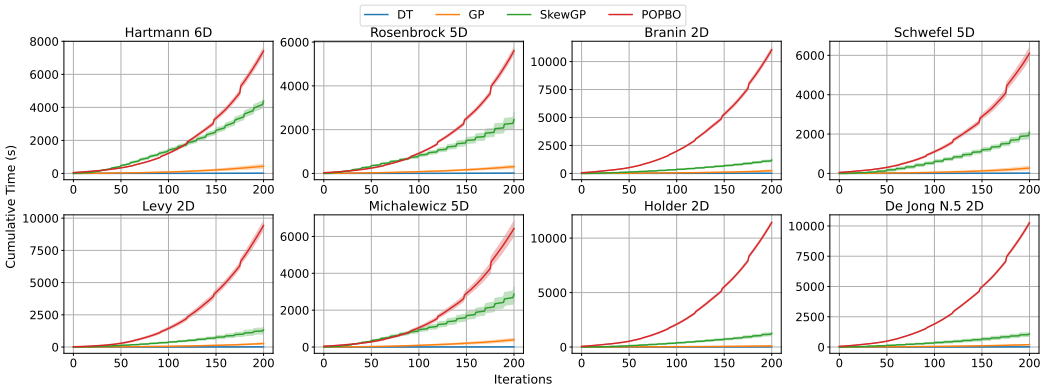

Figure 17: Cumulative running time for each benchmark optimization function for DT, GP and SkewGP. Results are averaged over 20 runs. The error bars represent one standard deviation.

Table 7: Final average runtimes (seconds) for the models across all eight benchmark functions. Results are averaged over 20 runs. The errors represent one standard deviation.

Function	**DT**	**GP**	**SkewGP**	**POP-BO**
Rosenbrock 5D	11.66 ± 1.07	308.23 ± 74.36	2461.15 ± 319.30	5603.30 ± 146.73
Hartmann 6D	11.49 ± 0.49	436.43 ± 122.44	4371.08 ± 224.22	7414.88 ± 207.53
Branin 2D	15.33 ± 1.81	246.50 ± 76.88	1194.08 ± 95.82	11043.56 ± 208.64
Levy 2D	14.27 ± 1.24	273.81 ± 56.14	1345.60 ± 264.21	9431.44 ± 337.55
Michalewicz 5D	11.31 ± 0.99	394.72 ± 85.11	2870.25 ± 396.14	6430.49 ± 399.14
Schwefel 5D	12.19 ± 0.94	271.44 ± 102.80	2086.21 ± 175.10	6103.73 ± 291.92
Holder 2D	16.91 ± 1.62	95.52 ± 45.21	1252.26 ± 145.76	11432.90 ± 125.04
De Jong N.5 2D	15.71 ± 1.76	195.34 ± 18.52	1091.30 ± 158.41	10256.36 ± 140.63

Dimensions Figure 18 presents the regret after 200 iterations for the Rosenbrock, Levy, Schwefel, and Michalewicz functions across varying dimensionalities. The results show that our model performs comparatively poorly on the Rosenbrock and Levy functions as the number of dimensions increases. A plausible explanation is that GP–based models benefit from the smoothness assumptions encoded in their kernels, which allow them to approximate these functions effectively with relatively limited data. In contrast, our model does not impose such assumptions and therefore requires more data to achieve comparable accuracy, particularly in higher-dimensional settings. Additionally, Decision trees can only model axis-aligned (vertical or horizontal in 2D) splits and therefore struggle with diagonal decision boundaries. Rosenbrock has a thin ridge of near-optimal points with strong variable interactions which cannot be efficiently approximated by the tree's axis-aligned splits, requiring many fragmented regions to capture the structure. POP-BO, which also relies on a kernel, behaves like the GP-based models: its regret on Rosenbrock and Levy remains far below that of DT-PBO as the dimension increases. For the Schwefel and Michalewicz functions, all models exhibit similar performance across dimensions. This suggests that the inductive bias introduced by smoothness assumptions is less beneficial for these functions, possibly because their landscapes are more irregular or contain structural properties that are not well captured by standard GP kernels. Although our model shows comparatively lower performance on the Levy and Rosenbrock functions beyond approximately 7–8 dimensions, interpretability becomes increasingly challenging in such high-dimensional settings. In particular, research has shown that making pairwise comparisons between two alternatives is only cognitively valid up to 5-dimensions [29]. Consequently, PBO for decision support is typically applied in lower-dimensional spaces to remain manageable for the DM. This suggests that, despite its reduced performance in higher dimensions, our model remains broadly applicable in the settings for which it is intended.

For all the four functions optima are known for each dimension, For Michalewicz they are only known until 10D [55]. For the remainder of the optima we have extrapolated the pattern in optima until 10D to 15D. The optima might not reflect the actual optima exactly, but for the purpose of

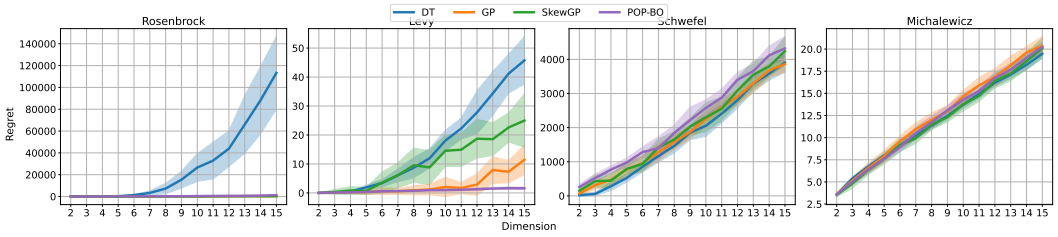

Figure 18: Final regret after 200 iterations for GP, skewGP, POP-BO, and DT-PBO whilst varying the function's dimensions for Rosenbrock, Levy, Schwefel, and Michalewicz. Results are averaged over 20 runs. The error bars represent one standard deviation.

finding the regret and comparing the models to each other this does not matter. Each model uses the same optimum so therefore, the results are comparable.

F.8 Noise

We model noisy utilities by

$$\tilde{u}(A) = u(A) + \epsilon_A, \quad \tilde{u}(B) = u(B) + \epsilon_B, \quad \epsilon_A, \epsilon_B \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2).$$

A noisy pairwise comparison is generated as $A \succ B$ if $\tilde{u}(A) > \tilde{u}(B)$, which is equivalent to

$$u(A) - u(B) > \epsilon_B - \epsilon_A.$$

Since ϵ_A and ϵ_B are independent, their difference follows

$$\epsilon_B - \epsilon_A \sim \mathcal{N}(0, 2\sigma^2),$$

which yields the probit likelihood defined in Section 2.2

$$P(A \succ B) = \Phi\left(\frac{u(A) - u(B)}{\sqrt{2}\,\sigma}\right).$$

To scale the noise relative to the utility function, we sample 10 000 random pairs and take the median absolute utility difference

$$\text{scale}(f) = \text{median}_{(x,x')} \left|u(x) - u(x')\right|,$$

and set

$$\sigma = \frac{r \cdot \text{scale}(f)}{\sqrt{2}}.$$

Substituting this into the likelihood gives

$$P(A \succ B) = \Phi\left(\frac{u(A) - u(B)}{r \cdot \text{scale}(f)}\right).$$

For different values of r, we compare the final regret of our model with the kernel-based models. Figure 19 shows that our model is robust to noise. The results show that in noisy settings DT-PBO performs equal or better than kernel-based models, except for Hartmann 6D, where GP and POP-BO still outperforms DT-PBO.

F.9 Tree Depth

For every benchmark, we varied the maximum tree depth to see whether interpretable trees can still reliably model the benchmarks. In Figure 20, we show the final regret after 200 iterations for every benchmark. The results show that for all functions but Michalewicz optimal results can already be achieved at a depth of $4 - 6$, at which trees are generally considered interpretable. Only for Michalewicz deeper trees still reduce the regret, although only slightly in comparison to depth 6.

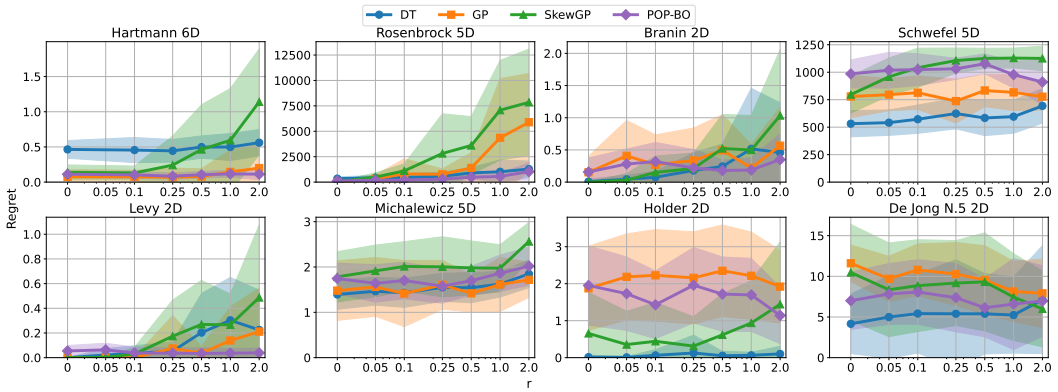

Figure 19: Final regret after 200 pairwise comparisons whilst varying the noise. The x-axis represents r which represents the noise level as a ratio of the median absolute difference in utility value of 10.000 randomly sampled pairs. A higher r means more noise is introduced. Results are averaged over 20 runs. The error bars represent one standard deviation.

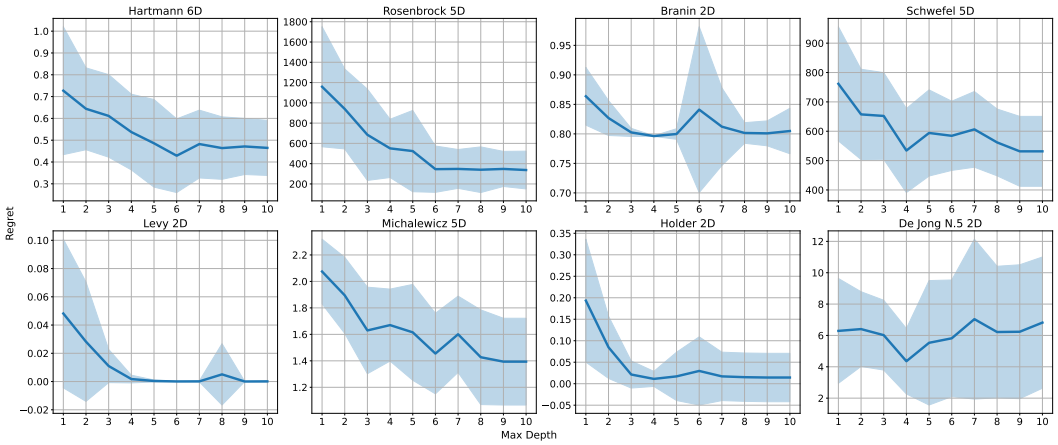

Figure 20: Final regret when varying the maximum tree depth for DT-PBO for every benchmark. The y-axis is the regret after 200 iterations and the x-axis represents the different ratios. Results are averaged over 20 runs. The error bars represent one standard deviation.

F.10 Exploration–Exploitation Behavior

To assess whether conditioning on a deterministic tree leads to overly exploitative queries, we compare DT-PBO with GP-qEUBO and a greedy DT variant that always samples both items uniformly from the leaf with the highest posterior mean. We use noiseless Michalewicz-2D with 20 seeds and 200 acquisition queries after 20 initial LHS comparisons, sharing the initial comparisons across methods for each seed. DT-PBO uses qEUBO without within-leaf prioritization, a noise-to-prior variance ratio of $1/4$, and a maximum depth of 50; the GP uses an RBF kernel and analytic qEUBO ($q = 2$).

As a behavioral proxy for exploration, we measure how far the queried items lie from the model's currently believed optimum. For a query $(\mathbf{x}_t, \mathbf{x}'_t)$ in a domain with lower and upper bounds $\mathbf{l}$ and $\mathbf{u}$, define

$$d_t = \frac{\|\mathbf{x}_t - \mathbf{z}_t\|_2 + \|\mathbf{x}'_t - \mathbf{z}_t\|_2}{2\|\mathbf{u} - \mathbf{l}\|_2},$$

where $\mathbf{z}_t$ is computed before observing the query outcome. For both DT variants it is the midpoint of the highest-mean leaf; for the GP it is the posterior-mean maximizer over a fixed set of 4,096 Sobol candidates. We average d_t over iterations 101–200 within each seed, then summarize these averages across seeds in Table 8.

Table 8: Distance of queried items from the model's believed optimum, normalized by the domain diagonal (mean ± sample standard deviation across 20 seeds).

Method	Normalized distance
DT-PBO	0.181 ± 0.063
GP-qEUBO	0.138 ± 0.022
Greedy DT (top leaf only)	0.022 ± 0.009

DT-PBO queries farther from its believed optimum on average than GP-qEUBO and the greedy DT variant. This suggests that its query behavior does not collapse to local exploitation on this benchmark. The metric is a spatial proxy, not a direct measure of information gain: the methods use different representations of their believed optimum, and even the greedy variant explores locally through uniform sampling within its top leaf. These results therefore do not establish an exploration guarantee or a general ordering across problems. They do, however, indicate that on this benchmark DT-PBO is not more exploitative than GP-qEUBO.

F.11 Extending DT-PBO Beyond Pairwise Queries: Best-of-q Feedback

We evaluate $q \in \{2, 3, 4\}$ on SUSHI-A and Michalewicz-5D. Each query returns only the preferred item and adds its $q - 1$ comparisons against the other items to the dataset. Thus, after n acquisition queries and n_0 initial pairwise comparisons, the model has received $n_0 + n(q - 1)$ comparison observations. Equal query budgets therefore differ in their comparison budgets. DT-PBO uses qEUBO without within-leaf prioritization in this ablation.

For SUSHI-A, we evaluate DT-qEUBO and GP-qEUBO on 100 distinct users, each evaluated once under every model–q configuration with the same four initial comparisons. Each run asks 25 acquisition queries. Following the recorded evaluation protocol, each user's metrics are averaged over the 25 model evaluations before these queries, corresponding to 0–24 answered acquisition queries. Table 9 reports the mean and sample standard deviation of these user-level averages across the 100 users; these are query-averaged metrics, not endpoint results. On SUSHI-A, $q = 4$ improves mean top-1 rank error over $q = 2$ for both models, although DT-qEUBO worsens at $q = 3$. The effect on ρ-regret is mixed: it decreases slightly for DT-qEUBO but increases for GP-qEUBO. A best-of-q answer supplies comparisons involving the winner but does not order the remaining items, so improved best-item identification need not improve the top-three ranking measured by ρ-regret.

Table 9: Best-of-q queries on SUSHI-A: query-averaged metrics (mean ± sample standard deviation across 100 paired users). Lower is better.

Model	q	ρ-regret	Top-1 rank error
GP qEUBO	2	0.571 ± 0.348	0.541 ± 0.690
	3	0.801 ± 0.562	0.334 ± 0.454
	4	0.837 ± 0.567	0.268 ± 0.414
DT qEUBO	2	1.008 ± 0.493	1.369 ± 1.049
	3	0.949 ± 0.600	1.568 ± 0.990
	4	0.924 ± 0.694	0.779 ± 0.604

For noiseless Michalewicz-5D, DT-PBO uses 20 paired seeds with the same 20 initial LHS comparisons across values of q and 200 acquisition queries per run. Table 10 reports regret after 50, 100, and 200 answered queries as mean ± sample standard deviation across seeds. Increasing q reduces mean regret at every reported query budget, from 1.655 to 1.169 after 200 queries.

Overall, $q = 4$ improves best-item identification on SUSHI-A and larger q reduces regret on Michalewicz-5D, while the effect on ρ-regret on SUSHI-A is mixed.

Table 10: DT-PBO best-of-q queries on Michalewicz-5D: regret at each query budget (mean $\pm$ sample standard deviation across 20 paired seeds).

q	Regret at 50 queries	Regret at 100 queries	Regret at 200 queries
2	2.166 ± 0.362	2.020 ± 0.350	1.655 ± 0.319
3	2.086 ± 0.323	1.747 ± 0.374	1.498 ± 0.385
4	1.982 ± 0.328	1.716 ± 0.274	1.169 ± 0.354

F.12 Quantitative Tree Complexity

Table 11 reports the complexity of the final DT-PBO models across 20 runs per benchmark, after 20 initial comparisons and 200 acquisition queries using the standard model settings. Depth is the maximum number of split conditions on a root-to-leaf path. The volume-weighted decision-path length averages these path lengths over leaves, weighting each leaf by its fraction of the search-domain volume; it is therefore the expected number of conditions evaluated for a uniformly sampled point in the domain. Across benchmarks, this average is approximately four conditions per prediction.

Table 11: Final DT-PBO tree complexity (mean $\pm$ standard deviation across 20 runs per benchmark). Path length is weighted by domain volume.

Problem	Leaves	Depth	Weighted path length
Branin 2D	11.95 ± 4.72	4.85 ± 1.15	3.69 ± 0.71
De Jong N.5 2D	19.80 ± 3.17	7.35 ± 0.91	4.50 ± 0.33
Holder 2D	13.65 ± 4.26	5.85 ± 0.79	4.39 ± 0.56
Levy 2D	16.50 ± 4.02	5.65 ± 0.85	3.99 ± 0.37
Michalewicz 5D	18.80 ± 3.06	6.05 ± 0.67	4.10 ± 0.33
Rosenbrock 5D	20.80 ± 1.96	7.30 ± 1.27	4.12 ± 0.27
Schwefel 5D	16.50 ± 6.94	6.10 ± 1.30	4.31 ± 0.75
Hartmann 6D	20.45 ± 2.42	6.45 ± 0.74	4.26 ± 0.18

F.13 GP-to-Tree Fidelity

We evaluate how closely a post-hoc regression tree can reproduce a GP's predicted preferences as its leaf count increases. For each of the eight benchmarks and 20 runs, we refit GP-qEUBO on its final comparison dataset and fit CART regression trees to its posterior mean. We consider leaf budgets $2, 4, \ldots, 4096$. The training-set size is $\max(50{,}000, 50B)$, where B is the largest requested leaf budget. With $B = 4096$, all trees in a run share 204,800 uniformly sampled, noise-free GP-labeled training points, supplying at least 50 training points per requested leaf. Evaluation uses 10,000 separate uniformly sampled points and 20,000 random pairs drawn from them.

Fidelity is the mean agreement between the tree's and GP's predicted preference on these held-out pairs. Matching strict preferences or ties receive credit 1, a tie in only one model receives credit $1/2$, and opposite strict preferences receive credit 0. Figure 21 plots fidelity against leaf count. For each run, $L90$ and $L95$ estimate the leaf counts needed to reach fidelity 0.90 and 0.95 by linearly interpolating the first crossing between tested sizes. Runs that do not cross a threshold by 4,096 leaves are reported as censored in Table 12; their required sizes remain unknown beyond the tested range.

DT-PBO needs no post-hoc approximation: its tree is the model itself, so its fidelity to its own utility predictions is exactly 1 at its original leaf count. The study checks that explicit rule traversal reproduces DT-PBO's predicted utilities to an absolute tolerance of 10^{-12} in every run. The stars in Figure 21 show each run at its own tree size. Fidelity here measures reproduction of the respective model's preferences, not agreement with the true utility function.

On the 2D benchmarks, mean $L95$ ranges from 90 to 236 leaves, compared with DT-PBO's mean sizes of 12–20 leaves. On each 5D/6D benchmark, a majority of runs do not reach 95% fidelity even

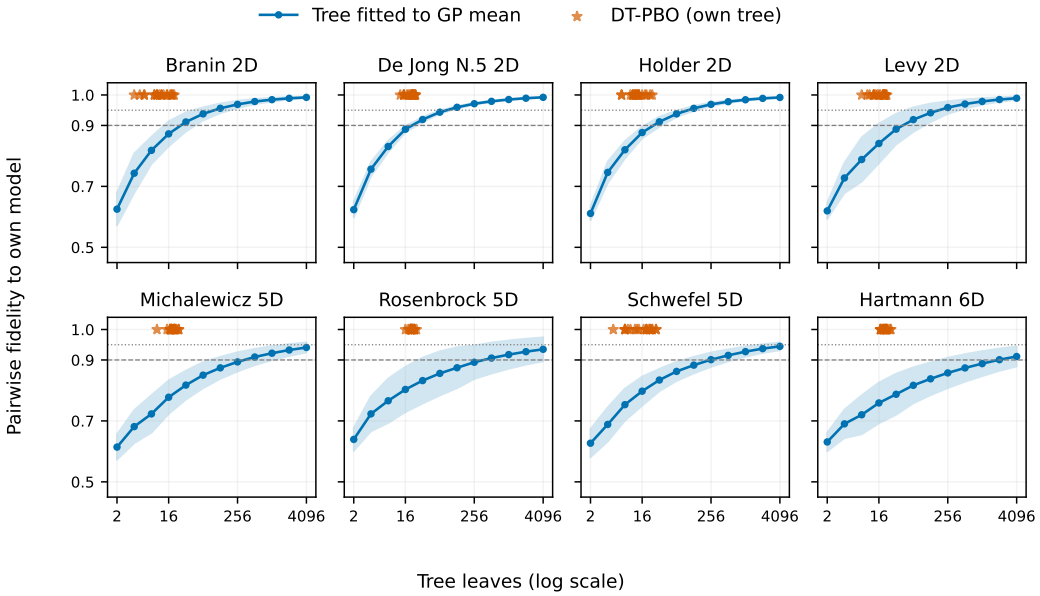

Figure 21: Fidelity versus tree size across eight benchmarks. GP curves show mean pairwise agreement with the GP posterior mean, with shaded bands of one standard deviation across 20 runs. Stars show DT-PBO's exact fidelity to its own model at each run's leaf count. Dashed and dotted lines mark 0.90 and 0.95 fidelity.

Table 12: Interpolated GP-to-tree fidelity thresholds. Means $\pm$ standard deviations include only runs reaching the threshold; adjacent columns count censored runs out of 20. A dash means no run reached the threshold.

	$L90$		$L95$	
Problem	Leaves	> 4096	Leaves	> 4096
Branin 2D	30.34 ± 21.89	0/20	122.20 ± 99.04	0/20
De Jong N.5 2D	22.31 ± 4.24	0/20	90.46 ± 17.40	0/20
Holder 2D	27.28 ± 9.24	0/20	110.36 ± 42.63	0/20
Levy 2D	58.22 ± 64.84	0/20	236.39 ± 270.44	0/20
Michalewicz 5D	653.52 ± 901.07	0/20	1787.62 ± 1079.49	13/20
Rosenbrock 5D	222.62 ± 250.47	2/20	3150.71 ± 704.27	14/20
Schwefel 5D	374.42 ± 377.38	0/20	2121.15 ± 1052.92	12/20
Hartmann 6D	1139.77 ± 1199.12	4/20	–	20/20

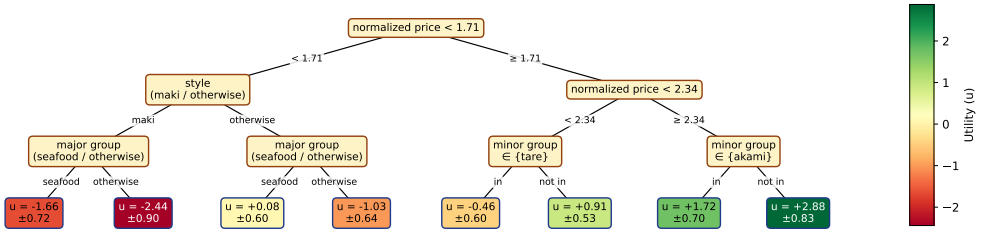

Figure 22: Learned tree for a user from the sushi dataset.

at 4,096 leaves. These results quantify the complexity of this post-hoc approximation of the GP posterior mean, however, they do not directly measure human comprehension or explain the complete acquisition process. They show that a tree faithful to the GP's preferences at the 95% level is several times larger than DT-PBO's own tree in 2D and, in most 5D/6D runs, exceeds 4,096 leaves. Taking tree size as a proxy for interpretability, DT-PBO's model is therefore considerably more compact than a post-hoc tree explanation of the GP.

Matched leaf budgets and self-distillation control. For each run, we also fit a regression tree to the GP posterior mean with its leaf budget set to that run's DT-PBO leaf count, L_{DT}. Its held-out pairwise fidelity, $F@L_{\mathrm{DT}}$, measures how faithfully the GP can be represented at the same tree size. To assess the overhead introduced by the tree-fitting procedure, we repeat the identical distillation pipeline with DT-PBO's own utility surface as the target, using the same training points, evaluation pairs, and leaf-budget grid. For this self-distillation control, $L99$ is the interpolated first crossing of 0.99 fidelity. This is distinct from the exact fidelity of DT-PBO's original tree: the control learns a new CART tree from its predictions.

Table 13: Follow-up fidelity controls (mean $\pm$ standard deviation across 20 runs per benchmark). GP fidelity is evaluated at each run's DT-PBO leaf budget; control $L99$ measures the size needed to reproduce DT-PBO's utility surface with 99% pairwise fidelity. All control runs reached the threshold.

Problem	DT-PBO leaves	GP $F@L_{\mathrm{DT}}$	Control $L99$
Branin 2D	11.95 ± 4.72	0.85 ± 0.05	16.76 ± 9.11
De Jong N.5 2D	19.80 ± 3.17	0.90 ± 0.01	24.97 ± 10.73
Holder 2D	13.65 ± 4.26	0.86 ± 0.04	14.43 ± 5.44
Levy 2D	16.50 ± 4.02	0.84 ± 0.06	18.73 ± 10.05
Michalewicz 5D	18.80 ± 3.06	0.78 ± 0.06	24.41 ± 8.50
Rosenbrock 5D	20.80 ± 1.96	0.82 ± 0.07	35.44 ± 14.95
Schwefel 5D	16.50 ± 6.94	0.79 ± 0.06	25.14 ± 14.08
Hartmann 6D	20.45 ± 2.42	0.77 ± 0.07	37.53 ± 10.93

At DT-PBO's leaf budget, mean GP fidelity ranges from 0.77 to 0.90 (Table 13). The self-distillation control reaches 99% fidelity at mean sizes of 14.4–37.5 leaves, compared with 12.0–20.8 leaves in the original DT-PBO trees. For every benchmark, the ratio of these mean sizes is below two, indicating modest fitting overhead for these tree-representable targets compared with the GP results.

G Use-cases details

G.1 Sushi

The sushi dataset consists of both item features and user features. The item features can be found in Table 14, and the user features can be found in Table 15. It consists of two datasets: dataset A where the same 10 different sushi were tried by all users, and dataset B, which includes a subset of 10 sushi items randomly selected from 100. For the user features subsection both datasets are used and for the remainder of the experiments dataset A is used.

Table 14: Item features of the sushi dataset.

Name	Type	Range
Style	binary	{0, 1}
Major group	binary	{0, 1}
Minor group	categorical	{0, ..., 11}
Oiliness	continuous	[0, 4]
Frequently eat	continuous	[0, 3]
Normalized price	continuous	[0, 1]

Table 15: User features of the sushi dataset.

Name	Type	Range
Gender	binary	{0, 1}
Age	categorical	{0, 5}
Time	continuous	[0, 1500]
Prefecture (< 15)	categorical	{0, ..., 47}
Region (< 15)	categorical	{0, ..., 11}
E/W (< 15)	binary	{0, 1}
Prefecture (curr)	categorical	{0, ..., 47}
Region (curr)	categorical	{0, ..., 11}
E/W (curr)	binary	{0, 1}
Prefecture chg	binary	{0, 1}

Table 16: Values assigned to each ranking which is used to calculate the regret.

Position	1	2	3	4	5	6	7	8	9	10
Value	1	2/3	1/3	1/7	2/7	3/7	4/7	5/7	6/7	1

Rho-regret calculation

The ρ-regret is defined as the difference between ranking R_{pred} and R_{true} represented by $\rho(R_{true}, R_{pred}) \geq 0$. The lower the true rank of the item that enters the top three, the higher the regret. For equal estimated ranks, competition ranking [18] is used, such that when the top three sushi fall in the same leaf, the regret is still zero; however if all sushi fall in the same leaf, the regret is high. This definition of regret is chosen due to its focus on the predictions of the top-ranked items, which, with the qEUBO acquisition function, is fairer than other ranking comparisons, such as the often chosen Kendall Tau, which focuses on the exact order of the ranking from top to bottom. Similarly, solely focusing on the single most preferred sushi says too little about the full prediction of the model. For the calculation, item is assigned a value based on its position in the ranking, which values are shown in Table 16. The regret is then defined as the sum of the values of all items that leave the top-3 from R_{true} and enter the top in R_{pred}. When highly ranked values leave the top-3 or when lowly ranked values enter the top-3, this will result in a high regret. The maximum value the regret can take would be: $\rho_{regret_{max}} = (1 + 2/3 + 1/3) + (5/7 + 6/7 + 1) = 6/3 + 18/7 \approx 4.57$.

Static score-scale baseline

To assess DT-PBO's ranking performance in comparison to another tree-based preference learning method, we compare DT-PBO with a static adaptation of the score-scale decision tree of Liu & Shih [33]. The score-scale tree splits on attributes of the judges, such as age or region; for a single user there is nothing to split on, so it reduces to ranking items by their accumulated scores: two points for a win and zero for a loss (the sushi rankings contain no ties). This baseline uses item identities rather than item features, so the comparison concerns the fixed set of ten sushi.

Both methods receive identical comparison histories for each of the 5,000 SUSHI-A users. For each user, we hold out one of the 45 possible pairs, select four initial comparisons using the same coverage-based initialization as the sushi experiments, and sample the remaining comparisons uniformly without replacement. We fit both models to the same history prefixes at budgets of 9, 15, 20, and 43 comparisons, including the four initial comparisons. DT-PBO uses random acquisition without within-leaf prioritization; thus the comparison isolates the surrogate model under a shared sampling protocol. Table 17 reports ρ-regret against each user's full ranking. DT-PBO has lower mean regret at all four budgets.

Incorporating user features

Up until now, when learning the preferences of a single user the optimization process is started from blank each time. Instead, when one has access to a database of users who already completed the

Table 17: Paired static-baseline comparison on SUSHI-A: ρ-regret (mean $\pm$ sample standard deviation across 5,000 users). Both methods use identical comparisons for each user. Lower is better.

Comparisons	DT-PBO (random)	Score-scale (random)
9	1.465 ± 0.889	1.640 ± 0.882
15	0.861 ± 0.678	1.198 ± 0.760
20	0.554 ± 0.536	0.951 ± 0.688
43	0.019 ± 0.090	0.098 ± 0.157

optimization process, one might want to use this historic data to speed up the learning process for new unseen users. Adding user features, however, presents a new problem. The consistency score is based on making splits between comparisons but comparisons are never made between users. Hence, the presented consistency score is not suitable for making splits on user features. A solution is to train a separate tree for user features, which uses a modified consistency score. Instead of leaf values, this user tree contains decision trees trained on item features from the comparisons in each leaf. While other approaches to combine user and item features exist, the mentioned approach is most interpretable, as it keeps both feature types separated and reduces the dimensionality each item tree must handle, which is something our algorithm struggles with. Essentially this approach builds a user DT that partitions users in clusters of similar users based on their preferences. For each cluster of users, a regular item DT is constructed to represent their joint preferences.

The User Consistency Score defined in Eq. 19 serves as the splitting criterion for the decision tree. The fundamental objective is to find splits that partition users into groups with the most internally consistent preferences. A group is considered consistent if, for any given pair of items (A, B), the users within that group strongly agree on the preference direction (i.e., most users state $A > B$ with very few stating $B > A$, or vice versa). For a given set of user comparisons D at a node, a split partitions D into two subsets, D_{left} and D_{right}. The gain G from this split is calculated as the sum of the consistency scores of the two resulting child nodes:

$$\begin{aligned} G(k,t) = &\sum_{i,j} C(i \succ j, D_{\text{left}}) - C(j \succ i, D_{\text{left}})| \\ &+ \sum_{i,j} |C(i \succ j, D_{\text{right}}) - C(j \succ i, D_{\text{right}})|. \end{aligned} \tag{19}$$

Where $C(i > j, D_k)$ is the count of comparisons where item i was preferred over item j within D_k. The model seeks to maximize this score, finding the split that creates child nodes with the strongest internal preference agreement.

During active learning, both the user- and the item tree are trained. The user tree determines which item tree a given user is assigned to. Then, that selected item tree is updated using the new user's data, with its prior set to the mean of the posterior obtained from the users previously assigned to that same item tree and a high uncertainty matrix to allow the tree to quickly adapt to the new user. The results for both dataset A and B are shown in Figure 23. This method of creating item trees at the leaves of user trees is not directly applicable to a GP, so the comparison with a GP is left out. For dataset A, using the posterior from previous users as the prior for item trees accelerates regret reduction, though the time to reach zero regret remains about the same. For dataset B, the final regret decreases as the item tree grows. However, it does not reach zero because the number of leaves is limited, meaning that some of the 100 sushi items inevitably share the same leaf. Overall, incorporating user data appears beneficial, but further refinement is needed to achieve significantly better results than without user data.

G.2 PMR feature extraction

An LLM is used to derive features from the patient messages, after which our decision tree is trained on these extracted features. As in Gatto et al. [20], `INBOX_A` is used for training and `INBOX_B` for testing. Ideally, such features would be specified directly by domain experts. In this study, however, we begin with a set of baseline features distilled from Gatto et al. [20]. To identify additional features, we construct a validation set from `INBOX_A`.

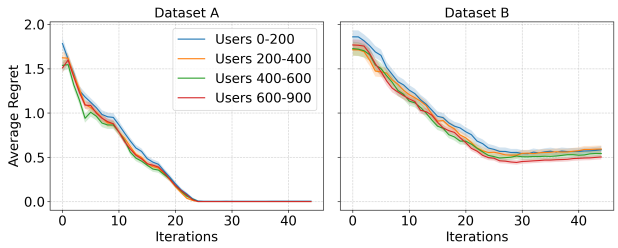

Figure 23: Average regret for 900 users as training the user tree progresses. After every 50 users, the user tree is updated. The maximum tree depth is limited to 5 for both trees. ± 1 Standard deviation is shown in the error bars for the users within these cohorts.

The intial features loosely extracted from the paper are:

- patient_requested_action
- message_actionability
- acuity_conflict

After this the feature extraction process was started. The LLM is prompted to propose candidate features for comparisons that (i) fall in the same leaf, (ii) have the same expected value but fall into different leaves, or (iii) fall into different leaves where the preferred patient has lower expected utility than the non-preferred one. The LLM is first prompted to reason why one patient was chosen over the other, for each comparison in the chosen PC set. This prompt is shown in 24. Then, the LLM is prompted to propose a feature out of those reasoning traces. This prompt is shown in 25. During the feature proposal, features that decrease the validation score are pruned.

Stage 1: Per-Comparison Reasoning Prompt

You are an expert clinical-decision analyst reviewing one patient-portal triage comparison.
The medical expert chose the winner over the loser. Explain the likely clinical rationale using concrete message/EHR details. Consider acuity, red flags, risk of deterioration, comorbidities, medications, recent encounters, diagnostic ambiguity, and vulnerability. If the choice seems counterintuitive, say so.
Return exactly JSON with keys: pair_index, split, clinical_rationale, certainty. certainty must be high, medium, or low.
Comparison: {comparison JSON}

Figure 24: The prompt shown to the LLM for each comparison.

The final list of extracted features is found in Table 18. Ordinal categorical features are converted to integers for the decision tree, message_actionability = 3 means same day traige or call, for example.

It should be highlighted that this extraction process is just one way to find these comparisons. It is not in any way meant to be the focus of this paper. The focus is showing the strength of the increased interpretability and applicability to other datasets. A leave-one-out feature sensitivity analysis is shown in Table 19. The pairwise tree test accuracy is most sensitive to removing `message_actionability`, which decreases test performance by 17.6 percentage points. This suggests that the LLM-extracted actionability signal captures a substantial part of the urgency information expressed directly in the patient message.

At the same time, the model does not rely exclusively on this feature: after removing `message_actionability`, the test accuracy remains above chance, indicating that other extracted features still support meaningful pairwise distinctions. The result also suggests that `message_actionability` is not merely a proxy for the remaining clinical descriptors, but provides a direct summary of how urgent or actionable the message appears to the LLM.

Stage 2: Feature Proposal Prompt

You are an expert clinical-decision analyst reviewing how primary-care physicians triage patient-portal messages.
TASK: Propose ONE typed feature derived from the comparison_reasoning below.
CONTEXT: These pairs share a decision-tree leaf because existing features are identical or non-discriminating. The new feature must be observable in plain message/EHR text, differ between winner and loser in EACH pair, and take the more-urgent value for the winner.
TYPE GUIDANCE: {type guidance} Allowed types: numeric, boolean, ordinal, categorical, multi_hot. Avoid high-cardinality categories and vague numeric scores unless the measurement is explicitly observable.
RULES: extraction_instruction must work for a single patient's message/EHR, without pairwise context, existing feature names, or logic over existing feature values. Avoid duplicate/rejected concepts, deterministic transforms of existing features, and broad restatements of urgency, severity, red flags, symptom category, actionability, or patient concern. Proposals are usually accepted on validation accuracy; test accuracy is used only when that run option is enabled.
OUTPUT — return exactly JSON with these keys: - name, type, allowed_values_or_range, extraction_instruction, rationale, failure_pattern, expected_variation.
Keep values concise. The proposed feature must be based on comparison_reasoning.
Failing leaf has {N} failure pair(s) ({train N} train, {val N} val).
Existing feature names: {existing feature names JSON}
Rejected proposal count this round: {count} Rejected feature names this round: {rejected names JSON} All rejected proposals this round (name, type, reason): {rejected summary JSON} Recent rejected proposal detail (last 8): {recent rejected JSON}
Comparison reasoning: {stage 1 reasoning JSON}
Failure examples: {failure examples JSON}

Figure 25: The prompt shown to the LLM to propose a new feature.

Table 18: Final features as a result of the feature extraction process.

Name	Type	Extraction instruction	Allowed values
symptom_ trajectory	categorical	Identify explicit trajectory language for the current issue. Use not_specified when no comparison with prior state is given.	rapidly_worsening, worsening, stable, improving, fluctuating, resolved, not_ specified
red_flag_ syndrome	multi_hot	Identify explicit or strongly implied red-flag syndromes in the current complaint. Do not mark a value for a remote history unless it is relevant to the current message.	acute_neuromuscular_weakness, new_ neurologic_deficit, syncope_or_near_ syncope, neck_stiffness_with_throat_ symptoms, systemic_febrile_illness, respiratory_distress_or_wheezing, chest_pain_or_possible_arrhythmia, new_bilateral_leg_edema, anaphylaxis_or_airway_allergy, significant_ fall_or_trauma, rapidly_spreading_ rash, possible_dehydration_or_poor_ intake
vascular_ thrombotic_risk_ context	multi_hot	Extract vascular or thrombotic risk factors relevant to the current complaint.	prior_dvt_or_pe, known_jugular_or_ unusual_site_thrombosis, anticoagulant_use, recent_flight_or_immobility, varicose_or_venous_disease, recent_ surgery_or_hospitalization, active_or_ recent_cancer, new_unilateral_limb_ symptom

Name	Type	Extraction instruction	Allowed values
infection_risk_context	multi_hot	Extract infection-related vulnerability or exposure context from the EHR and current message.	immunosuppressive_medication, chronic_steroid_use, diabetes_or_obesity, recent_urinary_or_genital_infection, recent_respiratory_infection, contagion_or_household_exposure_concern, possible_deep_neck_or_meningeal_source, older_age_with_fever
medication_complication_context	multi_hot	Mark medications or medication classes in the EHR/message that could plausibly contribute to the current complaint or increase triage risk.	sedating_or_fall_risk_medication, anticoagulant_or_bleeding_risk, steroid_or_immunosuppressant, beta_blocker_or_rate_control, diuretic_or_electrolyte_affecting, ace_arb_or_potassium_affecting, statin_or_myopathy_risk, gabapentinoid_or_edema_sedation_risk, glp1_or_metabolic_medication, polypharmacy
frailty_or_fall_harm_risk	ordinal	Assess risk of fall or harm from reduced mobility using current symptoms, age, medications, and EHR context.	none (0), mild (1), moderate (2), high (3)
diagnostic_uncertainty	ordinal	Estimate how broad or consequential the diagnosis is for the current complaint based on explicit symptoms and EHR context.	low (0), moderate (1), high (2)
patient_requested_action	categorical	Classify the most time-sensitive action explicitly requested by the patient or sender.	same_day_call_or_reply, same_day_visit_or_triage, pre_event_guidance, asks_whether_to_come_in, general_question_or_reassurance, informational_fyi, no_clear_request
message_actionability	ordinal	Assess how much clinician action is needed now from the content of the message, not from the pair label.	none_or_fyi (0), asynchronous_advice (1), schedule_routine_visit_or_labs (2), same_day_triage_or_call (3), emergency_or_immediate_evaluation (4)
symptom_distribution_or_spread	categorical	Classify whether symptoms are localized, bilateral/symmetric, systemic, spreading, or not specified.	localized, bilateral_or_symmetric, systemic, spreading_or_migrating, not_specified
acuity_conflict	categorical	Classify whether the apparent urgency of the patient's wording conflicts with the clinical content. Use none when wording and content align or there is no clear urgency signal.	none, urgent_wording_low_acuity_content, calm_wording_high_acuity_content, resolved_event_but_high_risk_history

Feature removed	Train acc. (%)	Δ train (pp)	Test acc. (%)	Δ test (pp)
None (baseline)	83.9	–	69.9	–
message_actionability	83.8	-0.1	52.3	-17.6
red_flag_syndrome	83.9	0.0	58.6	-11.3
diagnostic_uncertainty	83.9	0.0	64.3	-5.6
patient_requested_action	83.9	0.0	64.5	-5.4
vascular_thrombotic_risk_context	83.9	0.0	66.1	-3.8
medication_complication_context	83.9	0.0	67.3	-2.6
symptom_trajectory	83.9	0.0	67.5	-2.4
acuity_conflict	83.9	0.0	69.9	0.0
frailty_or_fall_harm_risk	83.9	0.0	69.9	0.0
infection_risk_context	83.9	0.0	69.9	0.0
symptom_distribution_or_spread	83.9	0.0	69.9	0.0

Table 19: Leave-one-out sensitivity of the pairwise tree feature set on PMR pairwise accuracy.

G.3 BELA

BELA [34] is a discrete choice experiment in which each respondent completes 18 choice tasks, each comparing between two treatment profiles, A and B, characterized by 10 attributes, alongside an explicit opt-out option. The opt-out comparisons are ignored as they provide no information for the utility function. Three of the ten attributes are varied between tasks, and no externally known utility function or optimum is available. The dataset contains 434 respondents of which 6 respondents who selected opt-out in more than 80% of their tasks were excluded. The attributes can be found in Table 20.

Figure 27 shows the tree learned for one respondent. The tree indicates that this respondent prefers a higher or equal than 85% sensitivity, especially in the case of mammography screening, somewhat prefers cheaper options, and prefers invitation by telephone. Because these preferences are clearly interpretable from the tree, the respondent could be asked to verify them and, where appropriate, edit the tree directly. Future treatment recommendations could then be personalized with minimal further questions. Under PBO, the optimal treatment could be identified in a few queries, and over actual candidate treatments rather than synthetic profiles constructed by varying three of the ten attributes. Train and test accuracies are reported in Table 21, where 16 random PCs are chosen as training set and 2 as test set. As the DT is kept shallow for interpretability, quite some ties between PCs are estimated by the DT, which explains the non perfect accuracy on the training set compared to GP. Both DT and GP have difficulties generalizing to the test set.

G.4 Presidential candidates dataset

The Presidential Candidates Conjoint dataset [5] is an experiment in which respondents choose between two hypothetical presidential candidates described by between four and twenty attributes, including features such as the largest campaign contributor, gender, and party affiliation. A detailed overview of the dataset can be found in Table 22. A learned decision tree (an example is shown in Figure 28) gives the preferences for a single respondent, making their preferences directly interpretable. This setting illustrates why interpretability matters in preference learning: when attributes include sensitive features such as gender, race, or religion, a black-box preference model offers no way to detect when predicted utilities depend on these attributes, whereas the tree makes any such dependence explicit and open to inspection by the respondent. The approach also extends naturally to group-level analysis: respondents can be clustered on demographic features, or the extension to user features explained in Section G.1 can be applied, to reveal shared preference structures across subpopulations. For the train and test accuracies in Table 21, 27 random PCs are chosen as training set and 3 as test set. On the test set the DT seems to perform slightly better, while on the training set the same as for BELA applies.

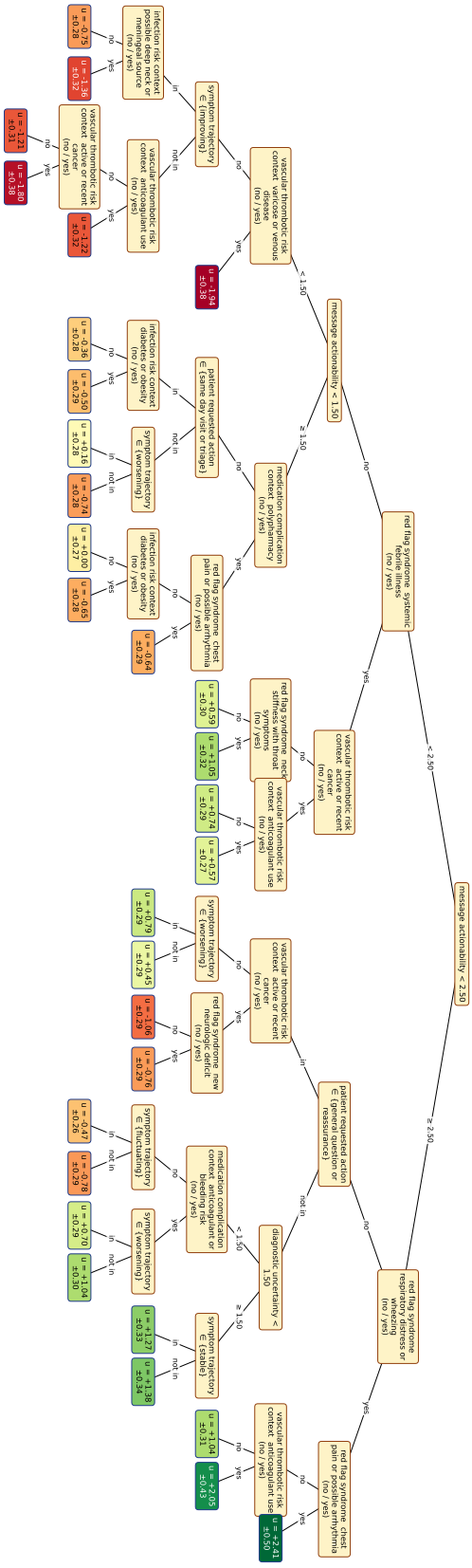

Figure 26: Full DT table for the PMR case study.

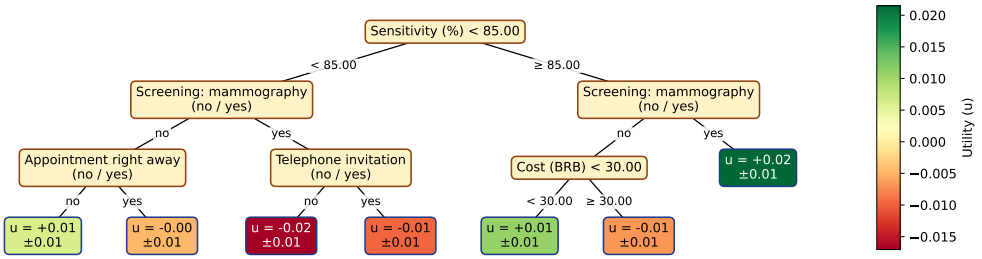

Figure 27: Learned DT for a user from the BELA dataset.

Table 20: Attributes and levels in the discrete choice experiment.

N	Attributes	Definition	Levels
1	Way of invitation	The approach a women prefers to be invited to screening	Postal letter / Telephone call
2	Possibility to arrange the appointment right away	Possibility to get the appointment arranged during the time of invitation (or fixed-appointment scheme)	Yes / No
3	Comprehensive information about screening	Receiving comprehensive information on breast cancer and screening during the invitation	Yes / No
4	Total travel time	Total travel time required for women to get from home to screening facility	20 minutes/ 40 minutes/ 60 minutes/ 90 minutes
5	Waiting time	Waiting time in healthcare facility during the screening visit	20 minutes/ 40 minutes/ 60 minutes
6	Perception of the physician as “a good doctor”	Perceiving the physician conducting the screening test as a “good” one, either because of the personal previous experience or trusted recommendation	Yes / No
7	Screening modality	Approach by which the breast cancer screening is conducted	Manual examination/ Mammography / Manually and by mammography
8	Test sensitivity	Ability of the test to detect cancer when a woman has it	60% / 70% / 80% / 90%
9	Possibility to combine the screening with other medical visits	Possibility to address several health issues within one visit to healthcare facility (for example, another screening test)	Yes / No
10	Cost of the test	Out-of-pocket costs of the screening (not reimbursed)	0 BRB / 20 BRB / 40 BRB

Table 21: Training and Test accuracy for the BELA and Presidential datasets

	Training acc.	Test acc.
Pairwise Tree		
BELA	0.99 ± 0.05	0.55 ± 0.33
Presidential	0.93 ± 0.07	0.68 ± 0.27
Pairwise GP		
BELA	1 ± 0.0	0.55 ± 0.40
Presidential	1 ± 0.0	0.62 ± 0.29

Table 22: Attributes and options in the presidential conjoint experiment.

Attribute	**Options**
Gender	Male / Female
Age	36 / 45 / 54 / 63 / 72
Race/Ethnicity	Asian American / Black / Hispanic / White
Religion	Catholic / Evangelical Protestant / Mainline Protestant / None
Highest education	Graduated from high school / Graduated from college
Profession	Business owner / Farmer / Fire fighter / Lawyer
Annual income	\$32k / \$75k / \$180k / \$5.1m
Marital status	Single / Married / Divorced
Party affiliation	Democrat / Republican
Prior elected office	None / State attorney general / Governor / U.S. senator
Military service	No military service / Served in U.S. military
Largest campaign contributor	Auto workers' unions / Oil companies / Teachers' unions / Wall Street firms
Position on abortion	Pro-choice / Neutral / Pro-life
Position on gay marriage	Favors gay marriage / Opposes gay marriage
Position on health care	Government should do more / Government should do less
Religious activity	Never attends church / Occasionally attends church / Attends church weekly / Prays daily
Favorite music	Classical / Country / Hip hop / Rock
Favorite professional sport	Baseball / Basketball / Football / Soccer
State of residence	Alabama / Colorado / Massachusetts / Ohio
Car	Ford Sedan / Ford pick-up truck / Toyota Sedan / Toyota pick-up truck

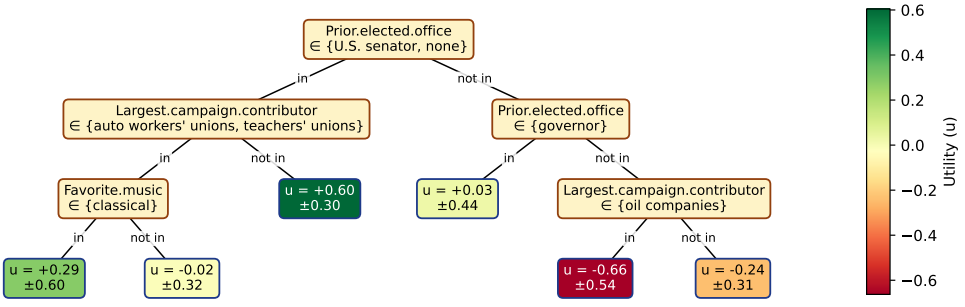

Figure 28: Learned tree for a user from the presidential candidates dataset.

Table 23: Existing assets used in this work.

Asset	Source / creator	License or terms	Use in this paper
Sushi preference dataset	Kamishima et al.	Research use permitted; redistribution prohibited	Sushi PBO case study
BELA dataset	Mandrik et al.	CC BY 4.0	Breast-cancer screening preference case study
PMR-Synth-Inboxes	PortalPal-AI / Gatto et al.	CC-BY-NC-4.0	PMR feature-extraction and tree case study
Qwen/Qwen3.5-27B	Qwen Team	Apache-2.0	LLM feature extraction for PMR
BoTorch	Meta/PyTorch	MIT	GP-qEUBO baseline
SkewGP-HB code	Takeno et al.	MIT	SkewGP-HB-EI baseline
Presidential conjoint dataset	Bansak et al.	CC0 1.0	Presidential preference case study

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract provides a brief overview of the main contributions of the paper. The final part of the introduction also summarizes the contributions.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Section 5, including the inability to model indifference statements and reduced performance on high-dimensional non-rugged functions. Additional robustness, noise, and scalability analyses are provided in Appendix F.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: An outline of the proofs is given in the main paper. Complete proofs are present in the appendix. Assumptions are outlined in both.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The algorithm is specified in Section 3 and Algorithm 1. Experimental settings, benchmark definitions, baselines, hyperparameters, optimizers, data splits, and evaluation metrics are described in Section 4 and Appendix F; code is released in an anonymous repository.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The anonymous code repository contains installation instructions, required package versions and scripts for reproducing the main experiments and baselines and third party datasets which possess the right licenses.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental settings are described in the main paper and in more detail in the appendix. Finally, the code is also published.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For repeated numerical experiments, results are averaged over 20 runs with different initial comparison samples, and standard deviations are reported. For fixed real-data case studies, train/test accuracies are reported separately and are not presented as statistical significance claims.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources are described in the appendix.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The ethics code has been reviewed and adhered to.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our method could be used in decision-support systems but the core of our idea is to make it interpretable such that it can be validated. We describe this positive impact and do not see a negative impact that will be caused by our model directly.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release a trained clinical triage model, LLM weights, scraped datasets, or a deployed decision system. The released asset is research code for DT-PBO; the PMR experiment is presented as an illustrative case study and is not intended for autonomous clinical deployment.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing datasets, codebases, and models used in the paper are cited, and their licenses or terms of use are summarized in Appendix Table 23. We do not redistribute third-party datasets where redistribution is prohibited.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release anonymized research code for DT-PBO, including documentation, installation instructions, experiment scripts, configuration files, and a license. The README describes intended use, limitations, and how to reproduce the main experiments.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The authors did not conduct new crowdsourcing experiments or collect new human-subject data. The paper analyzes existing publicly available or previously released datasets, and the original sources are cited.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The authors did not conduct new human-subject experiments or interact with participants. Existing datasets were used according to their released terms.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLM usage is described in Appendix G.2. We used Qwen3.5-27B as a feature-extraction and feature-proposal tool for the PMR case study; the prompts, extracted features, and role of the LLM are documented. The core DT-PBO algorithm itself does not rely on LLMs.