

# LILLO: BAYESIAN OPTIMIZATION WITH INTERACTIVE NATURAL LANGUAGE FEEDBACK

Anonymous authors

Paper under double-blind review

## ABSTRACT

For many real-world applications, feedback is essential in translating complex, nuanced, or subjective goals into quantifiable optimization objectives. We propose a language-in-the-loop framework that uses a large language model (LLM) to convert unstructured feedback in the form of natural language into scalar utilities to conduct BO over a numeric search space. Unlike preferential BO, which only accepts restricted feedback formats and requires customized models for each domain-specific problem, our approach leverages LLMs to turn varied types of textual feedback into consistent utility signals and to easily include flexible user priors without manual kernel design. At the same time, our method maintains the sample efficiency and principled uncertainty quantification of BO. We show that this hybrid method not only provides a more natural interface to the decision maker but also outperforms conventional BO baselines and LLM-only optimizers, particularly in feedback-limited regimes.

## 1 INTRODUCTION

Bayesian optimization (BO) is a powerful strategy for optimizing expensive-to-evaluate black-box objectives (Brochu et al., 2010; Shahriari et al., 2015; Frazier, 2018). Real BO deployments, however, frequently optimize for complex tradeoffs over many outcomes, where the objective is not clearly specified, and optimization must be guided by feedback from a decision maker (DM) rather than by direct access to ground-truth utility values. Existing preference BO methods address this by learning from comparisons or ratings (González et al., 2017; Chu & Ghahramani, 2005; Lin et al., 2022; Feng et al., 2024), yet they typically require narrow feedback formats, custom likelihoods models, and careful kernel or hyper-prior engineering for the surrogate model. In contrast, black box optimization approaches driven by large language models (LLMs) end-to-end offer a more flexible interface: they can translate rich natural language input and bring in domain knowledge (Liu et al., 2024; Yang et al., 2023; Brown et al., 2020). But unlike BO, such approaches usually lack calibrated uncertainty estimates, principled exploration-exploitation tradeoffs, and reliable convergence guarantees.

In this paper, we introduce *Language-in-the-loop Optimization* (LILLO), a framework designed to combine the complementary strengths of BO and LLMs while avoiding their respective weaknesses. Rather than letting the LLM directly control the optimization, we place it in a supporting role: the LLM interprets free-form feedback and prior knowledge expressed in natural language, and translates them into structured signals for Gaussian process (GP) utility surrogates. This design leverages LLMs’ strengths at understanding language and contextual reasoning, while addressing the instability, poor sample efficiency, and weak generalization that arise when relying on LLMs’ raw in-context learning (ICL) abilities (Brown et al., 2020; Zhao et al., 2024; Yin et al., 2024). By mapping natural language feedback into quantitative utilities that GPs can model, LILLO preserves BO’s calibrated uncertainty and acquisition mechanisms. Beyond showing that such a hybrid is feasible, we provide a systematic study of the design choices that make it effective and practical.

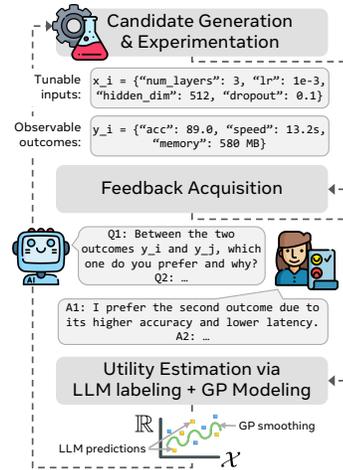


Figure 1: Core steps of LILLO.

The result is a principled, empirically validated framework that combines the expressivity of LLMs with the reliability of BO. In summary, our main contributions are as follows:

1. We propose LILLO, a novel BO approach for settings that require eliciting utility information from a DM. LILLO leverages natural language feedback, which can be more information-rich than pairwise comparisons or scalar ratings commonly used in other methods.
2. We show how to translate such natural language feedback into quantitative latent utilities that can be used effectively by a surrogate and acquisition function, systematically exploring the design choices required to render this approach both effective and practical.
3. We demonstrate the improved performance and sample efficiency of LILLO compared to existing baselines on a variety of synthetic and real-world environments, outperforming both classic GP-based and LLM-based Bayesian optimization methods.
4. We show that the LLM is able to not merely serve as a “utility translator”, but also successfully leverage its domain knowledge acquired during pre-training, enhancing the optimization.

### 1.1 PROBLEM DEFINITION

We consider the problem of optimizing the outputs of a black-box system with respect to preferences of a DM. Let  $\mathcal{X} \subset \mathbb{R}^d$  denote the search space of controllable parameters, and let each  $y \in \mathcal{Y} \subset \mathbb{R}^k$  define the outcomes of an experiment obtained through an expensive-to-evaluate function  $y = f(x)$ , where  $f : \mathcal{X} \rightarrow \mathcal{Y}$  maps parameters to outcomes in the outcome space  $\mathcal{Y}$ .

A DM associates with each observed outcome  $y$  a latent utility value  $u = g(y) = (g \circ f)(x) \in \mathbb{R}$ , where  $g : \mathcal{Y} \rightarrow \mathbb{R}$  is an unknown utility function that reflects the DM’s preferences. In practice, the DM is often unable to directly specify the closed form of the utility function, so the exact scalar utility value is not directly observable. Instead, the DM can provide feedback, e.g., based on pairwise comparisons (Lin et al., 2022) or, in this work, natural language, to inform about the shape of the utility function  $g$ . Our goal is to identify parameters  $x \in \mathcal{X}$  that maximize satisfaction of the DM, i.e., to solve the composite optimization problem defined as:

$$x^* := \arg \max_{x \in \mathcal{X}} g(f(x)).$$

For example, as illustrated in Figure 1, in hyperparameter tuning for machine learning models,  $x$  would be the collection of hyperparameters such as batch size or the learning rate and  $y$  would be the model’s evaluation metrics such as accuracy, training time, and memory usage. Given a specific set of outcomes  $y$ , the utility  $u$  reflects the DM’s overall satisfaction about the observed outcomes  $y$ .

Note that this is a highly flexible setup that can accommodate a wide variety of problems. When there is no intermediate outcome that can be observed and the preference is directly defined over the parameter space  $\mathcal{X}$ , it is equivalent to letting  $y = f(x) = x$ , reducing this problem to the classic *preferential BO* (PBO) problem (González et al., 2017; Astudillo et al., 2023). When the  $y$  is high-dimensional (e.g., image or text) or involves hard-to-quantify outcomes (e.g., the decision maker’s subjective feelings), the proposed method is still applicable by modeling the composite structure  $(g \circ f)(x)$  directly instead of  $f$  and  $g$  separately and skipping explicit modeling of  $y$  completely.

## 2 RELATED WORKS

**Preferential Bayesian Optimization.** Classic BO methods (Shahriari et al., 2015; Frazier, 2018) combine a probabilistic surrogate model – typically a GP – with an acquisition function that balances exploration and exploitation. *Preference learning* extends this idea to settings where explicit numerical objectives are unavailable and instead only preferential feedback (e.g. in form of pairwise or comparative judgments) can be obtained from a decision maker. The seminal work of Chu & Ghahramani (2005) introduced GP preference learning with a probit likelihood to model noisy pairwise data. González et al. (2017) formalized preferential BO, providing algorithms and guarantees for optimization from pairwise comparisons. Lin et al. (2022) extended this idea with *Bayesian optimization with preference exploration*, where DMs provide feedback on hypothetical outcomes of experiments in order to best learn the DM’s utility function. This decision-theoretic policy, Expected Utility of the Best (EUBO), had been extended to the best-out-of-k selection in Astudillo et al. (2023). Connections to dueling and multi-dueling bandits, made for instance by Sui et al. (2017), highlight links between kernelized BO and bandit feedback. Together, these methods demonstrate principled approaches to preference-guided optimization, but rely on rigidly structured feedback

(pairwise or best-of- $k$  comparisons), limiting their ability to exploit natural language feedback that is both more expressive and natural for human decision makers.

**Optimization with LLMs.** Modern LLMs have been shown to be effective optimizers when used with in-context demonstrations (Yang et al., 2023). Agarwal et al. (2025) explore dynamically updating LLM prompts with past solutions including uncertainty estimates. Other works (Ramos et al., 2023; Cai et al., 2025) explore the synergy of LLMs and BO based on GP surrogates. Liu et al. (2024) investigate using LLMs for warm starting BO, surrogate hints, and proposal generation. Kristiadi et al. (2024) treat LLMs as a fixed feature extractors and train standard BO surrogates on the embeddings. Austin et al. (2024) and Handa et al. (2024) use an LLM to conduct preference elicitation to perform BO over a discrete set of candidates based on binary feedback maintaining an explicit parametric posterior of each individual candidate point. While these studies show promising results, some of their designs may present challenges for scalability, data requirements, or coherent integration between LLM and BO components, as the roles of the LLM and surrogates are often used in parallel rather than in an inter-dependent manner within a unified framework.

In black-box optimization beyond BO, Alamdari et al. (2024) use LLMs to generate synthetic prior feedback to cold-start bandits. Ye et al. (2025) propose LOLA, integrating LLM predictions with UCB bandits for online content experiments. Kobalczyk et al. (2025) improve the LLM’s abilities to ask good clarifying questions by leveraging concepts of Bayesian Experimental Design (BED) – shifting the conversation to maximize information gain in interaction with the decision maker.

**LLM as Few-Shot Learners and Bayesian Predictors.** The foundation of much of this work’s natural language aspect lies in the ICL abilities of LLMs. The seminal GPT-3 paper (Brown et al., 2020) showed that LLMs can act as effective few-shot learners. Multiple subsequent works take a Bayesian view on ICL. Panwar et al. (2023) investigate conditions where transformers approximate Bayes-optimal predictors, while Falck et al. (2024) test the martingale property and find systematic deviations suggesting ICL is not strictly Bayesian. Müller et al. (2023) argue that ICL reflects approximate Bayesian posteriors learned during transformer pretraining. Requeima et al. (2024) propose extracting predictive distributions directly from LLMs, making the Bayesian link explicit through natural language conditioning. Zhu & Griffiths (2024) propose an iterated ICL mechanism to approximate the implicit Bayesian priors encoded in LLMs, offering a general-purpose interpretability tool that can inform how these models’ beliefs might be leveraged or mitigated in the downstream optimization tasks such as BO or preference elicitation.

### 3 BACKGROUND

In this section, we familiarize the reader with Pairwise GP models and the EUBO acquisition function which play a key role in preferential BO as well as our framework.

**Pairwise GP.** A standard model for pairwise preference data follows Chu & Ghahramani (2005), where a latent utility function  $g : \mathcal{Y} \rightarrow \mathbb{R}$  is endowed with a GP prior. Given comparison pairs  $\{(y_k, y'_k)\}$  and binary labels  $p_k \in \{0, 1\}$  indicating whether  $y_k$  is preferred over  $y'_k$ , preferences are assumed to arise from utility differences modelled with the probit likelihood:

$$\Pr(p_k = 1 \mid g) = \Phi(g(y_k) - g(y'_k)), \quad (1)$$

where  $\Phi$  is the standard normal CDF, and the likelihood factorizes over comparisons. Because the probit likelihood is non-Gaussian, the posterior over latent utilities is intractable. The Laplace approximation is therefore used to obtain a Gaussian approximation. This enables tractable prediction and acquisition optimization. Further details can be found in Chu & Ghahramani (2005).

**EUBO.** In the setup of Lin et al. (2022), the goal is to learn the latent utility function  $g$  via obtaining pairwise preference feedback. The Expected Utility of Best Option (EUBO) (Astudillo et al., 2023) acquisition function guides preference-exploration by selecting comparisons expected to yield the greatest increase in the utility of the best option. Concretely, consider two candidate outcome vectors  $(y_1, y_2)$ . The EUBO score for the pair  $(y_1, y_2)$  is defined as:

$$\text{EUBO}(y_1, y_2) = \mathbb{E}[\max\{g(y_1), g(y_2)\}]. \quad (2)$$

Under a Gaussian posterior, the above expression can be computed analytically by standard results for a truncated normal differences. By selecting the pair with highest EUBO, one actively queries in a way that, in expectation, improves knowledge about top-utility options, accelerating convergence towards high-utility outcomes while minimizing preference queries.

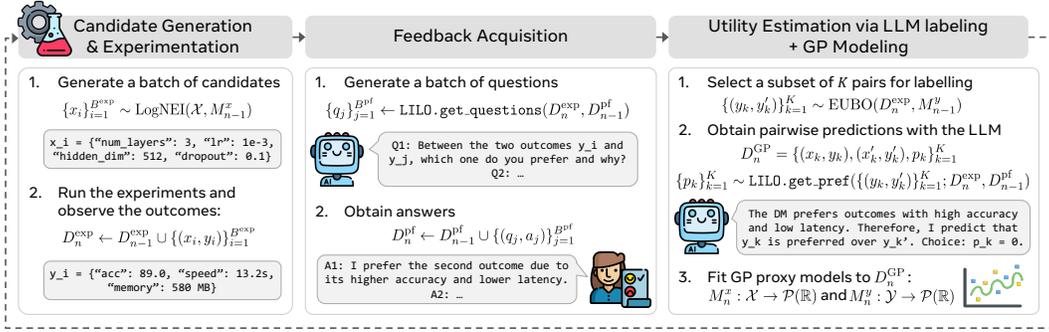


Figure 2: *Overview of LILLO*: Configurations to be evaluated on the black box function are generated with a BO acquisition function leveraging the GP utility model. After observing associated outcomes, the LLM generates a set of questions for the decision maker based on all previously observed outcomes and user feedback. Responses are added to the LLM’s conversation history, based on which the LLM is asked to provide preferential feedback on a set of pairwise comparisons. This feedback is then used to update the GP utility model, after which the iteration continues.

## 4 METHOD

### 4.1 LILLO: LANGUAGE-IN-THE-LOOP OPTIMIZATION

The key idea behind LILLO is to use an iterative preferential BO algorithm with an explicit GP-based utility model to steer the optimization in a principled fashion while harnessing the power of LLMs to extract preferences from the decision maker in the form of natural language feedback and agentically devise questions during such feedback elicitation. Figure 2 provides an overview of the optimization loop after initialization. We describe the core steps of LILLO below. Algorithm 1 presents pseudocode, with subroutines described in Appendix A.1. The exact formats of the prompts used can be found in Appendix A.2.

Our BO procedure consists of  $T$  sequential trials indexed by  $n \in \{1, 2, \dots, T\}$ . We define by  $D_n^{\text{exp}} = \{(x_i, y_i)\}$  the accumulated *experimental dataset* consisting of input–output pairs observed up to trial  $n$ . Additionally,  $D_n^{\text{pf}} = \{(q_j, a_j)\}$  denotes the *preference feedback dataset* that contains a history of the DM’s natural language feedback in the form of answers  $a_j$  to LLM-generated queries  $q_j$ . From these datasets, in each trial, we fit two proxy GP models (for details see below):

- $M_{n-1}^x: \mathcal{X} \rightarrow \mathcal{P}(\mathbb{R})$ , approximating the composite function  $g \circ f: \mathcal{X} \rightarrow \mathbb{R}$ ,
- $M_{n-1}^y: \mathcal{Y} \rightarrow \mathcal{P}(\mathbb{R})$ , approximating the DM’s utility function  $g: \mathcal{Y} \rightarrow \mathbb{R}$ .

**The entry point.** Before entering the main optimization loop, the algorithm begins by eliciting the DM’s high-level optimization goals. At this stage, no experimental outcomes exist ( $D_0^{\text{exp}} = \emptyset$ ), so the LILLO agent generates  $B^{\text{pf}}$  general questions for the DM. After obtaining the DM’s answers, the preference feedback dataset is instantiated as  $D_0^{\text{pf}} = \{(q_j, a_j)\}_{j=1}^{B^{\text{pf}}}$ .

Next, LILLO proceeds to the main optimization loop, in which each trial consists of three stages: 1) Candidate Generation & Experimentation, 2) Feedback Acquisition, and 3) Utility Estimation via LLM Labeling + GP Modeling.

**Candidate Generation & Experimentation.** At  $n = 1$ , when no proxy model has been fit yet, we generate the first batch of candidates uniformly at random. For  $n > 1$ , using the current input-space model  $M_{n-1}^x$ , we optimize an acquisition function over the search space  $\mathcal{X}$  to select a batch of  $B^{\text{exp}}$  candidate inputs  $\{x_i\}_{i=1}^{B^{\text{exp}}}$ . Our acquisition function of choice (although alternative options are also viable, see Appendix C.8) is the batch version of (*Log Noisy Expected Improvement* (Ament et al., 2023)), which is well-suited here because  $M_{n-1}^x$  is trained on noisy, LLM-derived utility estimates rather than evaluations of the true utility function. Each selected input  $x_i$  is then evaluated on the (black-box) function  $f$ , producing outcomes  $y_i = f(x_i)$ . With this, the experimental dataset becomes  $D_n^{\text{exp}} = D_{n-1}^{\text{exp}} \cup \{(x_i, y_i)\}_{i=1}^{B^{\text{exp}}}$ .

**Feedback Acquisition.** After obtaining new experimental outcomes, the LILLO agent generates  $B^{\text{pf}}$  questions for the domain expert to answer. These queries can include both: high-level questions regarding overall optimization goals, and questions about specific outcomes observed

**Algorithm 1:** LILLO: Language-in-the-loop Optimization**Require:** Max number of iterations  $T$ , experiment batch size  $B^{\text{exp}}$ , feedback batch size  $B^{\text{pf}}$ . **Optional:**Prior knowledge prompt  $P_{\text{prior}}$  $D_0^{\text{exp}} \leftarrow \emptyset$ ; $\{q_j\}_{j=1}^{B^{\text{pf}}} \leftarrow \text{LILLO.get\_init\_questions}()$ ; $\{a_j\} \leftarrow \text{DM.get\_answers}(\{q_j\}_{j=1}^{B^{\text{pf}}})$ ; $D_0^{\text{pf}} \leftarrow \{(q_i, a_i)\}_{i=1}^{B^{\text{pf}}}$ ;**for**  $n = 1$  **to**  $T$  **do**  **if**  $n = 1$  **then**    **if**  $P_{\text{prior}} \neq \emptyset$  **then**       $\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{LILLO.get\_init\_x}(D_0^{\text{pf}}, P_{\text{prior}})$ ;    **else**       $\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{Uniform}(\mathcal{X})$ ;  **else**     $\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{LogNEI}(\mathcal{X}; M_{n-1}^x)$ ;     $D_n^{\text{exp}} \leftarrow D_{n-1}^{\text{exp}} \cup \{(x_i, y_i) : y_i = f(x_i)\}_{i=1}^{B^{\text{exp}}}$ ;     $\{q_j\}_{j=1}^{B^{\text{pf}}} \leftarrow \text{LILLO.get\_questions}(D_n^{\text{exp}}, D_{n-1}^{\text{pf}})$ ;     $\{a_j\}_{j=1}^{B^{\text{pf}}} \leftarrow \text{DM.get\_answers}(\{q_j\}_{j=1}^{B^{\text{pf}}})$ ;     $D_n^{\text{pf}} \leftarrow D_{n-1}^{\text{pf}} \cup \{(q_j, a_j)\}_{j=1}^{B^{\text{pf}}}$ ;     $M_n^x, M_n^y \leftarrow \text{fit\_proxy\_models}(D_n^{\text{exp}}, D_n^{\text{pf}}, M_{n-1}^y)$      (See Algorithm 3 for details);

(which ones are preferred, how to improve them, etc.). The prompt for question generation contains all experimental outcomes and human feedback messages obtained. The LLM produces a set of  $B^{\text{pf}}$  questions  $q_j$ , and the DM provides corresponding answers  $a_j$ . The preference feedback dataset becomes  $D_n^{\text{pf}} = D_{n-1}^{\text{pf}} \cup \{(q_j, a_j)\}_{j=1}^{B^{\text{pf}}}$ .

**Utility Estimation via LLM Labeling + GP Modeling.** To convert the natural language feedback of the DM into a usable optimization signal, we construct a dataset of  $K$  pairwise preferences between outcomes in  $D_n^{\text{exp}}$ . The LILLO agent is tasked with labeling these comparisons based on the DM’s preference feedback in  $D_n^{\text{pf}}$ . We first select  $K$  informative outcome pairs  $\{(y_k, y'_k)\}$ —using the EUBO acquisition function (Astudillo et al., 2023), which we approximate based on  $M_{n-1}^y$ . For each pair  $(y_k, y'_k)$ , the LLM determines which outcome is more aligned with the DM’s preferences, producing a label  $p_k \in \{0, 1\}$  indicating whether or not  $y_k$  is preferred over  $y'_k$ . This results in a dataset  $D_n^{\text{GP}} = \{(x_k, y_k), (x'_k, y'_k), p_k\}_{k=1}^K$ , which we use to re-fit the pairwise GP (Chu & Ghahramani, 2005) proxy models: the input-space utility model  $M_n^x : \mathcal{X} \rightarrow \mathcal{P}(\mathbb{R})$  and the outcome-space utility model  $M_n^y : \mathcal{Y} \rightarrow \mathcal{P}(\mathbb{R})$ . These updated models will be used in the next trial to guide candidate generation and the LLM labeling process.

#### 4.2 INCORPORATING PRIOR KNOWLEDGE WITH LILLO

In addition to providing feedback on observed outcomes, decision makers with domain expertise often possess strong prior beliefs about the optimization problem at hand. These priors may include expectations about optimal parameters  $x$ , or insights into how specific parameters influence outcomes  $y$ —that is, information about the underlying mapping  $f$ . Conventional BO approaches, however, make it challenging to incorporate such priors, as this typically requires interpreting the decision maker’s knowledge by a human and manually encoding it into the model (e.g., through specialized kernels, custom mean functions, or carefully designed priors over the surrogate model’s parameters). This process can be both time-consuming and error-prone, particularly when the expert’s knowledge is qualitative and difficult to formalize mathematically. A natural language interface offers a more intuitive way for decision makers to express their prior knowledge.

In scenarios where such prior information is available, we introduce a modification to LILLO that enables warm-starting the optimization process. Inspired by Liu et al. (2024), who demonstrate that LLMs can serve as effective candidate samplers when contextual knowledge is present, we propose replacing the uniform sampling at  $n = 1$  with LLM-based sampling. Specifically, given

## Example Conversation of LILO with the DM agent

270  
271  
272 - DM: My goal is to keep all metrics within my thermal comfort  
273 preferences.  
274 - LILO: Can you provide more specific thresholds for each metric that  
275 you consider to be within your thermal comfort preferences?  
276 - DM: To be honest, I prefer it when the air doesn't feel drafty at my  
277 neck, and I don't feel too much warmth coming from the ceiling. I  
278 also like it when there's not a big difference in temperature between  
279 my head and ankles. As for the floor, I find it comfortable when it's  
280 around 22-24 degrees Celsius. If it's too cold or too hot, my feet  
281 feel uncomfortable, and that affects my overall comfort.  
282 - LILO: How do you prioritize these metrics relative to one another?  
283 For example, is minimizing PPD more important to you than minimizing  
284 DR, or are they equally important?  
285 - DM: For me, it's really important that I don't feel drafty or too  
286 warm from the ceiling. If I had to choose, I'd say minimizing those  
287 feelings is a bit more important than the other factors. But, at the  
288 same time, I also value not having a big temperature difference  
289 between my head and ankles. It's all about finding a balance, I  
290 suppose. If one thing is off, it can really affect how I feel overall.  
291 [...]

291 Example 1: Example fragments of feedback elicitation conversation of LILO with the DM in one of  
292 the test problem (Thermal Comfort Type A). LILO is able to elicit and incorporate such qualitative  
293 feedback about optimization effectively as shown in our experiments in Section 5.

294  $D_0^{\text{pf}}$  and additional prior information represented in textual form as  $P_{\text{prior}}$ , we generate  $B^{\text{pf}}$  candidate  
295 points  $x_i$  via LLM prompting (see Prompt 6, Appendix A.2). Incorporation of prior knowledge is  
296 optional in LILO; in our experiments in Section 5.3.3 we demonstrate how leveraging a good prior  
297 can significantly enhance optimization performance.

## 299 5 EXPERIMENTS

### 300 5.1 BENCHMARKING SETUP

302 **Test problems.** We evaluate LILO on synthetic and real-world outcome functions  $f$ , subject to  
303 various utility functions  $g$ . In all our test problems we let  $\mathcal{X}$  be a hypercube inside  $\mathbb{R}^d$  and  $\mathcal{Y} \subseteq \mathbb{R}^k$ .  
304 Our main benchmark considers four outcome functions (one fully synthetic, and three simulating  
305 real-world outcomes):

- 306 • *DTLZ2* (Deb et al. (2002),  $d = 4, k = 8$ ) is a synthetic outcome function commonly used as a  
307 benchmark for multi-objective optimization algorithms.
- 308 • *Vehicle Safety* (Tanabe & Ishibuchi (2020),  $d = 5, k = 3$ ), a simulator of a vehicle's mass and  
309 two safety-defining outcomes. The inputs are the thickness of the frontal structure of a car,  
310 which affect the vehicle's mass and safety.
- 311 • *Car Cab Design* (Liao et al. (2008),  $d = 7, k = 9$ ), a simulator of a side-impact car crash test.  
312 The inputs measure the thickness of a car's structure, which influence the vehicle's mass, the  
313 physical impact on the passenger, and the physical impact on the car.
- 314 • *Thermal Comfort* (Fanger (1970); ISO7730 (2005),  $d = 8, k = 5$ ) models perceptible thermal  
315 conditions and human dissatisfaction levels based on a set of environmental parameters that  
316 need to be optimized.

317 We consider several utility functions on the outcomes of these test problems:

- 318 • *piecewise linear*, modeling diminishing returns when outcomes reach their desired thresholds.
- 319 • *beta products*, describing bounded monotonic utilities that smoothly vary between increasing  
320 and decreasing marginal gains with respect to their inputs.
- 321 • *L1 distance*, measuring the L1 distance of outcomes from a pre-defined optimum point.
- 322 • For the Thermal Comfort problem, we consider two personas with different preferences:  
323 *Type A*) an office worker with light clothing and a moderate tolerance for different conditions;  
*Type B*) a summer athlete, wearing light sport kit, with a low tolerance for adverse conditions.

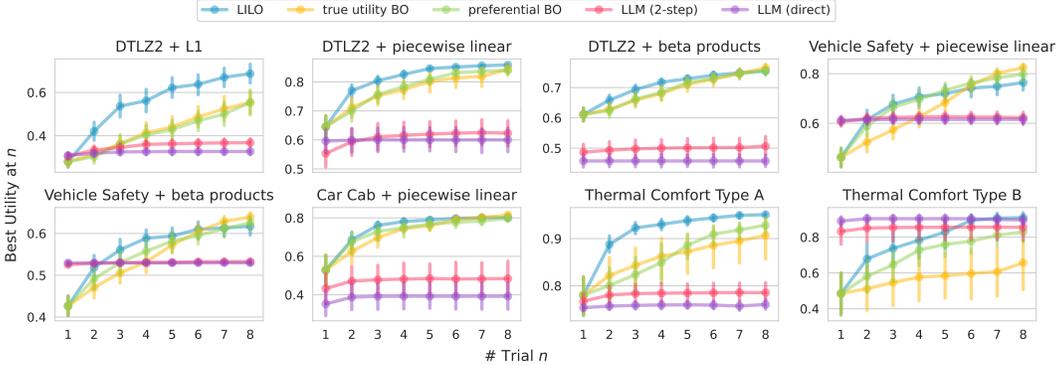


Figure 3: Maximum value of the ground-truth utility achieved after  $n$  iterations. Error bars are 95% confidence intervals of the mean max value across 30 replications.

All test problems and outcome/utility combinations are described in detail in Appendix B. All utility functions are designed to take values in  $[0, 1]$ .

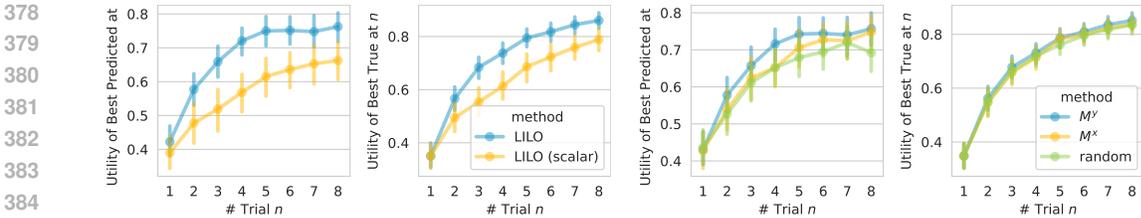
**Baselines.** We compare LILLO against the following baselines (full details for all baselines are provided in Appendix B): **► true utility BO**: a GP-based BO that directly observes ground-truth utilities  $u_i = g(y_i)$  for each outcome  $y_i$ . Since utilities are never directly accessible in practice, this baseline serves as an oracle “gold standard”. **► preferential BO**: a setup with binary pairwise comparison feedback instead of natural language feedback. Pairwise comparisons between two outcomes  $y, y'$  are derived from the ground-truth utility differences  $g(y) - g(y')$ . Similarly to true utility BO, this can be seen as a “gold standard” baseline, as in practice, pairwise comparison feedback is noisy due to inconsistencies in human decision-making. **► LLM (2-step)**: an ablation of our method in which the GP model and acquisition function are replaced by LLM-based candidate generation. Specifically, in each iteration, the LLM first estimates utilities for previously observed outcomes  $D_n^{\text{exp}}$  (via pairwise comparisons) and then selects the next candidate batch through in-context learning. The prompt for candidate generation contains the dataset  $D_n^{\text{exp}}$  with the utility estimates based on the  $M_n^y$  model and the history of conversation with the DM from  $D_n^{\text{pf}}$ . This baseline resembles the LLAMBO method of Liu et al. (2024). The key difference lies in the fact that in the original setup of LLAMBO, the ground-truth utilities of individual candidates  $x$  are directly observable. Here, they must be estimated based on the natural language feedback. **► LLM (direct)**: a fully end-to-end LLM approach, where new candidates are proposed directly based on the observed  $(D_n^{\text{pf}}, D_n^{\text{exp}})$  omitting the intermediate utility estimation step.

**Simulation environment.** We run the BO loop for  $T = 8$  iterations, setting  $B^{\text{exp}} = d$  (the dimension of the search space). For the main experiments we use a feedback batch size of  $B^{\text{pf}} = 2$  (see Appendix C.2 for an ablation across varying values of  $B^{\text{pf}}$ ). The true utility BO baseline receives true utility observations for  $B^{\text{pf}}$  outcomes selected from  $D_n^{\text{exp}}$ . The preferential BO baseline receives pairwise comparison feedback for  $B^{\text{pf}}$  outcome pairs selected from  $D_n^{\text{exp}}$ . In both methods, we use the EUBO acquisition function to select the data points on which the quantitative feedback is provided (see Appendix C.5 for an ablation on the choice of the feedback acquisition function). For methods involving natural language feedback, answers to questions posed by the LLM agent are simulated with another LLM containing a textual description of the ground-truth utility function in the prompt (see Appendix B.3 for details). In the main paper, both the DM and LILLO agents are instantiated with the Llama-3.3-70b-instruct language model. Results for other choices of LLM models are presented in Appendix C.1.

## 5.2 KEY RESULTS

We first present the results on a set of 8 environments, where no prior knowledge about the problem is provided ( $P_{\text{prior}} = \emptyset$ ), except for the variable names in the Vehicle Safety, Car Cab Design, and Thermal Comfort environments. For fairness of comparison with the quantitative baselines, we use LILLO with random initialization, as initialization leveraging the LLM’s world knowledge may give a significant performance boost as we later show in Section 5.3.3

Figure 3 shows the maximum ground-truth utility achieved after  $n$  trials of experimentation and feedback collection, that is,  $\max_{x \in D_n^{\text{exp}}} g(f(x))$ . We observe that LILLO consistently outperforms the baselines, especially early on during the optimization. In some environments, where outcomes



(a) Performance of LILO with pairwise vs. scalar utility estimation. (b) Comparison of methods for LLM pairwise preference labeling.

Figure 4: Comparison of LILO variants. Average results over three DTLZ2 environments (30 replications per environment, values min-max standardized within an environment before aggregation).

correspond to semantically meaningful quantities, the two LLM baselines show good zero-shot performance, but in general, they fail to improve meaningfully over the course of the optimization. This observation aligns with the in-context learning literature on its diminishing return over number of examples provided (Brown et al., 2020; Zhao et al., 2024; Yin et al., 2024). Remarkably, LILO also substantially outperforms the true utility and preferential BO baselines – this is due to the fact that **natural language feedback can convey much richer information about the overall DM’s preferences than only localized, point-specific feedback**. The DM may provide auxiliary feedback not only on the performance of specific outcomes, but also on the overall shape of their underlying utility function (e.g., the directionality of the utility function with respect to different metrics, their relative importance, etc.). See Example 1 and Appendix D for example conversations from the benchmark. With increasing number of experimental trials, the advantage of natural language feedback diminishes and scalar or pairwise utility feedback baselines catch up.

### 5.3 ADDITIONAL STUDIES

In this section, we present additional studies providing insights into the key elements of LILO.

#### 5.3.1 PAIRWISE COMPARISONS VS. DIRECT UTILITY ESTIMATION.

The default utility estimation step in LILO relies on LLM-generated pairwise comparisons. As an alternative, we consider directly prompting the LLM to output scalar utility values. In this variant, instead of labeling pairwise preferences, the LLM produces scalar predictions  $\hat{u}_i \in [0, 1]$  for each  $(x_i, y_i) \in D_n^{\text{exp}}$ , resulting in a dataset  $D_n^{\text{GP}} = \{(x_i, y_i, \hat{u}_i)\}$ . This dataset is then used to fit proxy models  $M_n^x$  and  $M_n^y$  as standard GPs. All other parts of the pipeline remain unchanged. The prompt used for this method and the modified algorithm are presented in Appendix A.

**Results.** To compare the two approaches, we conducted an ablation study on the three DTLZ2 environments with varying utility functions. Figure 4a reports average results over 30 replications per environment. The right pane shows the maximum ground-truth utility achieved until the  $n$ -th iteration. On the left, we report the ground-truth utility of the best candidate according to the proxy model  $M_n^y$ , i.e.  $g(\hat{x}_n^*)$ , where  $\hat{x}_n^* = \arg \max_{x \in D_n^{\text{exp}}} M_n^y(f(x))$ . In practice, it may be preferable or even necessary to use the utility model to do this “best point selection”; especially with many observations, it will be impractical for the DM to compare a large slate of options at once – in fact, they may not be able to do this well due to mental overload. The results demonstrate that **pairwise comparisons provide more reliable utility estimates than direct scalar predictions, leading to improved optimization performance**. This observation for LLMs mirrors findings in human preference elicitation, where pairwise comparisons have been shown to yield more consistent and accurate judgments than absolute scalar ratings (Phelps et al., 2015; Hoeijmakers et al., 2024).

#### 5.3.2 THE CHOICE OF PAIRS FOR LLM LABELING.

During the utility estimation step of LILO, we prompt the agent to label a subset of  $K$  pairwise comparisons between outcomes from  $D_n^{\text{exp}}$ ,  $\{(y_k, y'_k)\}_{k=1}^K$ . This set of outcomes is selected by computing the value of the EUBO acquisition functions with respect to the  $M_{n-1}^y$  model, and selecting the top  $K$  pairs with the highest EUBO values. In our experiments, we use  $K = 64$ . In this section, we ablate this step by comparing it against random pair selection, and against a method that uses the  $M_{n-1}^x$  instead of the  $M_{n-1}^y$  model. For the latter, the EUBO is computed on pairs of inputs  $(x_k, x'_k)$  and the top  $K$  pairs of corresponding outcomes  $(y_k, y'_k)$  are passed to the LLM for labeling.

**Results.** Figure 4b shows the results obtained on the three DTLZ2 environments. We observe that the use of the  $M_{n-1}^y$  model for pair selection with respect to EUBO improves the predictive performance of the outcome-space proxy in identifying the best candidate (left pane). This improvement, however, does not directly translate to significant differences in terms of the optimization performance measured with respect to the ground-truth utility function (right pane). Thus, if the outcome space  $\mathcal{Y}$  is high dimensional (e.g., text or image), and the mapping  $\mathcal{Y} \rightarrow \mathbb{R}$  hard to model, fitting the outcome space GP models may be omitted and we may simply select points for labeling using  $M_{n-1}^x$  or even randomly, simplifying the pipeline.

### 5.3.3 INCORPORATING PRIOR KNOWLEDGE.

As described in Section 4.2, LILLO can also incorporate domain priors to boost optimization performance. In the following, we demonstrate this empirically, considering three types of prior messages  $P_{\text{prior}}$ : **point**: A message providing a sample candidate with high expected utility: “Based on my experience, the following inputs should bring good results:  $\{x\}$ ”. The promising candidate is generated by sampling uniformly at random  $N$  candidates, computing their ground-truth utilities, and randomly sampling a single point from the top  $q\%$  of data points. **area**: A message providing expected bounds of good candidates: “Based on my experience, inputs within these ranges should bring good results:  $\{\text{bounds}\}$ ”. The bounds are computed by sampling uniformly at random  $N$  candidates, computing their ground-truth utilities, and taking the 0.25 and 0.75 quantiles of the top  $q\%$  of data points, as the lower and upper bounds, respectively. **domain**: A message contextualizing the input parameters and the outputs. This is applicable only to semantically-meaningful environments: Vehicle Safety, Car Cab Design and Thermal Comfort. Exact forms of these messages are provided in Appendix B.4. For the DTLZ2 outcome function, we apply the point and area priors. For point knowledge we use  $q = 10$  and for area knowledge we use  $q = 5$  with  $N = 5000$  in both cases. For Vehicle Safety, Car Cab Design and Thermal Comfort we apply the domain priors.

**Results.** Figure 5 presents a comparison of LILLO’s performance with and without prior knowledge integrated into the optimization pipeline. As expected, **incorporating prior knowledge through LLM-based initialization substantially improves the starting point, resulting in better overall optimization performance.** For a more detailed breakdown of the results, see Appendix C.4. We also note that point and area knowledge types depend on the accuracy of externally provided information about  $(g \circ f)$  and, in principle, could be incorporated into conventional BO pipelines using model-based approaches. However, LILLO’s success with domain priors relies on the contextual understanding provided by the LLM’s pre-training, which would be challenging to replicate with standard model-based methods. This ability to incorporate various types of prior knowledge in a unified fashion further underscores LILLO’s flexibility.

### 5.3.4 ADDITIONAL EXPERIMENTAL RESULTS

Beyond the results described above, we further conduct a series of additional experiments providing further insights. These include ablation studies on the choice of LLM models (Appendix C.1), the choice of the batch size of the feedback queries (Appendix C.2, C.3), a qualitative visualization of the fitted utility functions (Appendix C.6), and a performance analysis of LILLO on longer optimization horizons (Appendix C.7).

## 6 DISCUSSION AND CONCLUSION

We presented LILLO, a novel algorithm for language-in-the-loop optimization that enables decision makers to perform interactive black-box optimization by communicating their goals and preferences in natural language. This addresses a key challenge in settings where optimization objectives are not explicitly available, but where classic preference learning approaches (e.g. using pairwise feedback) are not sample-efficient enough. LILLO demonstrates compelling performance in our benchmarks, and has been successfully applied in multiple real-world optimization campaigns involving complex tradeoffs between multiple metrics in online A/B testing at a large internet company.

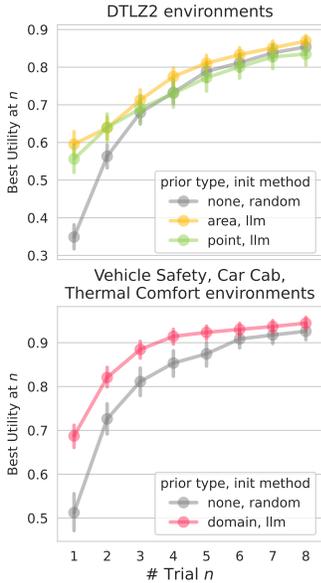


Figure 5: LILLO with and without prior knowledge.

The reason of LILLO’s superior performance is multi-faceted. Natural language feedback is far more information-rich than other common forms of preference elicitation such as pairwise comparison as demonstrated by the experimental results where LILLO achieves very competitive optimization results with just a few feedback queries. On the other hand, LLM-based optimization methods are fit to handle natural language feedback through ICL yet, their performance can plateau rather quickly. This phenomenon is observed throughout our experimentation where LLM-based methods’ optimization traces start relatively high in the first iteration and then stop progressing meaningfully throughout the rest of the experiment. LILLO is able to overcome such stagnation by keeping the valid probabilistic modeling and principled acquisition mechanism of BO while leveraging LLMs for its human-friendly natural language interface, agentic feedback acquisition, and domain knowledge (from both the DM as well as the LLM).

While our results are strong, several promising directions remain. The fidelity of the LLM estimator is ultimately limited by the model’s ability to interpret nuanced feedback. Our benchmarks in Appendix C.1 show robustness across models, with even moderately sized ones yielding meaningful gains. Future work could explore how scaling, fine-tuning, or architectural advances further enhance optimization quality. Hybrid approaches that combine unstructured natural language with structured quantitative feedback are another possibility to mitigate the limits of ICL in the utility estimation step of LILLO.

Moreover, LLM-based estimators may occasionally encode preferences or priors too strongly, risking subtle bias toward the model’s own world knowledge rather than the intended optimization target. In our experiments, we found this effect to be manageable by ablating over a variety of utility functions and test problem combinations. Yet, future work could explore calibration strategies or adaptive weighting schemes that better balance the advantages of informative model priors with fidelity of the provided feedback.

Although we validated LILLO extensively using both synthetic and real optimization problems, understanding its behavior across a broader range of tasks and feedback sources remains an exciting area. Evaluating performance with different natural language agents, including but not limited to human decision makers, can shed light on the robustness and generalization of the proposed method.

Looking forward, these directions open opportunities for advancing language-in-the-loop optimization toward even more adaptive, trustworthy, and broadly applicable optimization systems across diverse domains.

## REFERENCES

- Dhruv Agarwal, Manoj Ghuhana, Rajarshi (Raj) Das, Sandesh Swamy, Sopan Khosla, and Rashmi Gangadharaiyah. Searching for optimal solutions with llms via bayesian optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Parand A Alamdari, Yanshuai Cao, and Kevin H Wilson. Jump starting bandits with llm-generated prior knowledge. *arXiv preprint arXiv:2406.19317*, 2024.
- Sebastian Ament, Sam Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=1vyAG6j9PE>.
- ASHRAE55. *Thermal environmental conditions for human occupancy*. ASHRAE, 2020.
- 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 (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 1093–1114. PMLR, 25–27 Apr 2023. URL <https://proceedings.mlr.press/v206/astudillo23a.html>.
- David Austin, Anton Korikov, Armin Toroghi, and Scott Sanner. Bayesian optimization with llm-based acquisition functions for natural language preference elicitation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pp. 74–83, 2024.

- 540 Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, An-  
541 drew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo  
542 Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL  
543 <http://arxiv.org/abs/1910.06403>.  
544
- 545 Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive  
546 cost functions, with application to active user modeling and hierarchical reinforcement learning.  
547 *arXiv preprint arXiv:1012.2599*, 2010.
- 548 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,  
549 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are  
550 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.  
551
- 552 Chengkun Cai, Haoliang Liu, Xu Zhao, Zhongyu Jiang, Tianfang Zhang, Zongkai Wu, John Lee,  
553 Jenq-Neng Hwang, and Lei Li. Bayesian optimization for controlled image editing via llms. *arXiv*  
554 *preprint arXiv:2502.18116*, 2025.
- 555 Wei Chu and Zoubin Ghahramani. Preference learning with gaussian processes. In Luc De Raedt  
556 and Stefan Wrobel (eds.), *Machine Learning, Proceedings of the Twenty-Second International*  
557 *Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International*  
558 *Conference Proceeding Series*, pp. 137–144. ACM, 2005. doi: 10.1145/1102351.1102369. URL  
559 <https://doi.org/10.1145/1102351.1102369>.  
560
- 561 K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems.  
562 In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02*, 2002.
- 563 Fabian Falck, Ziyu Wang, and Chris Holmes. Is in-context learning in large language models  
564 bayesian? a martingale perspective. *arXiv preprint arXiv:2406.00793*, 2024.  
565
- 566 Poul O Fanger. *Thermal comfort: analysis and applications in environmental engineering*. Danish  
567 Technical Press, 1970.  
568
- 569 Qing Feng, Zhiyuan Jerry Lin, Yujia Zhang, Benjamin Letham, Jelena Markovic-Voronov, Ryan-  
570 Rhys Griffiths, Peter I Frazier, and Eytan Bakshy. Bayesian optimization of high-dimensional  
571 outputs with human feedback. In *NeurIPS 2024 Workshop on Bayesian Decision-making and*  
572 *Uncertainty*, 2024.
- 573 Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.  
574
- 575 Javier González, Zhenwen Dai, Andreas Damianou, and Neil D Lawrence. Preferential bayesian  
576 optimization. In *International Conference on Machine Learning*, pp. 1282–1291. PMLR, 2017.  
577
- 578 Kunal Handa, Yarin Gal, Ellie Pavlick, Noah Goodman, Jacob Andreas, Alex Tamkin, and Be-  
579 linda Z. Li. Bayesian preference elicitation with language models, 2024. URL <https://arxiv.org/abs/2403.05534>.  
580
- 581 E J I Hoeijmakers, B Martens, B M F Hendriks, C Muhl, R L Miclea, W H Backes, J E Wildberger,  
582 F M Zijta, H A Gietema, P J Nelemans, and C R L P N Jeukens. How subjective CT image quality  
583 assessment becomes surprisingly reliable: pairwise comparisons instead of likert scale. *European*  
584 *radiology*, 34(7):4494–4503, 2024.  
585
- 586 ISO7730. *Ergonomics of the thermal environment-Analytical determination and interpretation of*  
587 *thermal comfort using calculation of the PMV and PPD indices and local thermal comfort crite-*  
588 *ria*. ISO, 2005.
- 589 Katarzyna Kobalczyk, Nicolas Astorga, Tennison Liu, and Mihaela van der Schaar. Active task  
590 disambiguation with llms. *arXiv preprint arXiv:2502.04485*, 2025.  
591
- 592 Augustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and  
593 Geoff Pleiss. A sober look at llms for material discovery: Are they actually good for bayesian  
optimization over molecules? *arXiv preprint arXiv:2402.05015*, 2024.

- 594 Xingtao Liao, Qing Li, Xujing Yang, Weigang Zhang, and Wei Li. Multiobjective optimization for  
595 crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary*  
596 *Optimization*, 35(6):561–569, Jun 2008. ISSN 1615-1488. doi: 10.1007/s00158-007-0163-x.  
597 URL <https://doi.org/10.1007/s00158-007-0163-x>.  
598
- 599 Zhiyuan Jerry Lin, Raul Astudillo, Peter Frazier, and Eytan Bakshy. Preference exploration for  
600 efficient bayesian optimization with multiple outcomes. In *International Conference on Artificial*  
601 *Intelligence and Statistics*, pp. 4235–4258. PMLR, 2022.
- 602 Tension Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models  
603 to enhance bayesian optimization. *arXiv preprint arXiv:2402.03921*, 2024.
- 604 Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. Pfns4bo: In-context learning  
605 for bayesian optimization. In *International Conference on Machine Learning*, pp. 25444–25470.  
606 PMLR, 2023.
- 607 Madhur Panwar, Kabir Ahuja, and Navin Goyal. In-context learning through the Bayesian prism.  
608 *arXiv preprint arXiv:2306.04891*, 2023.
- 609 Andrew S Phelps, David M Naeger, Jesse L Courtier, Jack W Lambert, Peter A Marcovici, Javier E  
610 Villanueva-Meyer, and John D MacKenzie. Pairwise comparison versus likert scale for biomed-  
611 ical image assessment. *AJR Am. J. Roentgenol.*, 204(1):8–14, January 2015.
- 612 Mayk Caldas Ramos, Shane S Michtavy, Marc D Porosoff, and Andrew D White. Bayesian opti-  
613 mization of catalysts with in-context learning. *arXiv preprint arXiv:2304.05341*, 2023.
- 614 James Requeima, John Bronskill, Dami Choi, Richard Turner, and David Duvenaud. LLM pro-  
615 cesses: Numerical predictive distributions conditioned on natural language. *Advances in Neural*  
616 *Information Processing Systems*, 37:109609–109671, 2024.
- 617 Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the  
618 human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):  
619 148–175, 2015.
- 620 Yanan Sui, Vincent Zhuang, Joel W Burdick, and Yisong Yue. Multi-dueling bandits with dependent  
621 arms. *arXiv preprint arXiv:1705.00253*, 2017.
- 622 Ryoji Tanabe and Hisao Ishibuchi. An easy-to-use real-world multi-objective optimization problem  
623 suite. *Applied Soft Computing*, 2020.
- 624 Federico Tartarini, Stefano Schiavon, Toby Cheung, and Tyler Hoyt. Cbe thermal comfort tool:  
625 Online tool for thermal comfort calculations and visualizations. *SoftwareX*, 12:100563, 2020.
- 626 Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun  
627 Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning*  
628 *Representations*, 2023.
- 629 Zikun Ye, Hema Yoganarasimhan, and Yufeng Zheng. Lola: Llm-assisted online learning algorithm  
630 for content experiments. *Marketing Science*, 2025.
- 631 Qingyu Yin, Xuzheng He, Luoao Deng, Chak Tou Leong, Fan Wang, Yanzhao Yan, Xiaoyu Shen,  
632 and Qiang Zhang. Deeper insights without updates: The power of in-context learning over fine-  
633 tuning. *arXiv preprint arXiv:2410.04691*, 2024.
- 634 Siyan Zhao, Tung Nguyen, and Aditya Grover. Probing the decision boundaries of in-context learn-  
635 ing in large language models. *Advances in Neural Information Processing Systems*, 37:130408–  
636 130432, 2024.
- 637 Jian-Qiao Zhu and Thomas L Griffiths. Eliciting the priors of large language models using iterated  
638 in-context learning. *arXiv preprint arXiv:2406.01860*, 2024.
- 639  
640  
641  
642  
643  
644  
645  
646  
647

## A DETAILS ON THE METHOD

### A.1 PSEUDO CODE

Algorithm 2 presents a pseudo code of LILLO. The `fit_proxy_models` subroutine is presented in Algorithm 3.

The algorithm for LILLO with scalar utility estimation is identical to LILLO with pairwise preference labeling except for the `fit_proxy_models` subroutine, which is replaced with point-wise utility estimation as a scalar value in  $[0, 1]$ . The exact procedure is presented in Algorithms 4. Prompts used in the `LILLO.xxx` subroutines are presented in section A.2.

---

#### Algorithm 2: LILLO: Language-in-the-Loop Optimization

---

**Input:** Max number of iterations  $T$ , experiment batch size  $B^{\text{exp}}$ , feedback batch size  $B^{\text{pf}}$ .

**Optional:** Prior knowledge prompt  $P_{\text{prior}}$

```

663  $D_0^{\text{exp}} \leftarrow \emptyset;$ 
664 # Ask the DM about their optimization goals
665  $\{q_j\}_{j=1}^{B^{\text{pf}}} \leftarrow \text{LILLO.get\_init\_questions}();$ 
666  $\{a_j\} \leftarrow \text{DM.get\_answers}(\{q_j\}_{j=1}^{B^{\text{pf}}});$ 
667  $D_0^{\text{pf}} \leftarrow \{(q_i, a_i)\}_{i=1}^{B^{\text{pf}}};$ 
668 for  $n = 1$  to  $T$  do
669   # Sample a batch of candidates;
670   if  $n = 1$  then
671     if  $P_{\text{prior}} \neq \emptyset$  then
672        $\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{LILLO.sample\_init\_candidates}(D_0^{\text{pf}}, P_{\text{prior}});$ 
673     else
674        $\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{Uniform}(\mathcal{X});$ 
675     else
676        $\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{LogNEI}(\mathcal{X}; M_{n-1}^x);$ 
677     # Run experiments and update the experimental dataset
678      $D_n^{\text{exp}} \leftarrow D_{n-1}^{\text{exp}} \cup \{(x_i, y_i) : y_i = f(x_i)\}_{i=1}^{B^{\text{exp}}};$ 
679     # Generate a batch of questions
680      $\{q_j\}_{j=1}^{B^{\text{pf}}} \leftarrow \text{LILLO.get\_questions}(D_n^{\text{exp}}, D_{n-1}^{\text{pf}}, \{y_i\}_{i=1}^{2B^{\text{pf}}});$ 
681     # Obtain human answers
682      $\{a_j\}_{j=1}^{B^{\text{pf}}} \leftarrow \text{DM.get\_answers}(\{q_j\}_{j=1}^{B^{\text{pf}}});$ 
683     # Update the preference feedback dataset
684      $D_n^{\text{pf}} \leftarrow D_{n-1}^{\text{pf}} \cup \{(q_j, a_j)\}_{j=1}^{B^{\text{pf}}};$ 
685     # Fit proxy models
686      $M_n^x, M_n^y \leftarrow \text{fit\_proxy\_models}(D_n^{\text{exp}}, D_n^{\text{pf}}, M_{n-1}^y);$ 

```

---

### A.2 LILLO PROMPTS

In all prompts, `experiment_data` is a markdown-formatted table of outcomes from  $D_n^{\text{exp}}$ . `human_feedback` is the series of questions and answer stored in  $D_n^{\text{pf}}$ .

In prompt 2 for question generation, `selected_outcome_indices` is the set of  $2B^{\text{pf}}$  indices of selected outcomes with the feedback acquisition function (EUBO).

In prompts 3, 4, `human_feedback_summary` is a summary of `human_feedback` self-generated by LILLO using the prompt presented in 5. We empirically found that including this self-summarization step brings slight improvements to the resulting estimates of the LLM.

In prompt 3, `pair_str` is a markdown-formatted table with two rows indexed by `option_0` and `option_1`.

**Algorithm 3:** LILO fit\_proxy\_models subroutine

---

```

702 Algorithm 3: LILO fit_proxy_models subroutine
703
704 Input: Experimental dataset  $D^{\text{exp}}$ , Feedback dataset  $D^{\text{pf}}$ , optional preference model  $M^y$ ,
705 number of pairs for labeling  $K = 64$ .
706 # Generate all possible combinations of outcome pairs
707  $N \leftarrow |D^{\text{exp}}|$ 
708  $\{(y_i, y'_i)\}_{i=1}^{\frac{N(N-1)}{2}} \leftarrow \text{get\_outcome\_pair\_combinations}(D^{\text{exp}})$ 
709 if  $M^y = \text{None}$  then
710    $\{(y_k, y'_k)\}_{k=1}^K \leftarrow \text{random\_sample}(\{(y_i, y'_i)\}_{i=1}^{\frac{N(N-1)}{2}}, \text{size} = K)$ 
711 else
712   for  $i = 1$  to  $\frac{N(N-1)}{2}$  do
713      $v_i \leftarrow \text{EUBO}^2((y_i, y'_i), M^y)$ 
714     indices  $\leftarrow \text{argsort}(\{v_i\}_{i=1}^{\frac{N(N-1)}{2}})[-K:]$  // Get indices of top  $K$  EUBO
715     values
716      $\{(y_k, y'_k)\}_{k=1}^K \leftarrow \{(y_i, y'_i) : i \in \text{indices}\}$ 
717   # Label the selected pairs with LILO
718   for  $k = 1$  to  $K$  do
719      $p_k \leftarrow \text{LILO.get\_pairwise\_pref}(y_k, y'_k, D^{\text{exp}}, D^{\text{pf}})$ 
720      $D^{GP,y} \leftarrow \{(y_k, y'_k, p_k)\}_{k=1}^K$ 
721      $D^{GP,x} \leftarrow \{(x_k, x'_k, p_k)\}_{k=1}^K$  //  $x_k, x'_k$  are the inputs corresponding to
722      $y_k, y'_k$ 
723      $M^y \leftarrow \text{fit\_pairwise\_gp}(D^{GP,y})$ 
724      $M^x \leftarrow \text{fit\_pairwise\_gp}(D^{GP,x})$ 
725   return  $M^x, M^y$ 

```

---

**Algorithm 4:** LILO (scalar) fit\_proxy\_models subroutine

---

```

730 Algorithm 4: LILO (scalar) fit_proxy_models subroutine
731
732 Input: Experimental dataset  $D^{\text{exp}}$ , Feedback dataset  $D^{\text{pf}}$ .
733  $N \leftarrow |D^{\text{exp}}|$ 
734  $\{\hat{u}_i\}_{i=1}^N \leftarrow \text{LILO.estimate\_utilities}(\{y_i\}_{i=1}^N; D^{\text{exp}}, D^{\text{pf}})$ 
735  $D^{GP,y} \leftarrow \{(y_i, \hat{u}_i)\}_{i=1}^N$ 
736  $D^{GP,x} \leftarrow \{(x_i, \hat{u}_i)\}_{i=1}^N$   $x_i$ 's are the inputs corresponding to  $y_i$ 's
737  $M^y \leftarrow \text{fit\_simple\_gp}(D^{GP,y})$ 
738  $M^x \leftarrow \text{fit\_simple\_gp}(D^{GP,x})$ 
739 return  $M^x, M^y$ 

```

---

## A.3 GP MODELING AND ACQUISITION FUNCTIONS

In our implementation of the algorithm we rely on the BoTorch Python library (Balandat et al., 2020) to implement the subroutines of GP model fitting, acquisition function evaluation and candidate generation. Specifically, proxy GP models are instances of PairwiseGP or SingleTaskGP classes, with their default settings.

**The Pairwise GP.** To model pairwise preferences, we use a GP with a probit likelihood, following Chu & Ghahramani (2005). Given a set of observed outcomes  $(y_k, y'_k)$  and binary preference labels  $p_k \in \{0, 1\}$ , the latent utility function  $g(\cdot)$  is modeled as a GP. The probability that  $y_k$  is preferred over  $y'_k$  is given by  $\Phi(g(y_k) - g(y'_k))$ , where  $\Phi(\cdot)$  is the cumulative distribution function of the standard normal distribution. Since the probit likelihood is non-Gaussian, we use the Laplace approximation to perform inference. This approach enables principled modeling of uncertainty and efficient learning from pairwise comparisons. For further details, we refer the readers to Chu & Ghahramani (2005).

When querying the LLM to provide pairwise labels or scalar utility estimates, we use multiple LLM samples for more reliable estimates. In our simulations, the number of LLM calls per each labeling

instance is set to 5. We use all LLM predictions (with replicates) to fit the GP proxy models enabling the proxy models to capture the predictive uncertainty of the LLM in its decision.

**(Log) Noisy Expected Improvement.** Our acquisition function of choice for candidate generation is the (Log) Noisy Expected Improvement. We the batched version of this acquisition function, which jointly evaluates the utility of an entire batch via Monte Carlo sampling—rather than selecting points greedily. Specifically, we employ the `qLogNoisyExpectedImprovement` method from BoTorch, which follows the work of Ament et al. (2023).

**EUBO.** For paired outcome selection in LLM labeling step we use the Expected Utility of Best Option acquisition function (Lin et al., 2022). In this step, we use a greedy approach for selecting a batch of  $K$  pairs. For a pair of outcomes  $(y_k, y'_k)$  EUBO has an explicit analytic form. We thus employ the `AnalyticExpectedUtilityOfBestOption` from BoTorch to compute its value for all possible combination of pairs and select the set of  $K$  pairs with the highest utilities (see Algorithm 3).

#### Initial question generation

```
You are an expert in determining whether a human decision maker (DM)
is going to be satisfied with a set of experimental outcomes  $y = \{$ 
 $y\_names\}$ .

## Human feedback messages:
We have also received the following messages from the DM:

{human_feedback}

## Your task:
Given the above your task is to predict the probability of the
decision maker being satisfied with the experimental outcomes.

In order to better understand the decision maker's utility function
you want to ask them about their optimization goals.

Provide a list of questions you would ask the decision maker to better
understand their internal utility model.

Return your final answer a a json file with the following format
containing exactly {n_questions} most important questions:
```json
{{
  "q1" : <question1>,
  ...
  "q{n_questions}" : <question{n_questions}>
}}
```

Prompt 1: The prompt used for question generation in the `LILLO.get_init_questions` subroutine.

810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863

### Question generation

You are an expert in determining whether a human decision maker (DM) is going to be satisfied with a set of experimental outcomes  $y = \{y\_names\}$ .

## Experimental outcomes:

So far, we have obtained the following experimental outcomes:

{experiment\_data}

## Human feedback messages:

We have also received the following messages from the DM:

{human\_feedback}

## Your task:

Given the above your task is to predict pairwise preferences between experimental outcomes.

In order to better understand the decision maker's utility function you want to ask them about their optimization goals or for feedback regarding specific experimental outcomes.

First, analyse the decision maker's goals and feedback messages to understand their overall preferences.

Then, provide a list of questions you would ask the decision maker to better understand their internal utility model.

Your questions can be either general or referring to specific outcomes. For instance, you may ask the decision maker:

- questions clarifying the optimization objective,
- to rank two (or more) outcomes,
- how to improve certain outcomes,
- for a likert-scale rating regarding a specific outcome,
- etc.

When referring to specific outcomes, always state the arm\_index involved.

Your questions should help you predict pairwise preferences between any two experimental outcomes from the set of experimental outcomes provided above.

Return your final answer as a json file with the following format containing exactly {n\_questions} most important questions:

```
```json
{{
  "q1" : <question1>,
  ...
  "q{n_questions}" : <question{n_questions}>
}}
```

Prompt 2: The prompt used for question generation in the LILLO.get\_questions subroutine.

864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917

#### Pairwise comparisons

```
You are an expert in determining whether a human decision maker (DM)
is going to be satisfied with a set of experimental outcomes  $y = \{$ 
 $y\_names\}$ .

## All experimental outcomes:

{experiment_data}

## Human feedback messages:
We have also received the following messages from the DM:

{human_feedback}

{human_feedback_summary}

## Your task:
Given a pair of outcomes--option_0 and option_1, your goal is to
decide which one is more preferable according to the DM's preferences.

{pair_str}

Provide your prediction as a json file with the following format:
```json
{{
  "reasoning": "Your reasoning about the DM's preferences and
option_0 vs. option_1. Do not insert new lines in your reasoning.",
  "answer" : 0 or 1
}}
```
where in "answer" you should return 0 if option_0 is preferred, or 1
if option_1 is preferred.
Return just the json file (with the header ```json), nothing else.
```

Prompt 3: The prompt used for pairwise comparison labeling used in the LILO.get\_pairwise\_pref subroutine.

918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971

### Scalar utility estimation

You are an expert in determining whether a human decision maker (DM) is going to be satisfied with a set of experimental outcomes  $y = \{y\_names\}$ .

## Experimental outcomes:

So far, we have obtained the following experimental outcomes:

```
{experiment_data}
```

## Human feedback messages:

We have also received the following messages from the DM:

```
{human_feedback}
```

```
{human_feedback_summary}
```

## Your task:

Given the above your task is to predict the probability of the decision maker being satisfied with the experimental outcomes.

First, analyse the human feedback messages to understand the DM's preferences.

Then, provide your predictions for all  $y$ 's in the set of all experimental outcomes above.

Return your final answer as a jsonl file with the following format:

```
```jsonl
{{
  "arm_index": "{idx0}",
  "reasoning": <reasoning>,
  "p_accept": <probability>
}}
{{
  "arm_index": "{idx1}",
  "reasoning": <reasoning>,
  "p_accept": <probability>
}}
...
{{
  "arm_index": "{idxn}",
  "reasoning": <reasoning>,
  "p_accept": <probability>
}}
```
```

Where  $\langle reasoning \rangle$  should be a short reasoning for your prediction and  $\langle probability \rangle$  should be your best estimate for the probability between 0 and 1 that the DM will be satisfied with the corresponding outcome.

Provide your predictions for ALL  $y$ 's in the set of experimental outcomes above. That is, for EACH outcome from  $\{idx0\}$ . to  $\{idxn\}$ . Do not generate any Python code. Just return your predictions as plain text.

Prompt 4: The prompt used for scalar utility estimation used in the LILO.estimate.utilities subroutine.

972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025

#### Human Feedback Summarization

You are an expert in determining whether a human decision maker (DM) is going to be satisfied with a set of experimental outcomes  $y = \{y\_names\}$ .

## Experimental outcomes:

So far, we have obtained the following experimental outcomes:

```
{experiment_data}
```

## Human feedback messages:

We have also received the following messages from the DM:

```
{human_feedback}
```

## Your task:

Given the above your task is to summarize the human feedback messages into a clear description of the DM's optimization goals. Make your summary as quantitative as possible so that it can be easily used for utility estimation.

After analysis the human feedback messages, return your final answer as a json file with the following format:

```
```json
{
  "summary": <summary>
}
```

Remember about the ```json header!

**Prompt 5:** The prompt used for generating the `human_feedback_summary` by LILO for pairwise comparisons or scalar utility estimation.

1026  
 1027  
 1028  
 1029  
 1030  
 1031  
 1032  
 1033  
 1034  
 1035  
 1036  
 1037  
 1038  
 1039  
 1040  
 1041  
 1042  
 1043  
 1044  
 1045  
 1046  
 1047  
 1048  
 1049  
 1050  
 1051  
 1052  
 1053  
 1054  
 1055  
 1056  
 1057  
 1058  
 1059  
 1060  
 1061  
 1062  
 1063  
 1064  
 1065  
 1066  
 1067  
 1068  
 1069  
 1070  
 1071  
 1072  
 1073  
 1074  
 1075  
 1076  
 1077  
 1078  
 1079

#### Candidate Generation $n = 1$ , LILLO with prior knowledge

You are performing optimization of a utility function  $u(x) = g(y) = g(f(x))$ , where  $x$  is a vector of parameters:  $x = \{x\_names\}$  and  $y = f(x) = \{y\_names\}$  is a vector of outcomes. Each dimensions of  $x$  is in the range  $[0, 1]$ . Your goal is to find the parameters  $x$  that maximize the utility.

## Prior knowledge:

You have obtained the following prior knowledge about the experiment:  
`{prior_knowledge}`

## Human feedback messages:

You have also received the following messages from the DM:  
`{human_feedback}`

## Your task:

Given the above your task is the generate a set of `{n_candidates}` candidate parameters  $x$  for the next round of experimentation.

First, analyse the information above, then return your final answer as a json file with the following format:

```
```json
{
  "0": <candidate0>,
  "1": <candidate1>,
  ...
  "{n}": <candidate{n}>,
}
```

Where each `<candidatei>` is a list of the candidate parameter values in  $[0, 1]$ .

Do not write a python code for candidate generation. Just return the required json.

Do not add any comments to your json. Remember about the ````json` header.

Prompt 6: The prompt used for candidate generation by LILLO at  $n = 1$  when prior knowledge is available (LILLO.sample\_init\_candidates subroutine).

## B SIMULATION SETUP

We evaluate LILLO and the baselines on synthetic and real-world outcome functions as well as several utility functions. In all our test problems we have  $\mathcal{X} = \mathbb{R}^d$  and  $\mathcal{Y} = \mathbb{R}^k$ . The main benchmark considers three outcome functions: *DTLZ2* ( $d = 4, k = 8$ , Deb et al. (2002)), *Vehicle Safety* ( $d = 5, k = 3$ , Tanabe & Ishibuchi (2020)), and *Car Cab Design* ( $d = 7, k = 9$ , Liao et al. (2008)). These test problems are matched with several utility functions: piecewise linear, beta products, and the L1 distance. All outcome and utility functions are described in detail in this section.

In our simulations, we run the BO loop for  $T = 8$  iterations, setting  $B^{\text{exp}} = d$  (the dimension of the search space). At  $n = 1$  we hard-code the first message of the DM agent to seed the conversations. These messages are utility-dependent and presented in section B.2.

### B.1 OUTCOME FUNCTIONS

**DTLZ2** The DTLZ2 function was introduced by Deb et al. (2002), allowing for arbitrary input dimension  $d$  and output dimension  $k$  subject to  $d > k$ .  $\mathcal{X} = [0, 1]^d$ . For a DTLZ2 function  $f$  with a  $k$ -dimensional output and  $d$ -dimensional input, we have:

$$m := d - k + 1$$

$$h(x) := \sum_{i=m}^{d-1} (x_i - 0.5)^2$$

$$f_j(x) = -(1 + h(x)) \left( \prod_{i=1}^{k-j-1} \cos\left(\frac{\pi}{2} x_i\right) \right) \mathbb{1}_{j>1} \sin\left(\frac{\pi}{2} x_{k-j-1}\right)$$

In our experiments we use  $d = 8$  and  $k = 4$ .

**Vehicle Safety** This is a test problem for optimizing vehicle crash-worthiness with  $d = 5$  and  $k = 3$ .  $\mathcal{X} = [1, 3]^5$ . We refer the readers to Tanabe & Ishibuchi (2020) for details on function definition. We normalize each component of  $y = f(x)$  to lie between 0 and 1 based on empirical bound on the outcome space  $\mathcal{Y}$ .

**Car Cab Design** We refer the readers to Liao et al. (2008) for details. Note that in the original problem, there are stochastic components which we replace with deterministic components fixed at their original mean values in order to obtain a deterministic ground-truth outcome function. We normalize each component of  $y = f(x)$  to lie between 0 and 1 based on empirical bound on the outcome space  $\mathcal{Y}$ .

**Thermal Comfort** The problem setting follows the ISO 7730 and ASHRAE 55 models, which predict human thermal sensation and dissatisfaction based on six core factors: air temperature, mean radiant temperature, humidity, air speed, clothing insulation, and metabolic rate (Fanger, 1970; ISO7730, 2005; ASHRAE55, 2020). From these, outcome measures such as Predicted Percentage Dissatisfied (PPD), draft risk (DR), vertical air temperature difference, radiant temperature asymmetry, and floor surface temperature are derived, each with threshold values associated with acceptable comfort. The goal of the optimization agent is to find environmental parameters that minimize discomfort and keep all outcomes within desirable ranges, reflecting realistic expectations of the occupant rather than arbitrary synthetic functions. This setting is widely used in thermal comfort research and can be visualized via the CBE Thermal Comfort Tool (Tartarini et al., 2020). In our implementation, the outcome function has two fixed, non-optimizable parameters: clothing insulation ( $\text{clo} \in [0.3, 1.2]$ ) and metabolic rate ( $\text{met} \in [1.0, 2.0]$ ) which differ for the two versions of the environments considered in this paper, as detailed in the next section.

### B.2 UTILITY FUNCTIONS

**L1 distance** We consider a utility function which is the L1 distance from a pre-specified point  $y_{\text{opt}}$ . This choice of the utility function mimics the scenario where the DM wishes to keep the outcomes close to a specific desirable state.

For DTLZ2, we set  $y_{\text{opt}} = [0.8, 1.0, 0.7, 1.25]$ .

The message seeding the conversation takes the following form:

## Goal message (LI distance)

My goal is to bring all the outcome metrics as close to {opt\_y} as possible.

**Beta products** Prior work on preference learning has utilized the Beta CDF to form utility functions. The Beta CDF provides a convenient, bounded monotonic transform that smoothly varies between increasing and decreasing marginal gains with respect to their inputs. Our utility function takes the following form:

$$g(y; \alpha, \beta) = \prod_{i=1}^k \text{BetaCDF}(y_i; \alpha_i, \beta_i)$$

For the DTLZ2 outcome function we set:

$$\alpha = [0.5, 2.0, 2.0, 2.0]$$

$$\beta = [0.5, 1.0, 2.0, 5.0]$$

For the Vehicle Safety outcome function we set:

$$\alpha = [0.5, 1.0, 1.5]$$

$$\beta = [1, 2, 3]$$

For this utility function, the message seeding the conversation takes the following form:

## Goal message (beta products)

My goal is to bring all the outcome metrics as close to 1 as possible. Results are strongest only when every metric is high -- if any metric is low, it significantly reduces the overall performance.

**Piecewise linear** Analogously to Lin et al. (2022) we also consider a piecewise linear utility function. Its shape corresponds to diminishing marginal returns on outcomes and sharp declines in utility when desired thresholds are not met. For a  $k$ -dimensional input vector  $y$ , this utility function is defined as:

$$g(y; \beta_1, \beta_2, t) = \sum_{i=1}^k h_i(y_i),$$

where

$$h_i(y_i) = \begin{cases} \beta_{1,i}y_i + (\beta_{1,i} - \beta_{2,i})t_i & \text{if } y_i < t_i \\ \beta_{2,i}y_i & \text{if } y_i \geq t_i \end{cases}$$

For the DTLZ2 problem, we set

$$\beta_1 = [4.0, 3.0, 2.0, 1.0]$$

$$\beta_2 = [0.4, 0.3, 0.2, 0.1]$$

$$t = [1.0, 0.8, 0.5, 0.5]$$

For the Vehicle Safety problem, we set

$$\beta_1 = [2, 6, 8]$$

$$\beta_2 = [1, 2, 2]$$

$$t = [0.5, 0.8, 0.8]$$

For the Car Cab Design problem, we set

$$\beta_1 = [7.0, 6.75, 6.5, 6.25, 6.0, 5.75, 5.5, 5.25, 5.0, 4.75, 4.5]$$

$$\beta_2 = [0.5, 0.4, 0.375, 0.35, 0.325, 0.3, 0.275, 0.25, 0.225, 0.2, 0.175]$$

$$t = [0.64, 0.68, 0.96, 0.88, 1.06, 0.65, 0.84, 0.86, 0.58, 0.7, 0.53]$$

Here, thresholds  $t_i$  correspond to the 0.75 quintiles of the outcome values  $y_i$ . The seeding message takes the following form:

Goal message (piecewise linear)

My goal is to achieve the following thresholds in each outcome  $\{t\}$ . Improvements over the thresholds are always good, but less important than bringing the outcomes to their threshold values. The further away an outcome is from its threshold, the higher is its negative impact on the overall performance.

**Thermal Comfort** Our utility maps the outcome vector  $Y = [\text{PPD}, \text{DR}, dT_{\text{vert}}, dT_{\text{pr}}, T_{\text{floor}}]$  to a scalar  $U \in [0, 1]$  via per-outcome desirabilities that enforce being “within range”, with using the Derringer-Suich desirability functions. For the four “smaller is better” outcomes (PPD, DR,  $dT_{\text{vert}}$ ,  $dT_{\text{pr}}$ ) we use a one-sided acceptable band with a comfort threshold  $L$  and an unacceptable threshold  $H$  and define

$$d_{\text{small}}(y; L, H, s) = \begin{cases} 1, & y \leq L, \\ \left( \frac{H - y}{H - L} \right)^s, & L < y < H, \\ 0, & y \geq H, \end{cases}$$

so values at or below  $L$  are fully desirable, values beyond  $H$  are unacceptable, and values in between taper smoothly with shape  $s \geq 1$ . For floor temperature  $T_{\text{floor}}$  we target a comfort band  $[l, h]$  and tolerate a wider band  $[l_{\text{min}}, h_{\text{max}}]$  by

$$d_{\text{band}}(t; l, h, l_{\text{min}}, h_{\text{max}}, s) = \begin{cases} 1, & l \leq t \leq h, \\ \left( \frac{t - l_{\text{min}}}{l - l_{\text{min}}} \right)^s, & l_{\text{min}} < t < l, \\ \left( \frac{h_{\text{max}} - t}{h_{\text{max}} - h} \right)^s, & h < t < h_{\text{max}}, \\ 0, & t \leq l_{\text{min}} \text{ or } t \geq h_{\text{max}}. \end{cases}$$

The overall utility is the geometric mean of the five desirabilities,

$$U(Y) = \left( d_{\text{small}}(\text{PPD}) \cdot d_{\text{small}}(\text{DR}) \cdot d_{\text{small}}(dT_{\text{vert}}) \cdot d_{\text{small}}(dT_{\text{pr}}) \cdot d_{\text{band}}(T_{\text{floor}}) \right)^{1/5},$$

We consider two versions of this utility functions with varying comfortable ranges of the outcome metrics

**Type A.** These settings are meant to simulate preferences of an office worker in light clothing and a moderate tolerance for different conditions.

$$\begin{aligned} l_{\text{PPD}} &= 0.0, & h_{\text{PPD}} &= 30.0, \\ l_{\text{DR}} &= 10.0, & h_{\text{DR}} &= 35.0, \\ l_{dT_{\text{vert}}} &= 3.0, & h_{dT_{\text{vert}}} &= 9.0, \\ l_{dT_{\text{pr}}} &= 5.0, & h_{dT_{\text{pr}}} &= 22.0, \\ l_{\text{min}, T_{\text{floor}}} &= 16.0, & l_{T_{\text{floor}}} &= 19.0, & h_{T_{\text{floor}}} &= 26.0, & h_{\text{min}, T_{\text{floor}}} &= 30.0. \end{aligned}$$

In Thermal Comfort Type A the clothing and metabolic rate parameters of the outcome function are set to  $\text{clo} = 0.61$  and  $\text{met} = 1.0$ , respectively.

**Type B.** These settings are meant to simulate preferences of a summer athlete wearing light sport kit, with a lower tolerance for adverse conditions.

$$\begin{aligned}
 l_{\text{PPD}} &= 0.0, & h_{\text{PPD}} &= 24.0, \\
 l_{\text{DR}} &= 30.0, & h_{\text{DR}} &= 45.0, \\
 l_{\text{dT}_{\text{vert}}} &= 2.5, & h_{\text{dT}_{\text{vert}}} &= 6.0, \\
 l_{\text{dT}_{\text{pr}}} &= 4.0, & h_{\text{dT}_{\text{pr}}} &= 12.0, \\
 l_{\text{min}, \text{T}_{\text{floor}}} &= 19.0, & l_{\text{T}_{\text{floor}}} &= 20.0, & h_{\text{T}_{\text{floor}}} &= 23.0, & h_{\text{min}, \text{T}_{\text{floor}}} &= 25.0.
 \end{aligned}$$

In Thermal Comfort Type B the clothing and metabolic rate parameters of the outcome function are set to  $\text{clo} = 0.3$  and  $\text{met} = 2.0$ , respectively.

The seeding message takes the following form:

Goal message (Thermal Comfort)

My goal is to keep all metrics within my thermal comfort preferences.

### B.3 LLM-BASED SIMULATION OF THE HUMAN PREFERENCE FEEDBACK

Evaluating black box optimization algorithm rigorously is challenging due to the replications required to discern performance in the presence of the inherent variance of the evaluation of the optimization traces. This is exacerbated in our setting where the algorithm is based on feedback from human decision makers. To be able to evaluate LILLO rigorously, we therefore simulate the human preference feedback with another LLM.

For all our experiments we use Llama-3.3-70b-instruct as the language model for the human feedback simulator. A high-level representation of the prompt used to generate the answers is presented in prompt 7. `utility_func_desc` is a textual description of the specific utility function. `outcomes_markdown` is a markdown-formatted table of outcomes  $y_i \in D^{\text{exp}}$  and their corresponding pre-computed ground-truth utilities  $g(y_i)$ . `questions_str` are the questions  $\{q_j\}_{j=1}^{B^{\text{pr}}}$  generated by LILLO. Finally `utility_constraints` contain additional utility-specific instructions for the LLM to generate human-like feedback and to not reveal explicitly the exact functional form of the utility, ensuring the generated answers sound natural.

### B.4 PRIOR KNOWLEDGE MESSAGES $P_{\text{PRIOR}}$

Below we present the prior messages  $P_{\text{prior}}$  used in our experiments from section 5.3.3.

DTLZ2 (point knowledge)

- Based on my experience, the following inputs should bring good results: {promising\_point}.

DTLZ2 (area knowledge)

- Based on my experience, inputs within these ranges should bring good results {bounds}:

Vehicle Safety (domain knowledge)

- `y_1` measures the reduction in vehicle's mass, `y_2` measures the reduction in integration of acceleration between two time points, `y_3` measures the reduction in toe board intrusion in the offset-frontal crash.  
 - The parameters `x` measure the thickness of five reinforced members around the frontal structure of a car, which can significantly affect the crash safety.

1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349

#### Car Cab Design (domain knowledge)

- A car is subjected to a side-impact crash test. The outcome variables  $y$  measure the following:

- The effect of the side-impact on a dummy is measured in terms of head injury, load in abdomen, pubic symphysis force, viscous criterion ( $V * C$ ), and rib deflections at the upper, middle and lower rib locations.
- The effect on the car are considered in terms of the vehicle's weight, the velocity of the B-Pillar at the middle point and the velocity of the front door at the B-Pillar.
- The parameters  $x$  describe some design aspects of the car. An increase in dimension of the car parameters may improve safety, but with a burden of an increased weight of the car. These parameters are and their ranges are:
  - x1: Thickness of B-Pillar inner [0.5, 1.5]
  - x2: Thickness of B-Pillar reinforcement [0.45, 1.35]
  - x3: Thickness of floor side inner [0.5, 1.5]
  - x4: Thickness of cross members [0.5, 1.5]
  - x5: Thickness of door beam [0.875, 2.625]
  - x6: Thickness of door beltline reinforcement [0.4, 1.2]
  - x7: Thickness of roof rail [0.4, 1.2]
- NOTE: The presented values of outcomes  $y$  represent the reduction in mass, forces, velocities etc. So the goal is to increase  $y_1, \dots, y_{11}$ , corresponding to lowering the vehicle's weight and minimizing the impact on the dummy and the car.

## B.5 BASELINES

### B.5.1 QUANTITATIVE BASELINES

The quantitative baseline methods follow an analogous procedure to LILLO, where the Q&A natural language feedback is replaced with either scalar values of the utilities associated with a batch of outcomes (true utility BO) or pairwise comparisons between outcomes based on their ground-truth utilities (preferential BO). This feedback is obtained on  $B^{pf}$  outcomes  $y_i$  or paired outcomes  $(y_i, y'_i)$  sampled with the EUBO acquisition function. To extend predictions to the entire data set  $D_n^{exp}$ , we fit a simple / pairwise GP model  $M_n^y : \mathcal{Y} \rightarrow \mathcal{P}(\mathbb{R})$ . Subsequently a proxy model  $M_n^x : \mathcal{X} \rightarrow \mathcal{P}(\mathbb{R})$  is fit to the predictions of  $M_n^y$ .

We present the exact implementation of these methods in algorithms 5 and 6.

### B.5.2 LLM 2-STEP AND DIRECT

We also implement two versions of an LLM-based approach to candidate generation for optimization.

**LLM (2-step)** follows the same algorithm 2 as LILLO with the following exceptions. In the candidate generation step, instead of using the LogNEI acquisition function, we prompt the LLM to generate a set of candidates using prompt 8.

**LLM (direct)** Omits the step of utility estimation and generates the candidates directly based on the raw human feedback and observed inputs and outcomes from  $D_n^{exp}$ . The prompt used is presented in prompt 9

1350  
 1351  
 1352  
 1353  
 1354  
 1355  
 1356  
 1357  
 1358  
 1359  
 1360  
 1361  
 1362  
 1363  
 1364  
 1365  
 1366  
 1367  
 1368  
 1369  
 1370  
 1371  
 1372  
 1373  
 1374  
 1375  
 1376  
 1377  
 1378  
 1379  
 1380  
 1381  
 1382  
 1383  
 1384  
 1385  
 1386  
 1387  
 1388  
 1389  
 1390  
 1391  
 1392  
 1393  
 1394  
 1395  
 1396  
 1397  
 1398  
 1399  
 1400  
 1401  
 1402  
 1403

#### DM's answer generation

```

Suppose you are a decision maker evaluating the results of a multi-
objective optimization problem.

You are given a set of multi-dimensional outcomes  $y = \{y\_names\}$ 
{utility_func_desc}

You have observed the following outcomes with their corresponding
utility values and contributions to the overall utility.

## Outcomes:
{outcomes_markdown}

The utility values are on a scale [0, 1], where (1 - very satisfied,
0.5 - neutral, 0 - very dissatisfied).

Based on the above information, provide answers to the following
questions:

## Questions:
{questions_str}

Return your final answer as a json file with the following format:
```json
{{
  "q1" : <answer to q1>,
  ...
  "q{n_questions}" : <answer to q{n_questions}>
}}
```

Before providing your final answers, analyze the shape of the utility
function in light of the questions posed.
In your final answers, you cannot reveal the explicit formula of the
utility function.
The form and the values of the utility functions is a "latent" feature
of the human expert, thus you should not refer to it explicitly or
even mention its existence.
{utility_constraints}
State your answers in the first person (you are the decision maker).
Avoid vacuous statements.

```

Prompt 7: The prompt used for answer generation in the `DM.get_answers` subroutine.

1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457

#### Candidate Generation, LLM (2-step)

```

You are performing optimization of a utility function  $u(x) = g(y) = g(f(x))$ , where  $x$  is a vector of parameters:  $x = \{x\_names\}$  and  $y = f(x) = \{y\_names\}$  is a vector of outcomes.
Each dimensions of  $x$  is in the range  $[0, 1]$ .
Your goal is to find the parameters  $x$  that maximize the utility.

## Experimental Outcomes
So far, you have also observed the following inputs  $x$  and their estimated utilities:

{experiment_data}

## Human feedback messages:
We have also received the following messages from the DM:

{human_feedback}

## Your task:
Given the above your task is the generate a set of {n_candidates} candidate parameters  $x$  for the next round of experimentation.
Your candidates should maximize the expected improvement over the current best candidate  $x^* = \{x\_star\}$  with utility  $u(x^*) = \{u\_star\}$ .

First, analyse the information above, then return your final answer as a json file with the following format:
```json
{{
  "0": <candidate0>,
  "1": <candidate1>,
  ...
  "{n}": <candidate{n}>,
}}
```
Where each <candidatei> is a list of the candidate parameter values in  $[0, 1]$ .
Do not write a python code for candidate generation. Just return the required json.
Do not add any comments to your json. Remember about the ```json header.

```

**Prompt 8:** The prompt used for candidate generation by the LLM (2-step) baseline. In the above, `experiment_data` is a markdown formatted table of outcomes  $y_i$  and their estimate utilities via the LLM-based proxy model  $M_{n-1}^y(y_i)$ . `x_star` and `u_star` are determined based on the estimated utilities (not ground truth  $g(y_i)$ 's as these are latent, non-observable quantities).

1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511

## Candidate Generation, LLM (direct)

```
You are performing optimization of a utility function  $u(x) = g(y) = g(f(x))$ , where  $x$  is a vector of parameters:  $x = \{x\_names\}$  and  $y = f(x) = \{y\_names\}$  is a vector of outcomes.
Each dimensions of  $x$  is in the range  $[0, 1]$ .
Your goal is to find the parameters  $x$  that maximize the utility.
```

```
{experiment_data}
```

```
## Human feedback messages:
```

```
We have also received the following messages from the DM:
```

```
{human_feedback}
```

```
## Your task:
```

```
Given the above your task is the generate a set of {n_candidates} candidate parameters  $x$  for the next round of experimentation.
```

```
First, analyze the human feedback messages to understand the DM's preferences.
```

```
Then, generate a set of {n_candidates} candidate parameters  $x$ , trading -off exploration and exploitation.
```

```
Return your final answer as a json file with the following format:
```

```
```json
{
  "0": <candidate0>,
  "1": <candidate1>,
  ...
  "{n}": <candidate{n}>,
}
```

```
Where each <candidatei> is a list of the candidate parameter values: {  $x\_names$  }, each in  $[0, 1]$ .
```

```
Do not write a python code for candidate generation. Just return the required json.
```

```
Do not add any comments to your json.
```

Prompt 9: The prompt used for candidate generation in the LLM (direct) baseline. `experiment_data` is a markdown-formatted table of inputs and outcomes. `human_feedback` is the set of questions and answers from  $D_{n-1}^{pf}$ .

1512  
 1513  
 1514  
 1515  
 1516  
 1517  
 1518  
 1519  
 1520  
 1521  
 1522  
 1523  
 1524  
 1525  
 1526  
 1527  
 1528  
 1529  
 1530  
 1531  
 1532  
 1533  
 1534  
 1535  
 1536  
 1537  
 1538  
 1539  
 1540  
 1541  
 1542  
 1543  
 1544  
 1545  
 1546  
 1547  
 1548  
 1549  
 1550  
 1551  
 1552  
 1553  
 1554  
 1555  
 1556  
 1557  
 1558  
 1559  
 1560  
 1561  
 1562  
 1563  
 1564  
 1565

---

**Algorithm 5:** True utility BO
 

---

**Input:** Max number of iterations  $T$ , Experiment batch size  $B^{\text{exp}}$ , Feedback batch size  $B^{\text{pf}}$ .

$D_0^{\text{exp}} \leftarrow \emptyset$

$D_0^{\text{pf}} \leftarrow \emptyset$

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

  # Sample a batch of candidates

**if**  $n = 1$  **then**

$\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{Uniform}(\mathcal{X})$

**else**

$\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{LogEI}(\mathcal{X}; M_{n-1}^x)$

  # Run the experiments and update the experimental dataset

$D_n^{\text{exp}} \leftarrow D_{n-1}^{\text{exp}} \cup \{(x_i, y_i) : y_i = f(x_i)\}_{i=1}^{B^{\text{exp}}}$

  # Sample a batch of outcomes for feedback

**if**  $n = 1$  **then**

$\{y_i\}_{i=1}^{B^{\text{pf}}} \sim \text{random\_sample}(D^{\text{exp}})$

**else**

$\{y_i\}_{i=1}^{B^{\text{pf}}} \sim \text{EUBO}(D_n^{\text{exp}}, M_{n-1}^y)$

  # Update the preference feedback dataset

$D_n^{\text{pf}} \leftarrow D_{n-1}^{\text{pf}} \cup \{(y_j, u_j) : u_j = g(y_j)\}_{j=1}^{B^{\text{pf}}}$

  # Fit a Y->U GP

$M_n^y \leftarrow \text{fit\_simple\_gp}(D_n^{\text{pf}})$

  # Label all experimental datapoints

$D_n^{\text{GP}} \leftarrow \{(x_i, y_i, \hat{u}_i) : (x_i, y_i) \in D_n^{\text{exp}}\}$  //  $\hat{u}_i$ 's are the mean predictions  
     with respect to  $M_n^y : \mathcal{Y} \rightarrow \mathcal{P}(\mathbb{R})$

  # Fit the proxy model

$D_n^{\text{GP},x} \leftarrow \{(x_i, \hat{u}_i) : (x_i, y_i, \hat{u}_i) \in D_n^{\text{GP}}\}$

$M_n^x \leftarrow \text{fit\_simple\_gp}(D_n^{\text{GP},x})$

---

1566  
 1567  
 1568  
 1569  
 1570  
 1571  
 1572  
 1573  
 1574  
 1575  
 1576  
 1577  
 1578  
 1579  
 1580  
 1581  
 1582  
 1583  
 1584  
 1585  
 1586  
 1587  
 1588  
 1589  
 1590  
 1591  
 1592  
 1593  
 1594  
 1595  
 1596  
 1597  
 1598  
 1599  
 1600  
 1601  
 1602  
 1603  
 1604  
 1605  
 1606  
 1607  
 1608  
 1609  
 1610  
 1611  
 1612  
 1613  
 1614  
 1615  
 1616  
 1617  
 1618  
 1619

---

**Algorithm 6:** Preferential BO
 

---

**Input:** Max number of iterations  $T$ , Experiment batch size  $B^{\text{exp}}$ , Feedback batch size  $B^{\text{pf}}$ .

$D_0^{\text{exp}} \leftarrow \emptyset$ ,

$D_0^{\text{pf}} \leftarrow \emptyset$

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

  # Sample a batch of candidates

**if**  $n = 1$  **then**

$\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{Uniform}(\mathcal{X})$

**else**

$\{x_i\}_{i=1}^{B^{\text{exp}}} \sim \text{LogNEI}(\mathcal{X}; M_{n-1}^x)$

  # Run the experiments and update the experimental dataset

$D_n^{\text{exp}} \leftarrow D_{n-1}^{\text{exp}} \cup \{(x_i, y_i) : y_i = f(x_i)\}_{i=1}^{B^{\text{exp}}}$

  # Sample a batch of paired outcomes for feedback

**if**  $n = 1$  **then**

$\{(y_i, y'_i)\}_{i=1}^{B^{\text{pf}}} \sim \text{random\_sample}(D^{\text{exp}})$

**else**

$\{(y_i, y'_i)\}_{i=1}^{B^{\text{pf}}} \sim \text{EUBO}(D_n^{\text{exp}}, M_{n-1}^y)$

  # Update the preference feedback dataset

$D_n^{\text{pf}} \leftarrow D_{n-1}^{\text{pf}} \cup \{(y_j, y'_j, p_j) : p_j = \mathbb{1}\{g(y_j) > g(y'_j)\}\}_{j=1}^{B^{\text{pf}}}$

  # Fit a Y->U GP

$M_n^y \leftarrow \text{fit\_pairwise\_gp}(D_n^{\text{pf}})$

  # Label all experimental datapoints

$D_n^{\text{GP}} \leftarrow \{(x_i, y_i, \hat{u}_i) : (x_i, y_i) \in D_n^{\text{exp}}\}$  //  $\hat{u}_i$ 's are the mean predictions  
     with respect to  $M_n^y : \mathcal{Y} \rightarrow \mathcal{P}(\mathbb{R})$

  # Fit the proxy model

$D_n^{\text{GP},x} \leftarrow \{(x_i, \hat{u}_i) : (x_i, y_i, \hat{u}_i) \in D_n^{\text{GP}}\}$

$M_n^x \leftarrow \text{fit\_simple\_gp}(D_n^{\text{GP},x})$

---

C ADDITIONAL EXPERIMENTAL RESULTS

C.1 LLM ABLATION STUDY

We compare the performance of LILLO depending on the choice of the LLM model. We compare the following models: Llama-3.3-70b-instruct, Llama-4-scout-17b-16e-instruct, and Qwen-3-14b. The LLM used to simulate human feedback remains set to Llama-3.3-70b-instruct across the comparisons.

We run the ablation study for the following 3 environments: the DTLZ2 outcome function, combined with the L1, beta products and piecewise linear utility functions. As in the main benchmark, we set  $B^{pf} = 2$  and  $B^{exp} = d = 8$ .

**Results.** Table 1 presents the results. We observe that LILLO performs similarly across all three LLMs, demonstrating that the success of our method is agnostic to the choice of a specific language model. We were unable to test the performance of LILLO with smaller language models (e.g. LLama-3.1-8B or Qwen-3-8b) due to difficulties in ensuring that the LLM’s outputs follow the required json structure, leading to parsing errors.

C.2 PREFERENCE FEEDBACK BATCH SIZE ABLATION

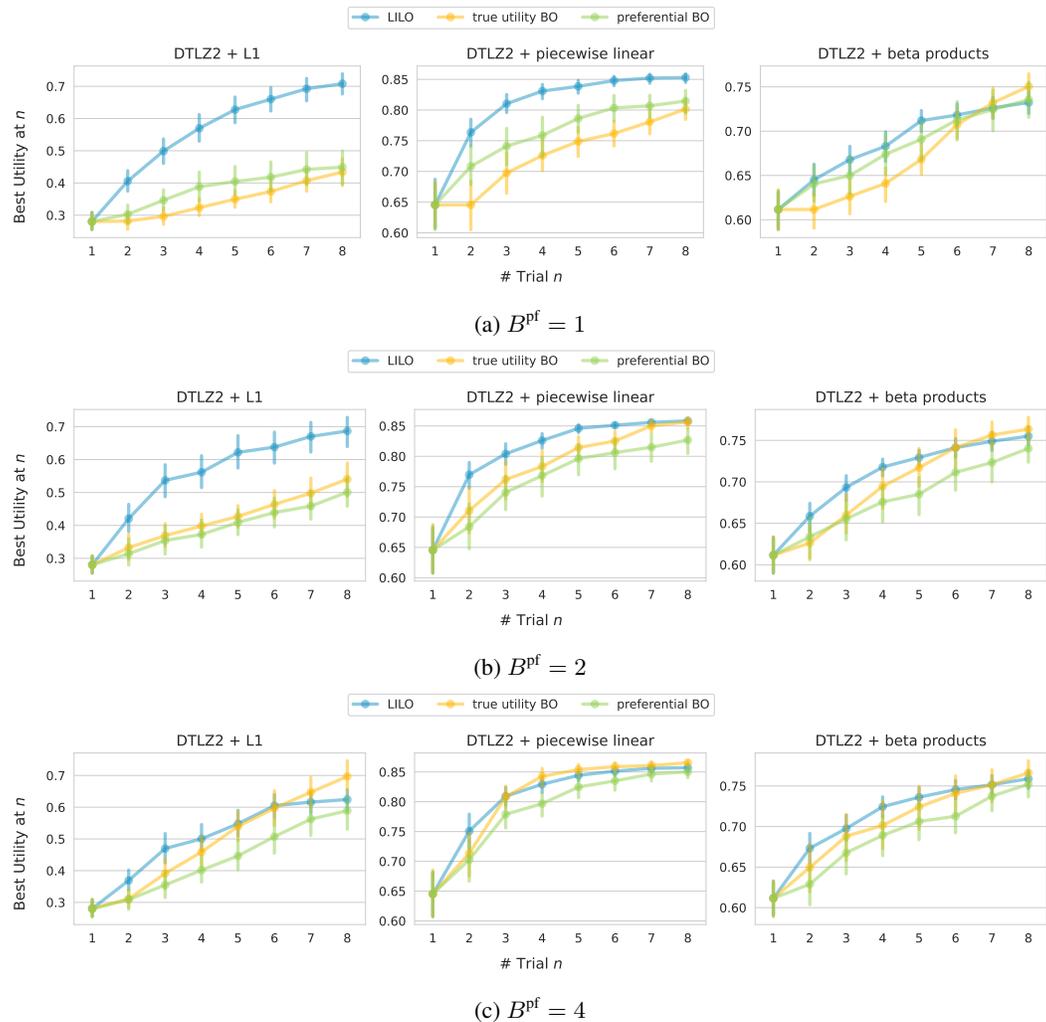


Figure 6: LILLO vs. baselines for different values of  $B^{pf}$ . Max value of the ground-truth utility achieved after  $n$  iterations. Error bars are 95% confidence intervals of the mean max value across 30 simulation replications.

Table 1: *LLM ablation study*. Max value of the ground-truth utility achieved after  $n$  iterations. Error bars are 1 standard deviation of the mean across 30 simulation replications.

(a) DTLZ2 + L1

method # trial	LILO (Llama-3.3-70b)	LILO (Llama-4-scout)	LILO (Qwen3-14B)	preferential BO	true utility BO
1	$0.28 \pm 0.01$	$0.28 \pm 0.01$	$0.28 \pm 0.01$	$0.28 \pm 0.01$	$0.28 \pm 0.01$
2	$0.42 \pm 0.02$	$0.43 \pm 0.02$	$0.42 \pm 0.02$	$0.31 \pm 0.02$	$0.33 \pm 0.02$
3	$0.54 \pm 0.03$	$0.53 \pm 0.02$	$0.54 \pm 0.03$	$0.35 \pm 0.02$	$0.37 \pm 0.02$
4	$0.56 \pm 0.03$	$0.59 \pm 0.02$	$0.59 \pm 0.03$	$0.37 \pm 0.02$	$0.4 \pm 0.02$
5	$0.62 \pm 0.02$	$0.64 \pm 0.02$	$0.63 \pm 0.03$	$0.41 \pm 0.02$	$0.43 \pm 0.02$
6	$0.64 \pm 0.02$	$0.67 \pm 0.02$	$0.66 \pm 0.03$	$0.44 \pm 0.02$	$0.46 \pm 0.02$
7	$0.67 \pm 0.02$	$0.7 \pm 0.02$	$0.69 \pm 0.02$	$0.46 \pm 0.02$	$0.5 \pm 0.02$
8	$0.69 \pm 0.02$	$0.71 \pm 0.02$	$0.71 \pm 0.02$	$0.5 \pm 0.02$	$0.54 \pm 0.03$

(b) DTLZ2 + beta products

method # trial	LILO (Llama-3.3-70b)	LILO (Llama-4-scout)	LILO (Qwen3-14B)	preferential BO	true utility BO
1	$0.61 \pm 0.01$	$0.61 \pm 0.01$	$0.61 \pm 0.01$	$0.61 \pm 0.01$	$0.61 \pm 0.01$
2	$0.66 \pm 0.01$	$0.67 \pm 0.01$	$0.66 \pm 0.01$	$0.63 \pm 0.01$	$0.63 \pm 0.01$
3	$0.69 \pm 0.01$	$0.69 \pm 0.01$	$0.69 \pm 0.01$	$0.66 \pm 0.01$	$0.66 \pm 0.01$
4	$0.72 \pm 0.01$	$0.71 \pm 0.01$	$0.7 \pm 0.01$	$0.68 \pm 0.01$	$0.69 \pm 0.01$
5	$0.73 \pm 0.0$	$0.73 \pm 0.01$	$0.71 \pm 0.01$	$0.69 \pm 0.01$	$0.72 \pm 0.01$
6	$0.74 \pm 0.01$	$0.74 \pm 0.01$	$0.73 \pm 0.01$	$0.71 \pm 0.01$	$0.74 \pm 0.01$
7	$0.75 \pm 0.01$	$0.75 \pm 0.01$	$0.73 \pm 0.01$	$0.72 \pm 0.01$	$0.76 \pm 0.01$
8	$0.76 \pm 0.01$	$0.75 \pm 0.0$	$0.73 \pm 0.01$	$0.74 \pm 0.01$	$0.76 \pm 0.01$

(c) DTLZ2 + piecewise linear

method # trial	LILO (Llama-3.3-70b)	LILO (Llama-4-scout)	LILO (Qwen3-14B)	preferential BO	true utility BO
1	$0.65 \pm 0.02$	$0.65 \pm 0.02$	$0.65 \pm 0.02$	$0.65 \pm 0.02$	$0.65 \pm 0.02$
2	$0.77 \pm 0.01$	$0.76 \pm 0.01$	$0.76 \pm 0.01$	$0.68 \pm 0.02$	$0.71 \pm 0.02$
3	$0.8 \pm 0.01$	$0.82 \pm 0.01$	$0.82 \pm 0.01$	$0.74 \pm 0.01$	$0.76 \pm 0.02$
4	$0.83 \pm 0.01$	$0.83 \pm 0.01$	$0.84 \pm 0.01$	$0.77 \pm 0.02$	$0.78 \pm 0.01$
5	$0.85 \pm 0.0$	$0.84 \pm 0.01$	$0.85 \pm 0.0$	$0.8 \pm 0.01$	$0.81 \pm 0.01$
6	$0.85 \pm 0.0$	$0.85 \pm 0.01$	$0.85 \pm 0.0$	$0.81 \pm 0.01$	$0.83 \pm 0.01$
7	$0.86 \pm 0.0$	$0.85 \pm 0.0$	$0.86 \pm 0.0$	$0.81 \pm 0.01$	$0.85 \pm 0.01$
8	$0.86 \pm 0.0$	$0.86 \pm 0.0$	$0.86 \pm 0.0$	$0.83 \pm 0.01$	$0.86 \pm 0.0$

### C.3 PREFERENCE FEEDBACK BATCH SIZE ABLATION (EXTENDED)

It is not straightforward to compare the DM’s workload necessary to answer natural language questions vs. providing e.g. pairwise comparisons. The DM’s workload would depend heavily on the specific questions being asked, and in the case of pairwise comparisons, it would depend on the complexity of the utility function and the kind of outcomes being presented. Experiments presented in this subsection are aimed at understanding how many pairwise comparisons or queries to the ground-truth utility are roughly equivalent to a single message of the DM in natural language. In Figures 7, 8, and 9, we compare the performance of LILLO with  $B^{pf} \in \{1, 2\}$  with the quantitative baselines (true utility BO and preferential BO) with  $B^{pf} \in \{1, 2, 4, 8, 16\}$ . The results demonstrate that even one natural language statement can outperform as many as 8-16 pairwise comparisons or point-wise evaluations of the ground-truth utility. The competitive performance of LILLO is most pronounced at the very initial stages of experimentation. This underscores the sample efficiency and information density of natural language feedback, when used effectively.

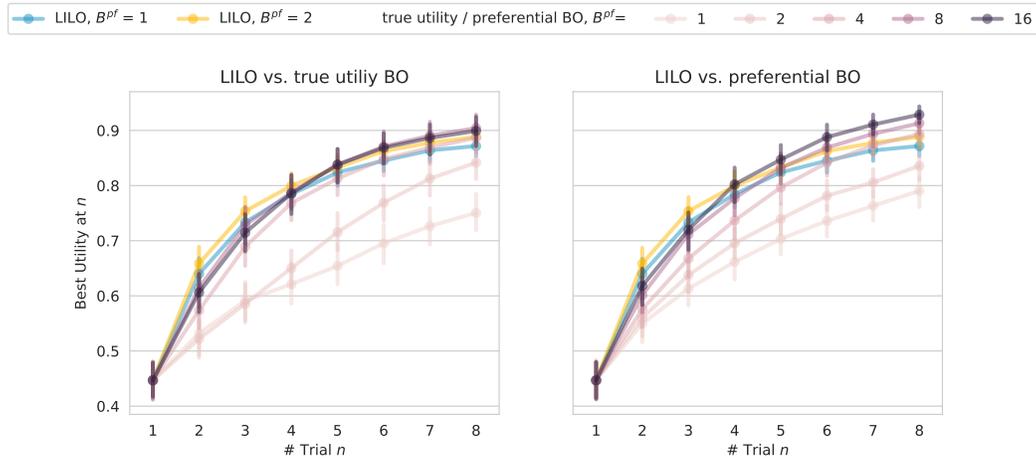


Figure 7: *LILLO vs. preferential BO and true utility BO with varying feedback batch size.* Results averaged across all environments, with min-max normalization applied within each environment prior to aggregation.

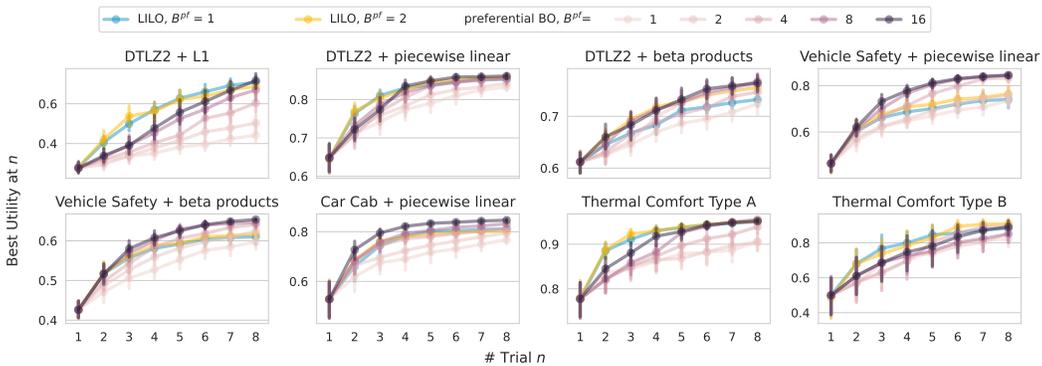


Figure 8: *LILLO vs. preferential BO with varying feedback batch size.* Results by environment.

1782  
 1783  
 1784  
 1785  
 1786  
 1787  
 1788  
 1789  
 1790  
 1791  
 1792  
 1793  
 1794  
 1795  
 1796  
 1797  
 1798  
 1799  
 1800  
 1801  
 1802  
 1803  
 1804  
 1805  
 1806  
 1807  
 1808  
 1809  
 1810  
 1811  
 1812  
 1813  
 1814  
 1815  
 1816  
 1817  
 1818  
 1819  
 1820  
 1821  
 1822  
 1823  
 1824  
 1825  
 1826  
 1827  
 1828  
 1829  
 1830  
 1831  
 1832  
 1833  
 1834  
 1835

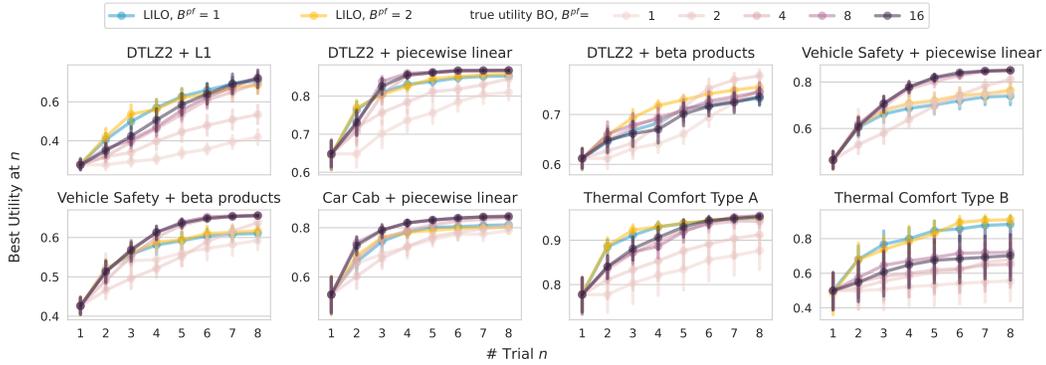


Figure 9: *LILLO* vs. *true utility BO* with varying feedback batch size. Results by environment.

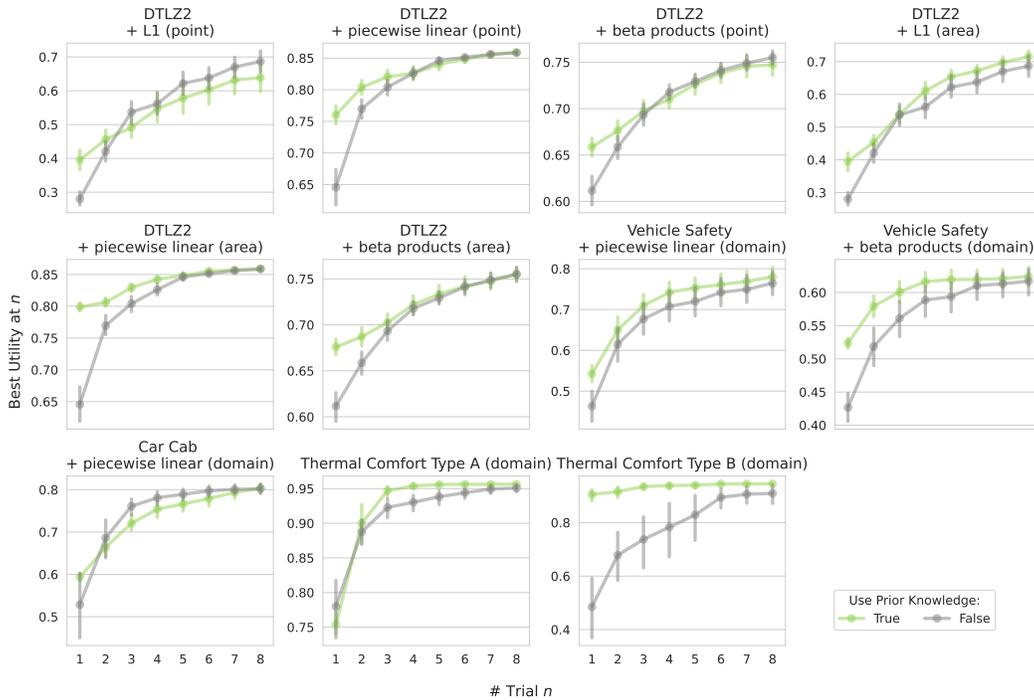


Figure 10: Performance of *LILLO* with and without prior knowledge. Results across environments and knowledge types.

#### C.4 INCORPORATING PRIOR KNOWLEDGE

In Figure 10 we present an environment-by-environment view of the results from Section 5.3.3.

#### C.5 BASELINES – FEEDBACK ACQUISITION ABLATION

This section is aimed to justify the choice of feedback acquisition functions used by the quantitative baselines.

Table 2: *The choice of feedback acquisition function.* Values represent the average of the best ground-truth utility at iteration  $n$ . Averaged across three simulation environments and 30 replications per environment; min-max standardized within an environment before aggregation. Error bars represent 1 standard deviation of the mean.

(a) True Utility BO

Method # trial	EUBO	Max Value Entropy Search	random
1	$0.45 \pm 0.017$	$0.45 \pm 0.017$	$0.45 \pm 0.017$
2	$0.537 \pm 0.017$	$0.528 \pm 0.017$	$0.529 \pm 0.017$
3	$0.61 \pm 0.017$	$0.587 \pm 0.017$	$0.603 \pm 0.017$
4	$0.668 \pm 0.017$	$0.652 \pm 0.017$	$0.668 \pm 0.017$
5	$0.723 \pm 0.017$	$0.712 \pm 0.017$	$0.721 \pm 0.017$
6	$0.775 \pm 0.016$	$0.762 \pm 0.017$	$0.771 \pm 0.016$
7	$0.812 \pm 0.016$	$0.806 \pm 0.017$	$0.802 \pm 0.016$
8	$0.851 \pm 0.015$	$0.84 \pm 0.016$	$0.831 \pm 0.016$

(b) Preferential BO

Method # trial	EUBO	BALD	random
1	$0.451 \pm 0.017$	$0.451 \pm 0.017$	$0.451 \pm 0.017$
2	$0.563 \pm 0.017$	$0.57 \pm 0.017$	$0.568 \pm 0.017$
3	$0.654 \pm 0.016$	$0.633 \pm 0.017$	$0.634 \pm 0.017$
4	$0.713 \pm 0.016$	$0.688 \pm 0.016$	$0.688 \pm 0.017$
5	$0.767 \pm 0.014$	$0.742 \pm 0.015$	$0.738 \pm 0.016$
6	$0.805 \pm 0.013$	$0.781 \pm 0.014$	$0.775 \pm 0.015$
7	$0.835 \pm 0.012$	$0.817 \pm 0.013$	$0.809 \pm 0.014$
8	$0.864 \pm 0.011$	$0.843 \pm 0.012$	$0.84 \pm 0.012$

For true utility BO, we consider EUBO, Max Value Entropy Search, and a random acquisition function. For preferential BO, replicating the setup of Lin et al. (2022), we compare EUBO, BALD, and a random acquisition function.

**Results.** We compute the results for the three DTLZ2 environments (L1, piecewise linear, beta products). As in the main results, we let  $B^{\text{exp}} = d$  and  $B^{\text{pf}} = 2$ . Table 2 shows the results averaged across all three environments. Overall, the differences in performance are not substantial among alternatives. Across all methods, we observe a slight advantage of EUBO against the alternatives, hence we have chosen it as a default choice of the acquisition function for the baseline methods.

C.6 LLM’S ACCURACY AND THE FITTED UTILITIES

**LLM’s accuracy.** First, we look at the accuracy of LILLO in generating pairwise preference labels in step 3. of our algorithm. Figure 11 shows the average accuracy at each trial, computed across 16 seeds and the 3 DTLZ2 environments. We find that already at the first iteration, LILLO yields high-fidelity predictions, with an average accuracy of 85%. As more information about the optimization objective is gathered, its accuracy reaches values above 90%.

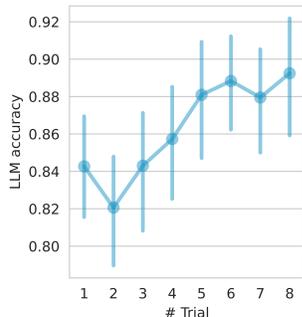


Figure 11: The accuracy of LILLO in generating pairwise preference choices in step 3. of the BO loop.

**Qualitative analysis.** In Figures 13 and 14, we show the plots of fitted utility functions  $M_n^y : \mathcal{Y} \rightarrow \mathbb{R}$  in comparison to the ground-truth utility function  $g : \mathcal{Y} \rightarrow \mathbb{R}$ . The plots are presented for both LILLO and the preferential BO baseline. We generate these plots by fixing the center point of the outcome space  $\mathcal{Y}$  and varying one of the outcome dimensions in its range. The plots show the posterior means of the GP models  $M_n^y$  averaged across 16 replications. We observe that with just a few iterations of the BO loop, LILLO’s fitted utility functions capture the important properties of the ground-truth and qualitatively resemble the shape of the ground-truth much better than the preferential BO baseline. For the piecewise linear utility, the fitted GP’s have the required monotonicity properties. Moreover, the relative range of the fitted utilities across the dimensions also matches the relative contributions of each outcome to the overall utility – an increase in the first outcome corresponds to larger utility gains than an increase in the last outcome. In terms of the L1 function, we observe that the peaks of the utility function match well the ground-truth – the utility is supposed to be maximized at a point closest to a desired optimum  $y^*$ .

C.7 LONGER TRIALS

We note that this paper primarily concerns settings where configurations are very costly to evaluate, and therefore in practice the number of trials is very limited. Our overarching goal is to minimize human effort during optimization, which is why the main experiments focus on the impact of queries after only a few batched rounds. Nevertheless, in this section, we present additional results evaluating the performance of LILLO over longer horizons, with up to 16 batched experimental trials. Figure 12 compares LILLO to both preferential and true utility BO (excluding the two LLM baselines, as they do not show meaningful progression across iterations). We observe that LILLO maintains competitive performance even in this extended setting. As noted in the main results, the advantage of LILLO tends to diminish as the number of experimental trials increases, with quantitative baselines eventually catching up and, in some cases, surpassing LILLO in the long run. This behavior is expected – the in-context learning capabilities of LLMs are inherently limited and do not offer the convergence guarantees provided by conventional baselines operating on quantitative feedback.

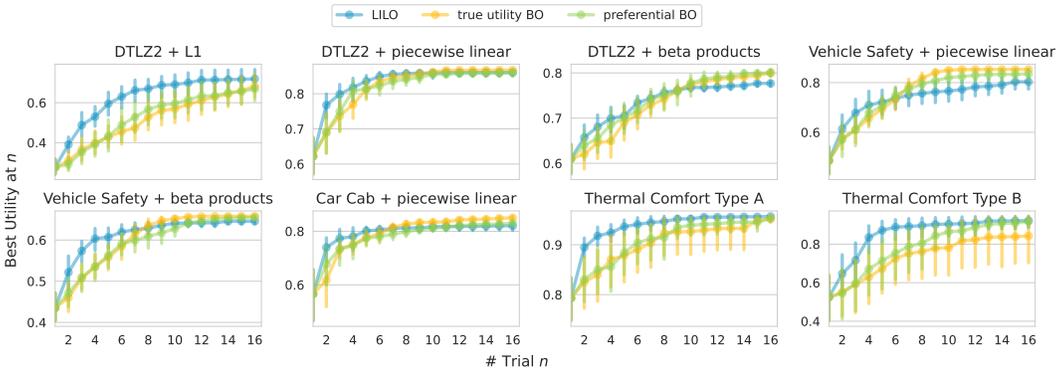


Figure 12: LILLO vs. true utility and preferential BO on longer horizons. Results aggregated across 16 replications per environment.



Figure 13: DTLZ2 + L1: Fitted GP utility models  $M_n^y : \mathcal{Y} \rightarrow \mathbb{R}$  in comparison to the ground-truth utilities.

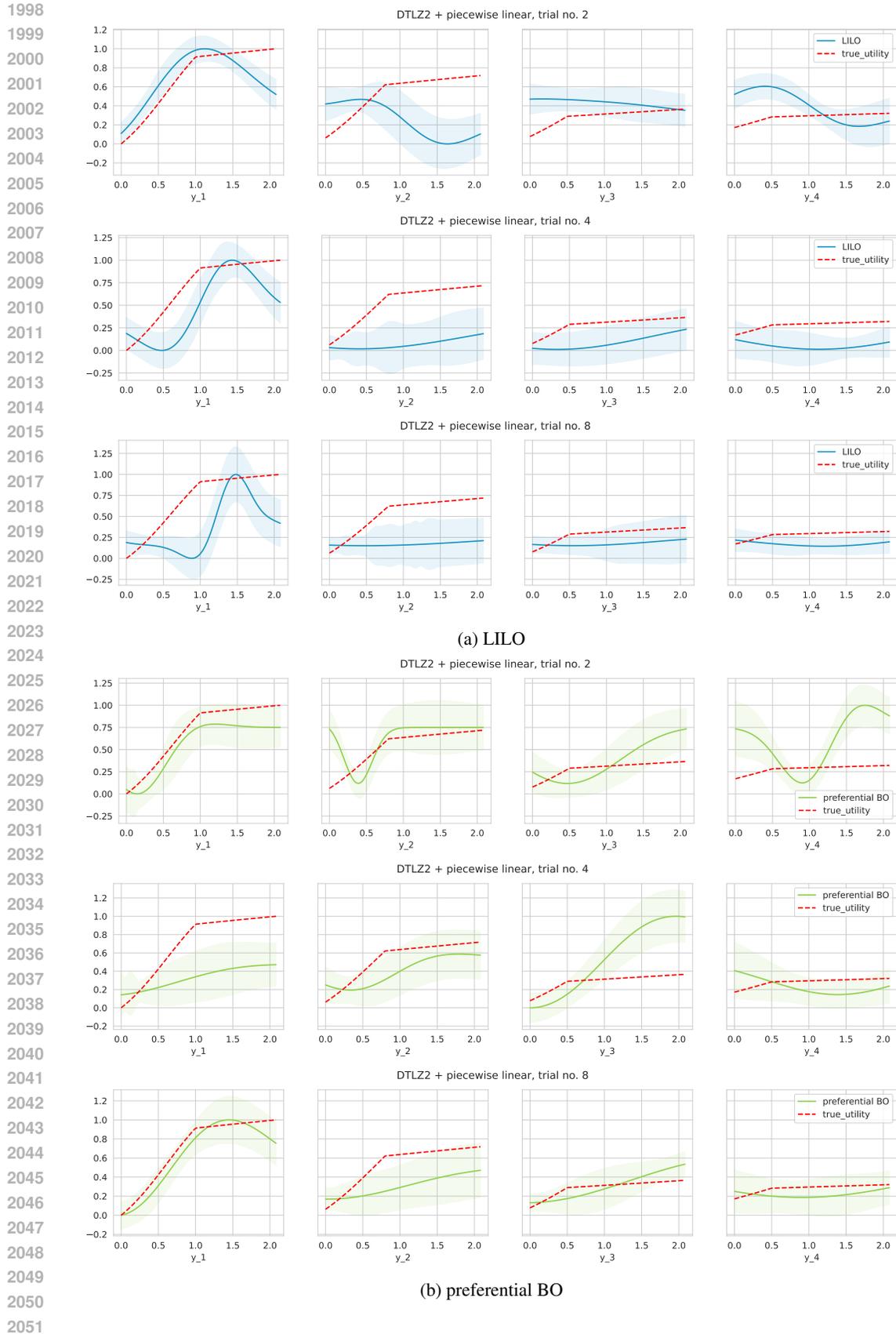


Figure 14: DTLZ2 + piecewise linear: Fitted GP utility models  $M_n^y : \mathcal{Y} \rightarrow \mathbb{R}$  in comparison to the ground-truth utilities.

## C.8 ACQUISITION FUNCTION ABLATION

In our main experiments, we demonstrate the performance of LILLO against baselines with all methods using the (Noisy) Expected Improvement as the acquisition function for candidate generation. However, this choice can be replaced with other alternatives. In this section, we demonstrate that LILLO maintains competitive performance irrespective of this choice.

In Figures 15 and 16 we demonstrate the performance of LILLO and the quantitative baselines with three different choices of acquisition functions for candidate generation: (Noisy) Expected Improvement (as in the main text), Upper Confidence Bound (UCB) with the parameter  $\beta$  controlling the trade-off between exploration and exploitation set to 0.5, and Thompson Sampling. All other elements of LILLO and the baselines stay fixed. Figure 15 shows results summarized across all environments considered and Figure 16 shows a detailed view of the results. We observe that the Expected Improvement and the UCB acquisition functions, overall, perform better than Thompson Sampling on our selected set of test problems. The advantage of LILLO against the baselines is most competitive in these two settings. With Thompson Sampling, LILLO performs similarly to the true utility BO baseline, slightly outperforming it at the very first iterations.

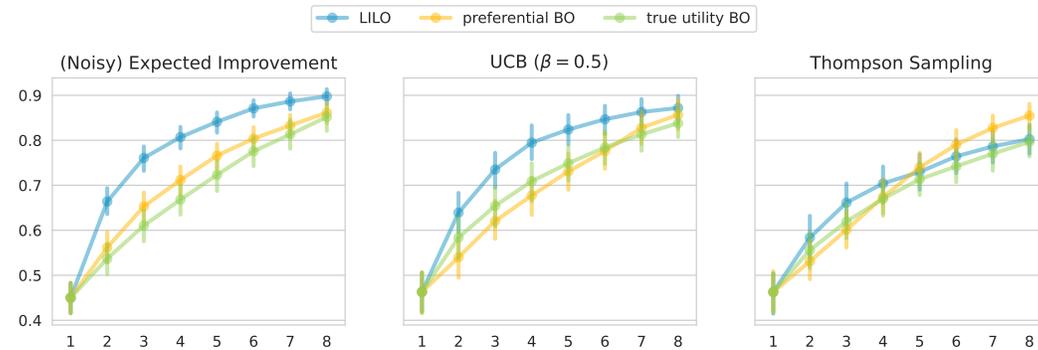
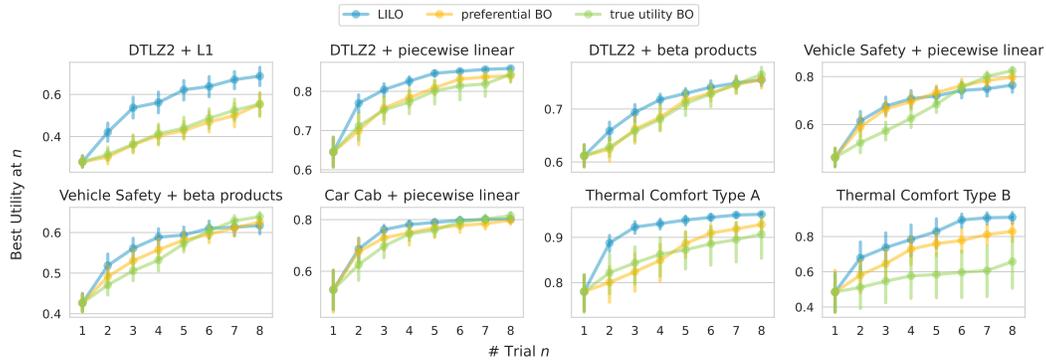
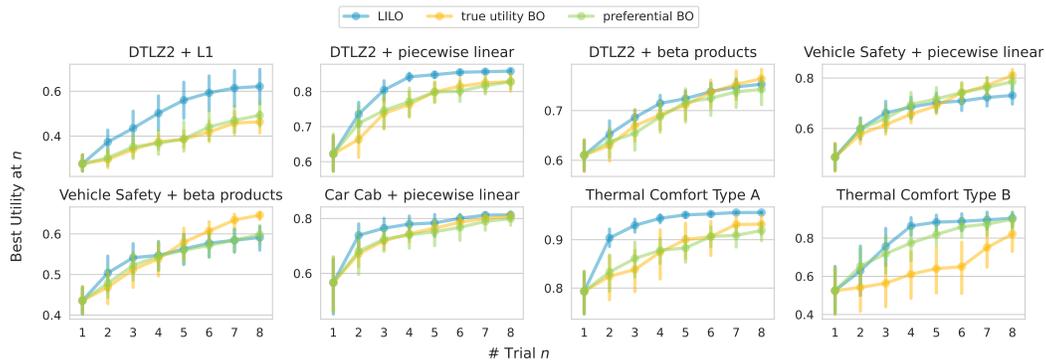


Figure 15: *LILLO vs. quantitative baselines using different acquisition functions for candidate generation.* Results averaged across all 8 environments with 16 replications per environment.

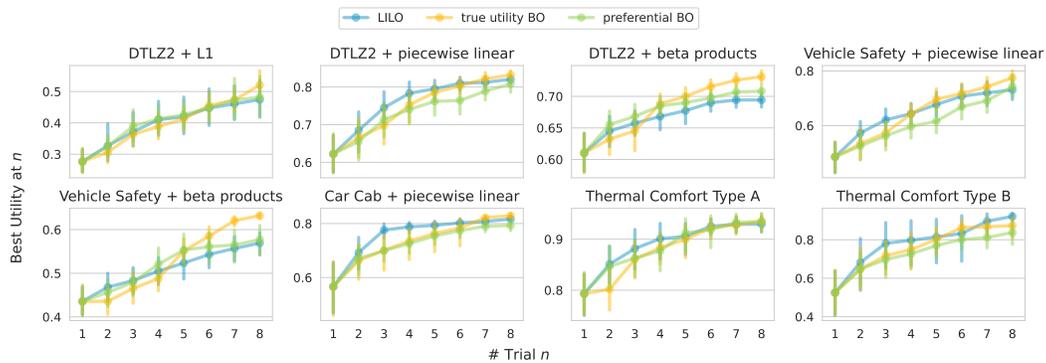
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159



(a) (Noisy) Expected Improvement



(b) UCB ( $\beta = 0.5$ )



(c) Thompson Sampling

Figure 16: *LILO* vs. quantitative baselines using different acquisition functions for candidate generation. Detailed results across all 8 environments with 16 replications per environment.

## D EXAMPLE CONVERSATIONS FROM THE BENCHMARKS

We present example conversations of LILLO with the DM agent to showcase what are the typical questions that our agent asks and what is the form of feedback that it receives. We would like to highlight how for the Thermal Comfort environment the simulated human responses are more qualitative rather than quantitative in nature.

### Example conversation (DTLZ2 + piecewise linear)

- DM: My goal is to achieve the following thresholds in each outcome:  $y_1 \geq 1.0$ ,  $y_2 \geq 0.8$ ,  $y_3 \geq 0.5$ ,  $y_4 \geq 0.5$ . Improvements over the thresholds are always good, but less important than bringing the outcomes to their threshold values. The further away an outcome is from its threshold, the higher is its negative impact on the overall performance.

- LILLO: How do you prioritize improvements in outcomes that are already above their threshold values compared to those below their threshold values?

- DM: I prioritize improvements in outcomes that are below their threshold values over those that are already above, as meeting these thresholds has a significant impact on my overall satisfaction.

- LILLO: Are there any interactions or trade-offs between outcomes that you consider when evaluating overall satisfaction (e.g., is underperforming in one outcome acceptable if others overperform)?

- DM: Yes, I consider interactions and trade-offs between outcomes, taking into account their relative importance to my overall satisfaction, which is not solely determined by their threshold values but by the impact of their improvement on my satisfaction.

- LILLO: Considering outcomes  $\$1_0\$$  and  $\$1_2\$$ , how do you trade off between  $\$y_1\$$  being slightly above its threshold in  $\$1_0\$$  and  $\$y_4\$$  being significantly above its threshold in  $\$1_2\$$ , given that  $\$y_2\$$  and  $\$y_3\$$  are below their thresholds in both cases? Specifically, which outcome do you prefer and why?

- DM: I prefer outcome  $\$1_0\$$  because  $\$y_1\$$  being slightly above its threshold of 1.0 takes precedence due to its higher importance.

- LILLO: Considering outcomes  $\$2_0\$$  and  $\$2_1\$$ , both have  $\$y_1\$$  above its threshold, but  $\$y_2\$$  is below its threshold in both cases. Given that  $\$y_4\$$  in  $\$2_0\$$  is above its threshold and in  $\$2_1\$$  is significantly below, and  $\$y_3\$$  is slightly above its threshold in both, which outcome do you prefer and why?

- DM: I prefer outcome  $\$2_0\$$  because it has  $\$y_4\$$  above its threshold, contributing positively to my overall satisfaction, unlike  $\$2_1\$$  where  $\$y_4\$$  is significantly below its threshold.

Example 2: Example fragments of conversation of LILLO with the DM on the DTLZ2 + piecewise linear environment.

2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
2226  
2227  
2228  
2229  
2230  
2231  
2232  
2233  
2234  
2235  
2236  
2237  
2238  
2239  
2240  
2241  
2242  
2243  
2244  
2245  
2246  
2247  
2248  
2249  
2250  
2251  
2252  
2253  
2254  
2255  
2256  
2257  
2258  
2259  
2260  
2261  
2262  
2263  
2264  
2265  
2266  
2267

#### Thermo + type A

- DM: My goal is to keep all metrics within my thermal comfort preferences.

- LILO: Can you provide more specific thresholds for each metric that you consider to be within your thermal comfort preferences?

- DM: To be honest, I prefer it when the air doesn't feel drafty at my neck, and I don't feel too much warmth coming from the ceiling. I also like it when there's not a big difference in temperature between my head and ankles. As for the floor, I find it comfortable when it's around 22-24 degrees Celsius. If it's too cold or too hot, my feet feel uncomfortable, and that affects my overall comfort.

- LILO: How do you prioritize these metrics relative to one another? For example, is minimizing PPD more important to you than minimizing DR, or are they equally important?

- DM: For me, it's really important that I don't feel drafty or too warm from the ceiling. If I had to choose, I'd say minimizing those feelings is a bit more important than the other factors. But, at the same time, I also value not having a big temperature difference between my head and ankles. It's all about finding a balance, I suppose. If one thing is off, it can really affect how I feel overall.

- LILO: Considering arm\_index 1\_0 (PPD: 25.67%, DR: 13.52%, dT\_vert: 0.24K, dT\_pr: 1.02K, T\_floor: 18.47C) and arm\_index 1\_3 (PPD: 9.02%, DR: 3.27%, dT\_vert: 4.14K, dT\_pr: 17.08K, T\_floor: 25.52C), which one do you prefer and why?

- DM: I prefer arm\_index 1\_3. I feel more comfortable with the conditions in 1\_3, even though the temperature difference between my head and ankles is a bit higher. The air feels calmer and less turbulent, which makes me feel more at ease. In 1\_0, I feel a bit chilly, especially on the floor, which is slightly too cold for my liking. I wish it were a bit warmer, around 20-22C would be perfect.

- LILO: On a scale of 1-5, how satisfied are you with arm\_index 1\_5 (PPD: 57.90%, DR: 0%, dT\_vert: 0.50K, dT\_pr: 19.41K, T\_floor: 27.10C), and would you like to see improvements in any specific metrics?

- DM: I'd rate my satisfaction with arm\_index 1\_5 as a 1 out of 5. I feel quite uncomfortable with these conditions. The air feels stale, and I feel a bit too warm from the ceiling, which is making me feel uneasy. Also, the floor temperature is a bit too high, I prefer it when it's around 20-24C. I'd like to see improvements in reducing the warmth from the ceiling and adjusting the floor temperature to a more comfortable range.

Example 3: Example fragments of conversation of LILO with the DM on the Thermal Comfort + Type A environment.