PropertyDAG: Multi-objective Bayesian optimization of partially ordered, mixed-variable properties for biological sequence design

Ji Won Park[1], **Samuel Stanton**[1], **Saeed Saremi**[1], **Andrew Watkins**[1], **Henri Dwyer**[1], **Vladimir Gligorijević**[1], **Richard Bonneau**[1], **Stephen Ra**[1], and **Kyunghyun Cho**[1,2,3,4]

[1]Prescient Design, Genentech
[2]Department of Computer Science, Courant Institute of Mathematical Sciences, New York University
[3]Center for Data Science, New York University
[4]CIFAR Fellow
park.ji_won@gene.com

Abstract

Bayesian optimization offers a sample-efficient framework for navigating the exploration-exploitation trade-off in the vast design space of biological sequences. Whereas it is possible to optimize the various properties of interest jointly using a multi-objective acquisition function, such as the expected hypervolume improvement (EHVI), this approach does not account for objectives with a hierarchical dependency structure. We consider a common use case where some regions of the Pareto frontier are prioritized over others according to a specified *partial ordering* in the objectives. For instance, when designing antibodies, we maximize the binding affinity to a target antigen only if it can be expressed in live cell culture—modeling the experimental dependency in which affinity can only be measured for antibodies that can be expressed and thus produced in viable quantities. In general, we may want to confer a partial ordering to the properties such that each property is optimized conditioned on its parent properties satisfying some feasibility condition. To this end, we present PropertyDAG, a framework that operates on top of the traditional multi-objective BO to impose a desired partial ordering on the objectives, e.g. expression $\rightarrow$ affinity. We demonstrate its performance over multiple simulated active learning iterations on a penicillin production task, toy numerical problem, and a real-world antibody design task.

1 Introduction

Designing biological sequences entails searching over vast combinatorial design spaces. Recently, deep sequence generation models trained on large datasets of known, functional sequences have shown promise in generating physically and chemically plausible designs [e.g. 1–3]. Whereas these models accelerate the design process, limited resources place a cap on how many designs we can characterize *in vitro* for assessing their suitability. Only once a design is validated *in vitro* and undergoes multiple rounds of optimization can it proceed down the drug development pipeline to preclinical development and clinical trials, where its performance is tested *in vivo*.

Because the wet lab cannot provide feedback on all of the candidate designs, we take an iterative, data-driven approach to select the the most informative subset to submit to the wet lab. Many drug design applications call for such an active learning approach, as the initial datasets available to train predictive models on our desired properties of interest tends to be small or nonexistent. The

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

measurements returned by the lab in each iteration is appended to our training set and we update our models using the augmented dataset for the next iteration.

The wet lab's measurement process can be viewed as a black-box function that is expensive to evaluate. In the context of identifying designs maximizing this function, Bayesian optimization (BO) emerges as a promising, sample-efficient framework that trades off exploration (evaluating highly uncertain designs) and exploitation (evaluating designs believed to carry the best properties) in a principled manner [4]. It relies on a probabilistic surrogate model that infers the posterior distribution over the objectives and an acquisition function that assigns an expected utility value to each candidate. BO has been successfully applied to a variety of protein engineering applications [5–7].

In particular, we cast our problem as multi-objective BO, where multiple objectives are jointly optimized. Our objectives originate from the molecular properties evaluated during *in vitro* validation. This validation process involves producing the design, confirming its pharmacology, and evaluating whether it is active against a given drug target of interest. If found to be potent, the design is then assayed for *developability* attributes—physicochemical properties that characterize the safety, delivery, and manufacturability [8].

The experimental process of *in vitro* validation signifies a hierarchy among the objectives. Consider the property "expression" in the context of antibody design, for instance. A designed antibody candidate must first be expressed in live cell culture. If the level of expression does not meet a fixed threshold, the lab cannot produce it and it cannot be assayed for potency and developability downstream. Supposing now that a design did express in viable amounts, if it does not bind to a target antigen with sufficient "affinity" (and is thus not potent), then the design fails as an antibody and there is little practical incentive in assaying it for developability (such as specificity and thermostability)—even if it is possible to do so. The dependency between properties, whether experimental or biological in origin, motivates us prioritize some objectives before others when selecting the subset of designs to submit to the wet lab. Our primary goal is to identify "joint positive" designs, designs that meet the chosen thresholds in all the parent objectives (expressing binders) according to the specified partial ordering and also perform well in the leaf-level objectives (high specificity, thermostability).

To this end, we propose PropertyDAG, a simple framework that operates on top of the traditional multi-objective BO to impose a desired partial ordering on the objectives, e.g. expression $\rightarrow$ affinity $\rightarrow \{$ specificity, thermostability$\}$. Our framework modifies the posterior inference procedure within standard BO in two ways. First, we treat the objectives as *mixed-variable*—in particular, each objective is modeled as a mixture of zeros and a wide dispersion of real-valued, non-zero values. The surrogate model consists of a binary classifier and a regressor, which infer the zero mode and the non-zero mode, respectively. We show that this modeling choice is well-suited for biological properties, which tend to carry excess zero, or null, values and fat tails [9]. Second, before samples from the posterior distribution inferred by the surrogate model enter the acquisition function, we transform the samples such that they conform to the specified partial ordering of properties. We run multi-objective BO with PropertyDAG over multiple simulated active learning iterations to a penicillin production task, a toy numerical problem, and a real-world antibody design task. In all three tasks, PropertyDAG-BO identifies significantly more joint positives compared to standard BO. After the final iteration, the surrogate models trained under Property-BO also output more accurate predictions on the joint positives in a held-out test set than do the standard BO equivalents.

2 Background

Bayesian optimization (BO) is a popular technique for sample-efficient black-box optimization [see 10, 11, for a review]. Suppose our objective $f : \mathcal{X} \rightarrow \mathbb{R}$ is a black-box function of the design space $\mathcal{X}$ that is expensive to evaluate. Our goal is to efficiently identify a design $\boldsymbol{x}^\star \in \mathcal{X}$ maximizing[1] f. BO leverages two tools, a probabilistic surrogate model and a utility function, to trade off exploration (evaluating highly uncertain designs) and exploitation (evaluating designs believed to maximize f) in a principled manner.

For each iteration $t \in \mathbb{N}$, we have a dataset $\mathcal{D}_t = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \cdots, (\boldsymbol{x}^{(N_t)}, y^{(N_t)})\} \in \mathscr{D}_t$, where each $y^{(n)}$ is a noisy observation of f. First, the probabilistic model $\hat{f} : \mathcal{X} \rightarrow \mathbb{R}$ infers the posterior

[1] For simplicity, we define the task as maximization in this paper without loss of generality. For minimizing f, we can negate f, for instance.

distribution $p(\hat{f}|\mathcal{D}_t)$, quantifying the plausibility of surrogate objectives $\hat{f} \in \mathcal{F}$. Next, we introduce a utility function $u : \mathcal{X} \times \mathcal{F} \times \mathscr{D}_t :\to \mathbb{R}$. The acquisition function $a(\boldsymbol{x})$ is simply the expected utility of $\boldsymbol{x}$ w.r.t. our beliefs about f,

$$a(\boldsymbol{x}) = \int u(\boldsymbol{x}, \hat{f}, \mathcal{D}_t) p(\hat{f}|\mathcal{D}_t) d\hat{f}. \tag{1}$$

For example, we obtain the expected improvement (EI) acquisition function if we take $u_{\mathrm{EI}}(\boldsymbol{x}, \hat{f}, \mathcal{D}) = [\hat{f}(\boldsymbol{x}) - \max_{(\boldsymbol{x}', y') \in \mathcal{D}} y']_+$, where $[\cdot]_+ = \max(\cdot, 0)$ [12, 4]. Generally the integral is approximated by Monte Carlo (MC) with posterior samples $\hat{f}^{(j)} \sim p(\hat{f}|\mathcal{D}_t)$. We select a maximizer of a as the new design, measure its properties, and append the observation to the dataset. The surrogate is then retrained on the augmented dataset and the procedure repeats.

Multi-objective optimization When there are multiple objectives of interest, a single best design may not exist. Suppose there are K objectives, $f : \mathcal{X} \to \mathbb{R}^K$. The goal of multi-objective optimization (MOO) is to identify the set of *Pareto-optimal* solutions such that improving one objective within the set leads to worsening another. We say that $\boldsymbol{x}$ dominates $\boldsymbol{x}'$, or $f(\boldsymbol{x}) \succ f(\boldsymbol{x}')$, if $f_k(\boldsymbol{x}) \geq f_k(\boldsymbol{x}')$ for all $k \in \{1, \ldots, K\}$ and $f_k(\boldsymbol{x}) > f_k(\boldsymbol{x}')$ for some k. The set of *non-dominated* solutions $\mathscr{X}^*$ is defined in terms of the Pareto frontier (PF) $\mathcal{P}^*$,

$$\mathscr{X}^\star = \{\boldsymbol{x} : f(\boldsymbol{x}) \in \mathcal{P}^\star\}, \quad \text{where } \mathcal{P}^\star = \{f(\boldsymbol{x}) \,:\, \boldsymbol{x} \in \mathcal{X},\ \nexists\, \boldsymbol{x}' \in \mathcal{X} \text{ s.t. } f(\boldsymbol{x}') \succ f(\boldsymbol{x})\}. \tag{2}$$

MOO algorithms typically aim to identify a finite approximation to $\mathscr{X}^\star$, which may be infinite, within a reasonable number of iterations. One way to measure the quality of an approximate PF $\mathcal{P}$ is to compute the hypervolume $\mathrm{HV}(\mathcal{P}|\boldsymbol{r}_{\mathrm{ref}})$ of the polytope bounded by $\mathcal{P} \cup \{\boldsymbol{r}_{\mathrm{ref}}\}$, where $\boldsymbol{r}_{\mathrm{ref}} \in \mathbb{R}^K$ is a user-specified *reference point*. We obtain the expected hypervolume improvement (EHVI) acquisition function if we take

$$u_{\mathrm{EHVI}}(\boldsymbol{x}, \hat{f}, \mathcal{D}) = \mathrm{HVI}(\mathcal{P}', \mathcal{P}|\boldsymbol{r}_{\mathrm{ref}}) = [\mathrm{HV}(\mathcal{P}'|\boldsymbol{r}_{\mathrm{ref}}) - \mathrm{HV}(\mathcal{P}|\boldsymbol{r}_{\mathrm{ref}})]_+, \tag{3}$$

where $\mathcal{P}' = \mathcal{P} \cup \{\hat{f}(\boldsymbol{x})\}$ [13, 14].

Noisy observations In the noiseless setting, the observed baseline PF is the true baseline PF, i.e. $\mathcal{P}_t = \{\boldsymbol{y} : \boldsymbol{y} \in \mathcal{Y}_t,\ \nexists\, \boldsymbol{y}' \in \mathcal{Y}_t \text{ s.t. } \boldsymbol{y}' \succ \boldsymbol{y}\}$ where $\mathcal{Y}_t \coloneqq \{\boldsymbol{y}^{(n)}\}_{n=1}^{N_t}$. This does not, however, hold in many practical applications, where measurements carry noise. For instance, given a zero-mean Gaussian measurement process with noise covariance Σ, the feedback for a design $\boldsymbol{x}$ is $\boldsymbol{y} \sim \mathcal{N}(f(\boldsymbol{x}), \Sigma)$, not $f(\boldsymbol{x})$ itself. The *noisy* expected hypervolume improvement (NEHVI) acquisition function marginalizes over the surrogate posterior at the previously observed points $\mathcal{X}_t = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N_t}$,

$$u_{\mathrm{NEHVI}}(\boldsymbol{x}, \hat{f}, \mathcal{D}) = \mathrm{HVI}(\hat{\mathcal{P}}'_t, \hat{\mathcal{P}}_t|\boldsymbol{r}_{\mathrm{ref}}), \tag{4}$$

where $\hat{\mathcal{P}}_t = \{\hat{f}(\boldsymbol{x}) \,:\, \boldsymbol{x} \in \mathcal{X}_t,\ \nexists\, \boldsymbol{x}' \in \mathcal{X}_t \text{ s.t. } \hat{f}(\boldsymbol{x}') \succ \hat{f}(\boldsymbol{x})\}$ and $\hat{\mathcal{P}}' = \hat{\mathcal{P}} \cup \{\hat{f}(\boldsymbol{x})\}$ [15].

3 Related Work

Existing work on multi-objective BO does not account for objectives with a hierarchical dependency structure [16–19, 15]. We refer to [20] for a formulation of single-objective BO with a hierarchy in how the objective is computed. A body of work focuses on constrained optimization, which optimizes a black-box function subject to a set of black-box constraints being satisfied [21–26]. For dealing with mixed-variable objectives, [27] propose reparameterizing the discrete random variables in terms of continuous parameters. Our approach here is to model them explicitly using the zero-inflated formalism.

4 Method

Overview Figure 1 illustrates our proposed PropertyDAG-BO framework alongside standard BO. The candidate generation step is identical in both cases; we first sample a large pool of design candidates from a proposal distribution, often implemented as a generative model. The difference lies

in the next step, in which the designs are scored by probabilistic surrogate models. In PropertyDAG-BO, we must first specify a partial ordering of our objectives (Section 4.1). We then assign a zero-inflated generative model for each objective (Section 4.2) such that a probabilistic classifier models its "zero" mode and a probabilistic regressor models the remaining continuous-valued "non-zero" mode. The raw posterior samples from the surrogates then undergo a "resampling" step (Section 4.3) that enforces the specified PropertyDAG. Finally, the modified posterior samples enter the multi-objective acquisition function, which scores the design candidates just as in standard BO.

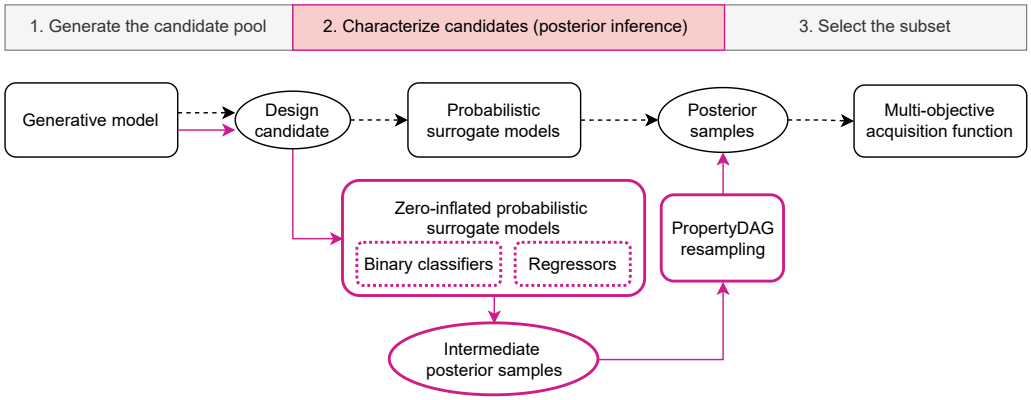


Figure 1: Comparison of a traditional multi-objective BO pipeline (dashed black arrows) with the proposed pipeline enabled by PropertyDAG (solid magenta arrows).

4.1 Defining a PropertyDAG

Many drug design applications motivate us to prescribe some hierarchy among our objective properties of interest. The partial ordering may arise from an experimental dependency, e.g. a design candidate must pass a certain threshold in one property for its other properties to be measured. In the context of antibody design, a design candidate is a sequence of amino acids representing an antibody that must first be expressed in cell culture. If the level of expression does not exceed some threshold in mass per volume, the lab cannot produce it in viable amounts and it cannot be assayed for other properties, such as binding affinity to a target antigen. Our PropertyDAG may then take the form: expression $\rightarrow$ affinity. Experimental dependencies like this creates an asymmetry among the objectives; it reduces the information content of designs that do not express much more than that of designs that do not bind, because non-expressing designs cannot provide binding measurements.

Alternatively, the partial ordering may encode our preference for the types of designs we want to obtain. We may prioritize a property, for instance, so that we reject designs that performs poorly in this property, no matter how well they perform in all the others. If a designed antibody does not bind to the target antigen, it has failed in its primary function, so we have little interest in its developability properties, such as specificity to the target antigen and thermostability, even though, unlike for non-expressers, they often remain measurable. We then impose the following PropertyDAG: expression $\rightarrow$ affinity $\rightarrow$ { specificity, thermostability }.

A PropertyDAG can be expressed as ordered sets of properties: $\{y_{0,0}, \ldots, y_{0,M_0}\} \rightarrow \{y_{1,0}, \ldots, y_{1,M_1}\} \rightarrow \cdots \rightarrow \{y_{L,0}, \ldots, y_{L,M_L}\}$, where $y_{l,m}$ is the property at level $l \in \{0, \ldots, L-1\}$ of the hierarchy and $m \in \{0, \ldots, M_l - 1\}$ is its index among the M_l sibling properties at the same level l. Figure 2 shows an example of a PropertyDAG with three properties and two levels.

4.2 Zero-inflated modeling

Biological properties tend to carry excess zeros, or null values. Their zero-inflated nature motivates us to employ statistical models that account for large incidences of zeros [28]. The zero-inflated negative binomial (ZINB) distribution has been applied to model discrete counts in single-cell RNA-seq data [29]. For each objective y_k, our zero-inflated model assigns a binary random variable $b_k \in \{0, 1\}$ to generate the zeros and $r_k \in \mathbb{R}$ to generate the remaining dispersion of continuous non-zero values.

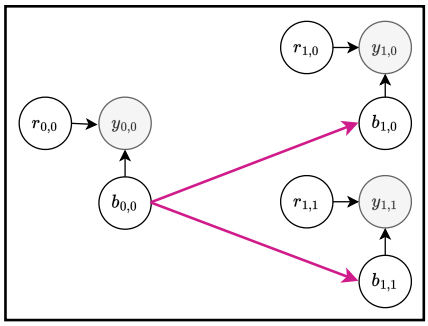


Figure 2: Example of a PropertyDAG, with three properties and two levels of hierarchy. Each property is denoted $y_{l,m}$, where l indexes the level and m indexes sibling properties at the same level. Magenta arrows refer to the dependencies imposed by PropertyDAG. Black arrows make it explicit that each $y_{l,m}$ is modeled as zero-inflated, where $b_{l,m}$ governs the zero events and $r_{l,m}$ governs the continuous non-zero events.

Assume f is non-negative (equivalently, that it is bounded from below). Given a training dataset $\mathcal{D}_t$ available at time t, we decompose $p(\hat{f}_k \mid \mathcal{D}_t)$ as follows:

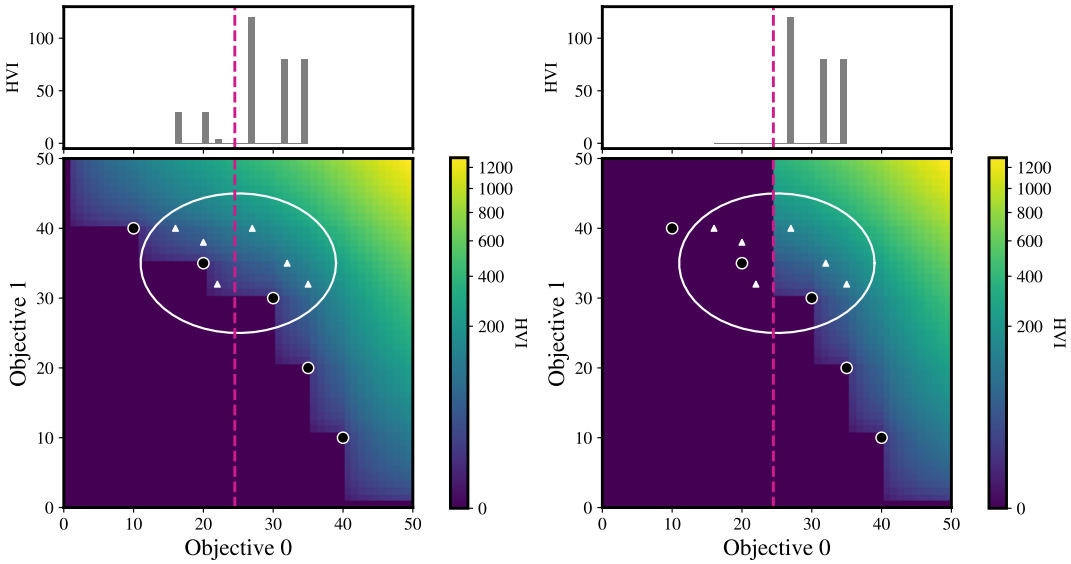
$$p(\hat{f}_k(\boldsymbol{x}) = c \mid \mathcal{D}_t) = \begin{cases} p(b_k = 0 \mid \boldsymbol{x}, \mathcal{D}_t) & \text{if } c = 0, \\ p(b_k = 1 \mid \boldsymbol{x}, \mathcal{D}_t)\, p(r_k = c \mid \boldsymbol{x}, \mathcal{D}_t, \theta_r) & \text{else,} \end{cases} \tag{5}$$

where $p(r_k \mid \boldsymbol{x}, \mathcal{D}_t, \theta_r)$ is a regressor parameterized by θ_r trained to predict f_k given $b_k = 1$. For simplicity we have assumed $p(r_k = 0 | \boldsymbol{x}, \mathcal{D}_t, \theta_r) = 0$. Since Gaussian processes (GPs) are often used as surrogates for BO [see, e.g. 30, for a review on GPs] and common GP assumptions fail for sparse, multi-modal data, separating out the non-zero mode of the data can improve posterior inference.

To accommodate objective hierarchy, each $p(b_k \mid \boldsymbol{x}, \mathcal{D}_t)$ decomposes further as

$$p(b_k \mid \boldsymbol{x}, \mathcal{D}_t) = \begin{cases} 0 & \text{if } \exists j \in \mathrm{pred}(k) \text{ s.t. } b_j = 0, \\ p(b_k \mid \boldsymbol{x}, \mathcal{D}_t, \theta_b) & \text{else,} \end{cases} \tag{6}$$

where $p(b_k \mid \boldsymbol{x}, \mathcal{D}_t, \theta_b)$ is a classifier parameterized by θ_b trained to predict $\mathbb{1}\{f_k(\boldsymbol{x}) > 0\}$ and $\mathrm{pred}(k)$ are the predecessors, or ancestral nodes, in the PropertyDAG corresponding to property k.



(a) Default

(b) PropertyDAG resampling

Figure 3: Effect of resampling the surrogate posteriors on the acquisition function. Suppose the magenta dashed lines represent our threshold for Objective 0 such that we want to acquire candidates that maximize Objective 1 *and* exceed this threshold in Objective 0. *Bottom:* Given our baseline points (black dots), the colors of the grid indicate HVI (Equation 3) computed from a posterior sample at each location in the objective space, in the (a) default and (b) PropertyDAG settings. Consider six samples (white triangles) from the posterior (white contour). PropertyDAG transforms the posterior samples below the threshold in Objective 0 such that their HVI contribution is zero. *Top:* The HVI corresponding to each sample before (a) and after (b) the resampling.

This general framework, presented in terms of a zero-inflated, continuous-valued objective (a mixture of a delta function at zero and a continuous distribution), applies to binary-valued objectives and continuous-valued objectives without zero inflation, which can be viewed as specific cases taking $p(r_k|\boldsymbol{x}, \mathcal{D}_t, \theta_r) = p(r_k) = \mathcal{N}(0, \sigma^2)$ with very small σ and $p(b_k = 1 \mid \boldsymbol{x}, \mathcal{D}_t, \theta_b) = 1$, respectively.

4.3 Resampling

Using a simple resampling trick, we modify the posterior samples from the surrogate models to enforce the parent-child relationships specified in PropertyDAG. As before, consider a property y_k and refer to its predecessors, or ancestral nodes, as $\mathrm{pred}(k)$. Suppose we have drawn single a posterior sample and obtained $\beta_{k'} \sim p(b_{k'}|\boldsymbol{x}) \in \{0, 1\}$ and $\rho_{k'} \sim p(r_{k'}|\boldsymbol{x}) \in \mathbb{R}$ for each $k' \in \{1, \dots, K\}$. Without any modification, Equation 5 would yield the following sample γ_k of y_k:

$$\gamma_k = \begin{cases} 0 & \text{if } b_k = 0 \\ \rho_k & \text{if } b_k = 1. \end{cases} \tag{7}$$

Instead, we begin at the top level and proceed down the levels of PropertyDAG to impose dependencies between b_k and its predecessor properties $\{b_{k'}\}_{k' \in \mathrm{pred}(k)}$. If y_k is a top-level property, then $\mathrm{pred}(k) = \emptyset$ and $\hat{\beta}_k = \beta_k$. Otherwise, y_k has parent properties and we have

$$\hat{\beta}_k = \begin{cases} 0 & \text{if } \exists j \in \mathrm{pred}(k) \text{ s.t. } b_j = 0 \\ \beta_k & \text{else.} \end{cases} \tag{8}$$

We can then use the modified binary samples $\{\hat{\beta}_k\}_{k=1}^K$ to obtain our effective sample $\hat{\gamma}_k$ of y_k:

$$\hat{\gamma}_k = \begin{cases} 0 & \text{if } \beta_k = 0 \text{ or } \exists j \in \mathrm{pred}(k) \text{ s.t. } b_j = 0 \\ \rho_k & \text{else.} \end{cases} \tag{9}$$

Let $\boldsymbol{\gamma} := [\gamma_1, \dots, \gamma_K] \in \mathbb{R}^K$ and $\hat{\boldsymbol{\gamma}} := [\hat{\gamma}_1, \dots, \hat{\gamma}_K] \in \mathbb{R}^K$ and denote the transformation at the sample level described in Equations 8 and 9 as $h : \mathbb{R}^K \to \mathbb{R}^K$ such that $h(\boldsymbol{\gamma}) = \hat{\boldsymbol{\gamma}}$.

We repeat this resampling procedure to other posterior samples. The transformed samples are then used to evaluate NEHVI (Equation 4) via MC integration. More precisely, suppose we draw S posterior samples in parallel for a design candidate $\boldsymbol{x}^*$ (reflecting both aleatoric and epistemic uncertainties) and the previously observed designs $\mathcal{X}_t = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N_t}$ (reflecting the aleatoric uncertainty) and denote each draw as $\boldsymbol{\gamma}_s^*$ and $G_s := \{\boldsymbol{\gamma}_s^{(n)}\}_{n=1}^{N_t}$, respectively, for $s = 1, \dots, S$. Then the MC approximation of NEHVI can be efficiently evaluated as:

$$\begin{aligned} &\hat{\boldsymbol{\gamma}}_s^* = h(\boldsymbol{\gamma}_s^*),\ \hat{\boldsymbol{\gamma}}_s^{(n)} = h(\boldsymbol{\gamma}_s^{(n)}) \quad \forall s = 1, \dots, S,\ \forall n = 1, \dots, N_t \\ &\alpha_{\mathrm{NEHVI}}(\boldsymbol{x}^*) \approx \frac{1}{S} \sum_{s=1}^{S} \mathrm{HVI}\left(\mathcal{P}_t^{*[s]}, \mathcal{P}_t^{[s]} | \boldsymbol{r}_{\mathrm{ref}}\right), \end{aligned} \tag{10}$$

where $\mathcal{P}_t^{[s]} = \{\boldsymbol{\gamma}_s : \boldsymbol{\gamma}_s \in G_s,\ \nexists\, \boldsymbol{\gamma}_s' \in G_s \text{ s.t. } \boldsymbol{\gamma}_s' \succ \boldsymbol{\gamma}_s\}$ and $\mathcal{P}_t^{*[s]} = \mathcal{P}_t^{[s]} \cup \{\hat{\boldsymbol{\gamma}}_s^*\}$.

Figures 3a and 3b illustrate the effect of the PropertyDAG resampling on each posterior sample's HVI contribution to EHVI.

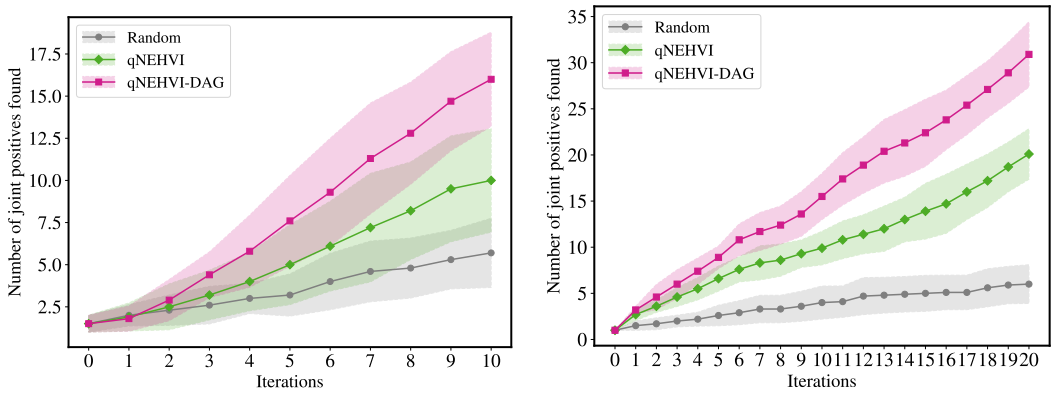
5 Experiments

We perform simulated active learning experiments on two synthetic tasks and a real-world antibody design task. We use NEHVI (Equation 4) as our acquisition function and evaluate it via MC integration (Equation 10). In each experiment, we test three types of acquisitions: (1) batched, multi-objective BO with PropertyDAG ("qNEHVI-DAG"), (2) without PropertyDAG ("qNEHVI"), and (3) random. Our main metric is the number of acquired "joint positive" designs, designs that exceed the chosen thresholds in all objectives according to the specified PropertyDAG. We refer to the batch size as q.

5.1 Penicillin production dataset

This task is based on the penicillin production simulator [31]. We defined the goal as minimizing the CO_2 byproduct emission while ensuring that the fermentation time is below a set threshold and the yield exceeds a set threshold ($K = 3, \mathcal{X} = \mathbb{R}^7$). We negated the latter two objectives to define a maximization problem and assume the PropertyDAG: $\{y_{0,0}\} \rightarrow \{y_{1,0}\} \rightarrow \{y_{2,0}\}$, where $y_{0,0}$ = Yield ("Objective 0"), $y_{1,0}$ = Negative fermentation time ("Objective 1"), and $y_{2,0}$ = Negative CO_2 byproduct ("Objective 2"). Zero-mean Gaussian noise was added to the input, following [19]. We fit an exact GP to model r_k and an approximate GP with the variational evidence lower bound (ELBO) to model b_k, separately for each Objective k. We drew 512 posterior samples to evaluate qNEHVI.

We executed 10 rounds of simulated active learning by initializing the surrogates with 8 training points and selecting $q = 4$ out of 80 randomly-sampled pool of candidate points in iteration. The three acquisition modes (qNEHVI-DAG, qNEHVI, and Random) were subject to the same initial training points and candidate pool each round. The entire experiment was repeated 5 times. Figure 4a shows that qNEHVI-DAG identifies significantly more joint positives than do qNEHVI and Random over active learning iterations.



(a) Penicillin production

(b) Branin-Currin toy problem

Figure 4: Number of joint positives over simulated active learning iterations. Error bands are standard deviations over (a) five and (b) ten trials.

5.2 Branin-Currin toy problem

This task is based on an analytic Branin-Currin test function from [19] with $\mathcal{X} = \mathbb{R}^2$ and $K = 2$. We reformulated this task to simulate the antibody design task (Section 5.3) in a controlled environment. We defined the PropertyDAG, $\{y_{0,0}\} \rightarrow \{y_{1,0}\}$, where $y_{0,0}$ = Dimension 0 ("Objective 0") and $y_{1,0}$ = Dimension 1 ("Objective 1"). Objective 0 was transformed into binary values using a set threshold. Objective 1 was zero-inflated and real-valued. Posterior inference was performed following the same procedure described in Section 5.1.

We executed 20 rounds of simulated active learning by initializing the surrogates with 6 training points and selecting $q = 4$ out of 40 randomly-sampled pool of candidate points in iteration. The entire experiment was repeated 10 times. Figure 4b shows that qNEHVI-DAG identifies significantly more joint positives than do qNEHVI and Random over active learning iterations. Figure 5 compares the qNEHVI-DAG, qNEHVI, and Random selections for Objective 1, for the final selections stacked across the 10 repeated trials. Overall, qNEHVI-DAG identifies more examples to the right of the threshold (black dashed lines) than do qNEHVI and Random, and the improvement is more pronounced for the identification of joint positives (middle panel).

5.3 Antibody design

The antibody design task is derived from real-world dataset of antibody sequences and their measured *in vitro* properties, affinity and expression. As in the toy problem (Section 5.2), we defined the PropertyDAG, $\{y_{0,0}\} \rightarrow \{y_{1,0}\}$, where $y_{0,0}$ = Expression ("Objective 0") and $y_{1,0}$ = Affinity

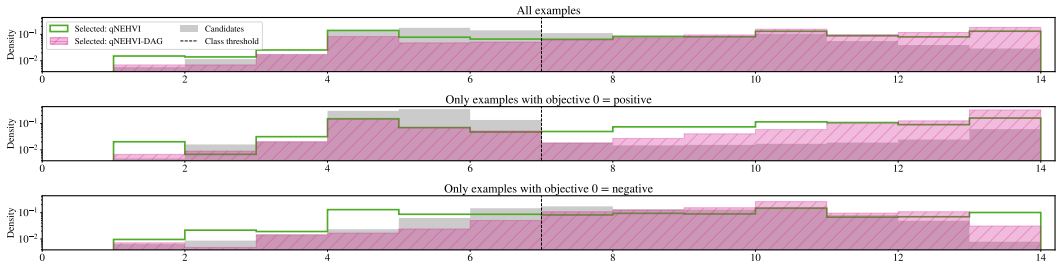
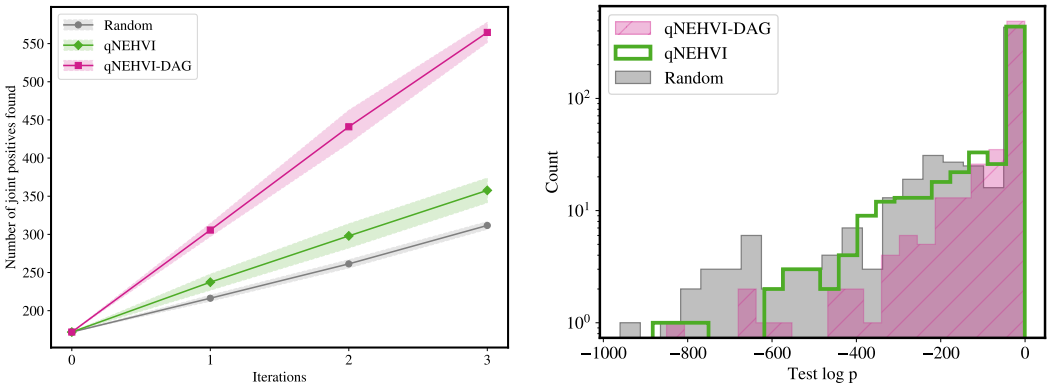


Figure 5: Distribution of final selected candidates in the Branin-Currin problem. We stack the selections over 10 repeated trials. *Top*: all selections. *Middle*: selections for which Objective 0 = 1. *Bottom*: selections for which Objective 0 = 0.

("Objective 1"). Objective 0 was binary-valued, i.e. expressing or not. Objective 1 was zero-inflated and real-valued. We executed 3 iterations of simulated active learning and repeated the entire procedure 5 times. To simulate active learning, we split the entire dataset of 4,023 variable-length protein sequences, designed as antibodies for an anonymized target antigen A, into 5 groups of sizes 1230, 736, 746, 711, and 600. The first group served as the initial training set for the surrogates, the following three groups as the "candidate pools" from which we selected 200 candidates in each iteration, and the last served as a held-out test set. As shown in Figure 6a, qNEHVI-DAG once again outperforms qNEHVI and Random in the number of joint positives. The log posterior density evaluated at the affinity measurements for the joint positives (expressing binders) in the test set is also highest for qNEHVI-DAG, which indicates that the surrogate models from qNEHVI-DAG had the most accurate beliefs about the joint positives after the final iteration.



(a) Joint positives identified

(b) Log posterior density for joint positives

Figure 6: (a) Number of joint positives (expressing binders) for the antibody design problem. Error bands are standard deviations over three data splits for qNEHVI-DAG and qNEHVI and additionally over five repeated trials for each data split for Random. (b) Log posterior density on affinity evaluated at the lab affinity measurements ($\log p$) for the test set, averaged over five data splits.

6 Conclusion

Our proposed method, PropertyDAG, sits on top of the existing multi-objective BO framework to make it amenable to a common scenario in drug design, where a hierarchical structure, or partial ordering, exists among the objectives. It modifies the surrogate posteriors so that each objective is modeled as zero-inflated (a mixture of excess zeros and a continuous distribution) and parent properties in PropertyDAG are prioritized before the children. Empirical evaluations shows that PropertyDAG-BO can identify significantly more designs that are jointly positive (i.e. exceeding a chosen threshold in all properties) than does standard BO. By encapsulating our experimental and biological priors on the relationship between molecular properties, our method promises to accelerate computational drug discovery.

References

[1] Surojit Biswas, Grigory Khimulya, Ethan C Alley, Kevin M Esvelt, and George M Church. Low-n protein engineering with data-efficient deep learning. *Nature methods*, 18(4):389–396, 2021.

[2] Payel Das, Tom Sercu, Kahini Wadhawan, Inkit Padhi, Sebastian Gehrmann, Flaviu Cipcigan, Vijil Chenthamarakshan, Hendrik Strobelt, Cicero Dos Santos, Pin-Yu Chen, et al. Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations. *Nature Biomedical Engineering*, 5(6):613–623, 2021.

[3] Vladimir Gligorijevic, Daniel Berenberg, Stephen Ra, Andrew Watkins, Simon Kelow, Kyunghyun Cho, and Richard Bonneau. Function-guided protein design by deep manifold sampling. *bioRxiv*, 2021.

[4] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[5] Edward O Pyzer-Knapp. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development*, 62(6):2–1, 2018.

[6] Hugo Bellamy, Abbi Abdel Rehim, Oghenejokpeme I Orhobor, and Ross King. Batched bayesian optimization for drug design in noisy environments. *Journal of Chemical Information and Modeling*, 2022.

[7] Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating bayesian optimization for biological sequence design with denoising autoencoders. *arXiv preprint arXiv:2203.12742*, 2022.

[8] Alexander Jarasch, Hans Koll, Joerg T Regula, Martin Bader, Apollon Papadimitriou, and Hubert Kettenberger. Developability assessment during the selection of novel therapeutic antibodies. *Journal of pharmaceutical sciences*, 104(6):1885–1898, 2015.

[9] Tushar Jain, Tingwan Sun, Stéphanie Durand, Amy Hall, Nga Rewa Houston, Juergen H Nett, Beth Sharkey, Beata Bobrowicz, Isabelle Caffry, Yao Yu, et al. Biophysical properties of the clinical-stage antibody landscape. *Proceedings of the National Academy of Sciences*, 114(5):944–949, 2017.

[10] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[11] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[12] Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

[13] Michael Emmerich. *Single-and multi-objective evolutionary design optimization assisted by gaussian random field metamodels*. PhD thesis, Dortmund, Univ., Diss., 2005, 2005.

[14] Michael TM Emmerich, André H Deutz, and Jan Willem Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 2147–2154. IEEE, 2011.

[15] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in Neural Information Processing Systems*, 34:2187–2200, 2021.

[16] Michael A Gelbart, Jasper Snoek, and Ryan P Adams. Bayesian optimization with unknown constraints. *arXiv preprint arXiv:1403.5607*, 2014.

[17] Takashi Wada and Hideitsu Hino. Bayesian optimization for multi-objective optimization and multi-point search. *arXiv preprint arXiv:1905.02370*, 2019.

[18] Kaifeng Yang, Pramudita Satria Palar, Michael Emmerich, Koji Shimoyama, and Thomas Bäck. A multi-point mechanism of expected hypervolume improvement for parallel multi-objective bayesian global optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 656–663, 2019.

[19] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. *Advances in Neural Information Processing Systems*, 33:9851–9864, 2020.

[20] Raul Astudillo and Peter Frazier. Bayesian optimization of function networks. *Advances in Neural Information Processing Systems*, 34:14463–14475, 2021.

[21] Jian Wu and Peter Frazier. The parallel knowledge gradient method for batch bayesian optimization. *Advances in neural information processing systems*, 29, 2016.

[22] Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *ICML*, volume 2014, pages 937–945, 2014.

[23] José Miguel Hernández-Lobato, Michael A Gelbart, Ryan P Adams, Matthew W Hoffman, and Zoubin Ghahramani. A general framework for constrained bayesian optimization using information-based search. 2016.

[24] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to parallelize optimization. In *Computational intelligence in expensive optimization problems*, pages 131–162. Springer, 2010.

[25] Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

[26] Gustavo Malkomes, Bolong Cheng, Eric H Lee, and Mike Mccourt. Beyond the pareto efficient frontier: Constraint active search for multiobjective experimental design. In *International Conference on Machine Learning*, pages 7423–7434. PMLR, 2021.

[27] Samuel Daulton, Xingchen Wan, David Eriksson, Maximilian Balandat, Michael A Osborne, and Eytan Bakshy. Bayesian optimization over discrete and mixed spaces via probabilistic reparameterization. In *ICML2022 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2022.

[28] Julia Eggers. On statistical methods for zero-inflated models, 2015.

[29] Dominic Grün, Lennart Kester, and Alexander Van Oudenaarden. Validation of noise models for single-cell transcriptomics. *Nature methods*, 11(6):637–640, 2014.

[30] CE Rasmussen and CKI Williams. Gaussian processes for machine learning (adaptive computation and machine learning), 2005.

[31] Qiaohao Liang and Lipeng Lai. Scalable bayesian optimization accelerates process optimization of penicillin production. In *NeurIPS 2021 AI for Science Workshop*, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

(b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will release the material after the work is accepted by a peer-reviewed journal for publication.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Additional results for the penicillin production task

Figure 7 compares the qNEHVI and qNEHVI-DAG selections for every pair of objectives, for the final (after Iteration 10) selections stacked across the 5 repeated trials. For every objective, qNEHVI-DAG identifies more examples to the right of the threshold (black dashed lines) than do qNEHVI and Random.

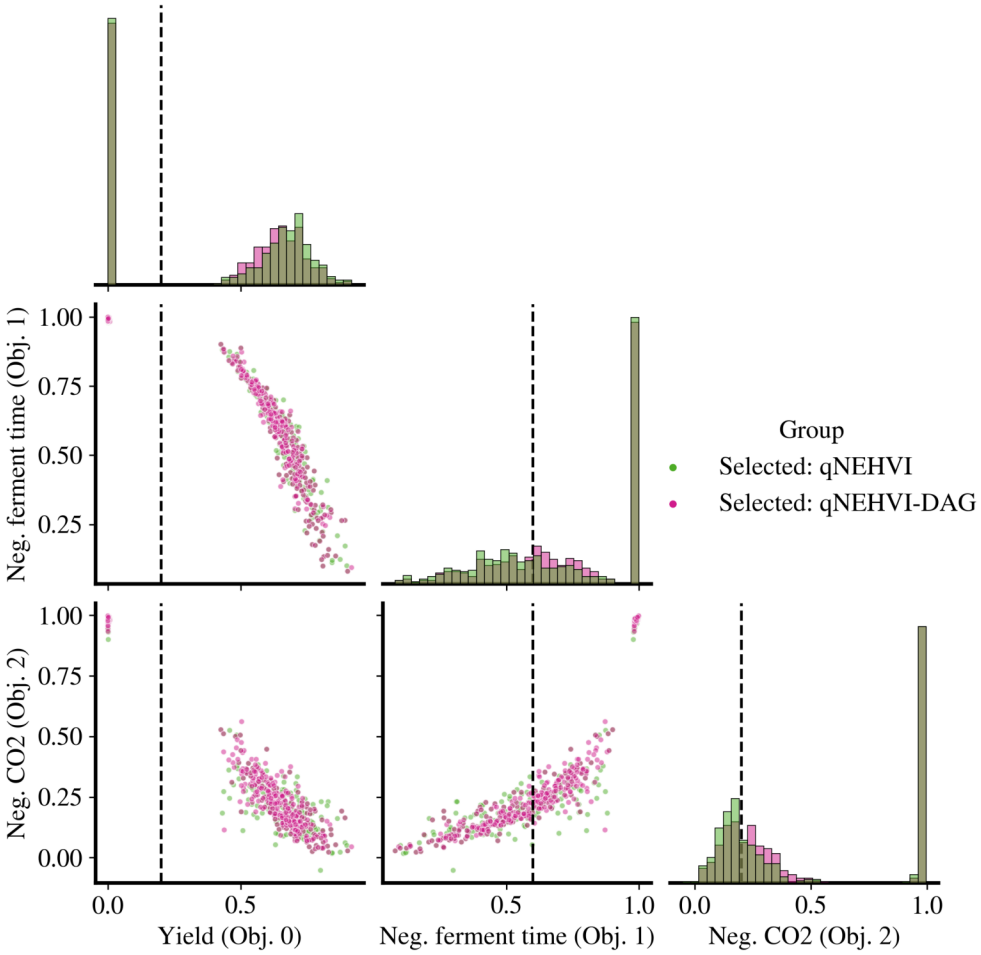


Figure 7: Pairwise visualization of ground-truth objectives for the penicillin production task. We pool the selections over five trials. Black dashed lines are the thresholds we impose.