ACQUISITION FUNCTION-GUIDED GENERATION FOR BAYESIAN OPTIMIZATION OVER DISCRETE SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian optimization (BO) over discrete spaces is challenging due to missing gradients and the sparsity of valid solutions. Latent-space BO sidesteps both issues by fitting a surrogate in the latent space of a generative model, which however substantially complicates the direct incorporation of domain knowledge that is naturally expressed in the original discrete space. In this work, we instead combine a pretrained generative model with a surrogate defined directly in data space, where inductive biases such as domain-specific kernels can be readily incorporated. Rather than maximizing the acquisition function over the discrete design space, our method samples candidates from the generative model under a distribution tilted by the acquisition function. To recover useful gradients for sampling, we embed this tilted distribution into a diffusion bridge that yields a smooth, posterior-averaged acquisition function. This leads to *acquisition function guided generation* (AFGG), which we instantiate with a categorical diffusion model and an autoregressive normalizing flow. Across a broad suite of molecular benchmarks, the diffusion model variant performs on par with leading BO methods, while the flow model variant outperforms state-of-the-art methods in most cases and remains highly competitive otherwise. Our implementation is available at `https://anonymous.4open.science/r/afgg`.

1 INTRODUCTION

Bayesian optimization (BO) offers a sample-efficient approach to problems such as molecular and protein design, where objective evaluations are costly and the search space is discrete and structured (Garnett, 2023; Yang et al., 2025). Unfortunately, direct optimization over such spaces is hard because valid designs are sparsely populated in a combinatorial space and standard continuous optimizers have no useful gradients to follow (Moss et al., 2020). Latent-space Bayesian optimization (LSBO) addresses both challenges by training a generative model over the discrete space—thereby capturing the sparse structure of valid designs—and by performing BO in its continuous latent space, where gradients for acquisition optimization are readily available (Gómez-Bombarelli et al., 2018).

However, domain knowledge is harder to incorporate in an abstract latent space, while in data space, such knowledge can often be encoded directly through inductive biases such as domain-specific kernels (Rogers & Hahn, 2010). Furthermore, the latent space must also "do two jobs at once:" support faithful generation and provide a geometry in which the objective is easy to model by a surrogate. These goals need not align, which motivates many LSBO methods to adapt the latent representation during BO (Chu et al., 2024; Deshwal & Doppa, 2021; Lee et al., 2023; 2025; Maus et al., 2022; Tripp et al., 2020), despite the small-data regime in which such adaptation can overfit (Ober et al., 2021). Moreover, even a strong generative model only approximates the sparse set of valid designs, so acquisition optimization can move into poorly supported regions of latent space where decoding becomes unreliable (Bodin et al., 2025; Moss et al., 2025).

A natural strategy for avoiding these drawbacks is to decouple the two roles: use a strong pretrained generative model to capture the sparse structure of valid designs, while keeping the surrogate and acquisition function in data space, so that domain knowledge and inductive biases can be included. To the best of our knowledge, COWBOYS (Moss et al., 2025) is the only prior BO method that follows this strategy, but it is tied to a VAE as the generative model and probability of improvement as the acquisition function.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this paper, we propose *acquisition function guided generation* (AFGG), which replaces acquisition function maximization with sampling from a distribution that tilts a frozen pretrained generative model toward designs favored by the acquisition function. The construction is generic in both components: it requires only a generative model that supports guided sampling and any non-negative acquisition function that can be evaluated pointwise; the latter need neither be differentiable nor define a probability. Since the discrete, combinatorial design space makes direct sampling from the tilted distribution difficult, we embed it into a diffusion process over the generative model's continuous space. At each noise level, this yields a smooth posterior-averaged acquisition function that provides the gradients needed for guided sampling. The resulting gradients can then guide sampling in the same spirit as guidance in diffusion models (Dhariwal & Nichol, 2021; Ho & Salimans, 2022), which gives the method its name.

Our contributions are threefold:

- **Acquisition function guided generation.** We formulate the BO inner loop as sampling from an acquisition-tilted distribution that combines a frozen pretrained generative model with a surrogate defined directly in data space (Section 3.1). We show that this target can be recovered exactly in the generative model's continuous space and that the resulting guidance term reduces to replacing the model's posterior mean by its acquisition-weighted counterpart (Section 3.3).
- **A unified construction for different generative models.** We derive AFGG from a common diffusion-bridge construction and show how it applies to both categorical diffusion models and autoregressive normalizing flows (Section 3.4). The generator remains frozen throughout BO, while the surrogate and acquisition function stay in data space.
- **State-of-the-art empirical performance.** Across a broad suite of molecular design benchmarks under tight evaluation budgets, AFGG with the categorical diffusion model performs on par with leading BO methods. More strikingly, AFGG with the flow model achieves state-of-the-art performance on most benchmarks and remains highly competitive on the rest (Section 4).

2 Background

2.1 Bayesian Optimization over Discrete Spaces

We consider Bayesian optimization over discrete, structured design spaces. In particular, we focus on sequence-valued settings and denote the design space as $\mathcal{X} := \mathcal{V}^L$, the set of length-$L$ sequences over a discrete vocabulary $\mathcal{V}$ of size V. We assume that valid or useful designs occupy a constrained and sparsely populated subset of the combinatorial space. This setting covers, for example, molecules represented as SELFIES strings (Krenn et al., 2020) and proteins represented as amino-acid sequences. We seek the maximizer w^* of an expensive-to-evaluate and black-box objective $f\colon \mathcal{X} \to \mathbb{R}$, which provides no gradients and returns measurements y that may be contaminated by Gaussian noise:

$$w^* \in \arg\max_{w \in \mathcal{X}} f(w), \qquad y(w) = f(w) + \varepsilon_{\text{obs}}, \quad \varepsilon_{\text{obs}} \sim \mathcal{N}(0, \sigma^2_{\text{obs}}). \tag{1}$$

Bayesian optimization (BO) fits a probabilistic surrogate to the observations collected so far and uses its predictive uncertainty to choose the next design to evaluate (Garnett, 2023; Shahriari et al., 2016). Throughout this work, we use Gaussian processes (GPs) (Rasmussen & Williams, 2006), whose posterior mean and variance are available in closed form. However, our method is not tied to GPs and only requires a probabilistic surrogate from which an acquisition function can be computed or approximated.

Defining the surrogate directly on the discrete space allows domain knowledge to be incorporated through suitable inductive biases, such as domain-specific kernels. For example, in our molecular experiments we use the Tanimoto kernel (Ralaivola et al., 2005; Tripp et al., 2023) on binary Morgan fingerprints $\psi(w) \in \{0,1\}^{d_{\text{fp}}}$, where usually $d_{\text{fp}} = 2048$ (Rogers & Hahn, 2010),

$$k(w, w') = \frac{\langle \psi(w), \psi(w') \rangle}{\|\psi(w)\|^2 + \|\psi(w')\|^2 - \langle \psi(w), \psi(w') \rangle} \in [0, 1]. \tag{2}$$

An acquisition function $\alpha\colon \mathcal{X} \to \mathbb{R}$ ranks candidate designs by balancing predicted performance against uncertainty. We consider two standard choices, probability of improvement (PI) and ex-

pected improvement (EI), relative to the incumbent $y^\star$:

$$\alpha^{\mathrm{PI}}(w) = \Pr\left[f(w) \geq y^\star \mid \mathcal{D}\right] \in [0,1], \qquad \alpha^{\mathrm{EI}}(w) = \mathbb{E}\left[\max(f(w) - y^\star, 0) \mid \mathcal{D}\right] \in [0,\infty), \quad (3)$$

where $\mathcal{D} = \{(w_j, y_j)\}_{j=1}^n$ denotes the observations collected so far. Both are available in closed form under a GP (Ament et al., 2023). PI is a probability, whereas EI is non-negative but potentially unbounded; this distinction will matter for our method. Standard BO selects the next design by maximizing the acquisition function $w_{\mathrm{next}} \in \arg\max_{w \in \mathcal{X}} \alpha(w)$. For discrete, combinatorial $\mathcal{X}$, this inner optimization is generally hard, and gradient-based methods for continuous BO do not apply directly.

2.2 Generative Models over Sequences

Our method requires a pretrained generative model over $\mathcal{X}$ that supports guided sampling. We consider two well-suited model classes for discrete sequences: a categorical diffusion model and an autoregressive normalizing flow. Both connect the discrete design space $\mathcal{X} = \mathcal{V}^L$ to a continuous embedding space $\mathbb{R}^{L \times d}$ in the same way: an embedding table $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ (learned by the model) with rows e_v, $v \in \mathcal{V}$, maps a sequence to its embeddings, and a deterministic decode map $\tau \colon \mathbb{R}^{L \times d} \to \mathcal{X}$ clamps each position to the closest token in cosine similarity,

$$x = E(w) \triangleq \left(e_{w_1}, \ldots, e_{w_L}\right) \in \mathbb{R}^{L \times d}, \qquad \tau(x)_i = \arg\max_{v \in \mathcal{V}} \cos\left(x_i, e_v\right), \quad i = 1, \ldots, L. \quad (4)$$

Continuous diffusion for categorical data (CDCD) (Dieleman et al., 2022) runs a variance-exploding diffusion (Karras et al., 2022) on the embeddings,

$$z_t = x + t\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \quad (5)$$

so that the forward kernel is $q_t(z_t \mid x) = \mathcal{N}(z_t; x, t^2 I)$. Samples are generated by integrating the reverse-time SDE or probability-flow ODE from $t = T$ to $t = 0$ (Song et al., 2021), which requires the score $\nabla_{z_t} \log p_t(z_t)$ of the noised marginal. Since $\nabla_{z_t} \log q_t(z_t \mid x) = (x - z_t)/t^2$ is affine in x, the score is the conditional score averaged over the denoising posterior, and in the discrete case, where each position of x is one of $\mathcal{V}$'s embeddings, it has the form

$$\nabla_{z_t} \log p_t(z_t) = \frac{\hat{z} - z_t}{t^2}, \qquad \hat{z}(z_t, t)_i = \sum_{v=1}^{V} p(w_i = v \mid z_t)\, e_v. \quad (6)$$

The score therefore depends on the joint denoising posterior only through its per-position marginals (Dieleman et al., 2022), and these are what must be learned: a transformer, trained with categorical cross-entropy on noised embeddings, outputs L mass functions over $\mathcal{V}$,

$$p_\theta(w_i = v \mid z_t, t) = \mathrm{softmax}\big(\mathrm{logits}_\theta(z_t, t)_i\big)_v, \qquad i = 1, \ldots, L. \quad (7)$$

Substituting p_θ for p in Eq. (6) gives the learned score.

Autoregressive normalizing flow (SeqFlow) (Lee et al., 2025) draws a Gaussian latent variable $z \sim \mathcal{N}(0, I)$, $z \in \mathbb{R}^{L \times d}$, and maps it through a diffeomorphism $g_\theta : \mathbb{R}^{L \times d} \to \mathbb{R}^{L \times d}$ to a sequence of continuous token embeddings, $x = g_\theta(z)$, which τ decodes by Eq. (4). Because g_θ is a bijection and τ is deterministic, the resulting distribution over designs is exactly the pushforward of the latent Gaussian,

$$P_\theta(w) = \int_{A_w} \mathcal{N}(z; 0, I)\, dz, \qquad A_w := (\tau \circ g_\theta)^{-1}(w) \subset \mathbb{R}^{L \times d}, \quad (8)$$

where the cells $\{A_w\}_{w \in \mathcal{X}}$ partition the latent space. We refer to Lee et al. (2025) for details of the architecture and training procedure.

3 Acquisition Function-Guided Generation

We now present AFGG, which steers a pretrained generative model toward designs with high acquisition values. We first define the acquisition-tilted target distribution and explain why it is hard to sample it directly in data space (Section 3.1). We then introduce a diffusion bridge that smooths this target and recovers informative gradients (Section 3.2). The resulting guidance term reduces to a difference of posterior means, which we estimate by sampling (Section 3.3). Finally, we show how the same construction applies to our two generative backbones, which differ only in how they provide the base score and posterior samples (Section 3.4).

3.1 THE SAMPLING PROBLEM

As described in Section 2, the generative model defines a density $p_\theta(x)$ on the continuous embedding space $\mathbb{R}^{L\times d}$ together with a deterministic decoder $\tau\colon \mathbb{R}^{L\times d} \to \mathcal{X}$. Each discrete design sequence $w \in \mathcal{X}$ corresponds to the region of embedding space that decodes to it,

$$\tau^{-1}(w) \triangleq \{x \in \mathbb{R}^{L\times d}\colon \tau(x) = w\}. \tag{9}$$

These regions partition the embedding space, and the probability of a sequence is the mass that p_θ assigns to its corresponding region. Hence, the decoder pushes p_θ forward to a probability mass function P_θ over the discrete sequence space,

$$P_\theta(w) = \int_{\tau^{-1}(w)} p_\theta(x)\,\mathrm{d}x. \tag{10}$$

Definition 3.1 (Tilted target)**.** Given the induced probability mass function P_θ over $\mathcal{X}$ and a non-negative acquisition function $\ell\colon \mathcal{X} \to \mathbb{R}_+$, we define the *tilted target distribution*

$$P^\star(w) \triangleq \frac{1}{Z} P_\theta(w)\ell(w), \qquad Z \triangleq \sum_{w\in\mathcal{X}} P_\theta(w)\ell(w). \tag{11}$$

The tilted distribution combines the two components of our approach: P_θ places mass on sequences supported by the pretrained generative model, while ℓ shifts this mass toward sequences favored by the surrogate. The requirement that ℓ be non-negative is mild, as a real-valued acquisition function $\tilde{\ell}$ can be turned into a positive tilt, for example through $\ell(w) = \exp(\tilde{\ell}(w))$.

This product form also has a natural interpretation through entropy-regularized control (Uehara et al., 2024). Indeed, $P^\star$ is the solution of

$$P^\star = \arg\max_P \left\{\mathbb{E}_P\left[\log \ell(w)\right] - D_{\mathrm{KL}}\big(P \,\|\, P_\theta\big)\right\}. \tag{12}$$

The first term rewards designs with high acquisition values, while the KL term keeps the resulting distribution close to the pretrained generative model. Equivalently, entropy-regularized control yields a target of the form $P_\theta(w)\exp(r(w))$, and our tilted distribution follows by taking $r(w) = \log \ell(w)$.

While sampling from the generative model P_θ is easy, the acquisition tilt changes this distribution such that directly sampling from $P^\star$ would require handling the combinatorial space $\mathcal{X}$, facing much the same difficulty as maximizing the acquisition function over $\mathcal{X}$. To sample from $P^\star$, we instead return to the continuous space of the generative model. Using Eq. (10), the corresponding tilted density is

$$p^\star(x) \triangleq \frac{1}{Z} p_\theta(x)\ell(\tau(x)). \tag{13}$$

Sampling $x \sim p^\star$ and decoding $w = \tau(x)$ yields exactly $w \sim P^\star$; we prove this in Appendix A.1.

The main technical challenge is that $\ell(\tau(x))$ is constant within each cell $\tau^{-1}(w)$, so $\nabla_x\, \ell(\tau(x)) = 0$ almost everywhere. LSBO avoids this difficulty by moving the surrogate to latent space, but in doing so inherits the limitations discussed above. The rest of this section shows how to recover informative gradients while keeping the surrogate in data space.

3.2 LIFTING THE TARGET TO A BRIDGE

We recover informative gradients by introducing a family of smoothed versions of $p^\star$, indexed by $t \in [0, T]$, with $t = 0$ corresponding to the tilted target and $t = T$ to the noise distribution. Note that, although this smoothing construction is borrowed from diffusion models (Ho et al., 2020; Song et al., 2021), the generative model itself need not be a diffusion model; it only needs to support the guided sampling procedure developed below.

Definition 3.2 (Bridge marginals)**.** Starting from the tilted density $p^\star$ and the original generative density p_θ, we define their *bridge marginals* at noise level t as

$$p_t^\star(z_t) = \int p^\star(x)\, q_t(z_t \mid x)\,\mathrm{d}x, \qquad p_t(z_t) = \int p_\theta(x)\, q_t(z_t \mid x)\,\mathrm{d}x, \tag{14}$$

where

$$q_t(z_t \mid x) = \mathcal{N}\big(z_t; \alpha_t x, \sigma_t^2 I\big) \tag{15}$$

is a Gaussian noising kernel applied to the continuous variable x.

A key observation is that the acquisition tilt is preserved under this smoothing in a convenient form. Let

$$p(x \mid z_t) = \frac{p_\theta(x) q_t(z_t \mid x)}{p_t(z_t)} \tag{16}$$

denote the posterior over the clean variable x under the original generative model.

Proposition 1 (Tilted marginal)**.** *The tilted marginal can be written as*

$$p_t^\star(z_t) \propto p_t(z_t)\, \ell_t(z_t), \qquad \ell_t(z_t) \triangleq \mathbb{E}_{p(x|z_t)}\big[\ell(\tau(x))\big], \tag{17}$$

where ℓ_t is the acquisition function averaged over the posterior uncertainty about the clean design sequence. Moreover, the corresponding tilted posterior satisfies

$$p^\star(x \mid z_t) \propto p(x \mid z_t)\, \ell(\tau(x)). \tag{18}$$

Proof. The proof is in Appendix A.2. □

This factorization makes guidance convenient. To sample from $p^\star$ by reversing the bridge, we need the score of the tilted marginal,

$$\nabla_{z_t} \log p_t^\star(z_t) = \nabla_{z_t} \log p_t(z_t) + \nabla_{z_t} \log \ell_t(z_t). \tag{19}$$

The first term is the score of the pretrained generative model at noise level t, while the second steers it toward regions favored by the acquisition function. This has the same form as guidance in diffusion models (Dhariwal & Nichol, 2021).

Importantly, ℓ_t is no longer the piecewise-constant function $\ell \circ \tau$. Instead, it averages $\ell(\tau(x))$ over the posterior $p(x \mid z_t)$, and can therefore vary smoothly with the continuous variable z_t. Hence, $\nabla_{z_t} \log \ell_t(z_t)$ can provide an informative guidance signal even though $\nabla_x \ell(\tau(x)) = 0$ almost everywhere.

Evaluating Eq. (19) therefore requires two ingredients: (i) the untilted score $\nabla_{z_t} \log p_t(z_t)$ and (ii) samples from the posterior $p(x \mid z_t)$. The next section shows how these ingredients yield the guidance term, while Section 3.4 explains how they are obtained for two generative backbones.

3.3 Guidance Is a Tilted Mean Difference

The central object in Eq. (19) is the guidance term $\nabla_{z_t} \log \ell_t(z_t)$. For the Gaussian kernel Eq. (15), this term has a particularly simple form: it is proportional to the difference between the ordinary posterior mean and its acquisition-weighted counterpart.

Definition 3.3 (Tilted estimate)**.** With $p^\star(x \mid z_t)$ as in Eq. (18), define

$$\hat{z}^{\mathrm{tilt}} \triangleq \mathbb{E}_{p^\star(x|z_t)}\big[x\big] = \frac{\mathbb{E}_{p(x|z_t)}\big[\ell(\tau(x))\, x\big]}{\mathbb{E}_{p(x|z_t)}\big[\ell(\tau(x))\big]}, \qquad \hat{z} \triangleq \mathbb{E}_{p(x|z_t)}\big[x\big]. \tag{20}$$

Thus, $\hat{z}$ is the posterior mean under the pretrained generative model, whereas $\hat{z}^{\mathrm{tilt}}$ gives more weight to posterior samples whose decoded designs have high acquisition value.

Proposition 2 (Guidance score as tilted mean difference)**.** *Let the kernel of Definition 3.2 be the Gaussian kernel (15), with $\alpha_t, \sigma_t > 0$. Then*

$$\nabla_{z_t} \log \ell_t(z_t) = \frac{\alpha_t\big(\hat{z}^{\mathrm{tilt}} - \hat{z}\big)}{\sigma_t^2}, \qquad \nabla_{z_t} \log p_t(z_t) = \frac{\alpha_t\, \hat{z} - z_t}{\sigma_t^2}. \tag{21}$$

Substituting both expressions into Eq. (19) gives

$$\nabla_{z_t} \log p_t^\star(z_t) = \frac{1}{\sigma_t^2}\Big[\big(\alpha_t \hat{z} - z_t\big) + \alpha_t\big(\hat{z}^{\mathrm{tilt}} - \hat{z}\big)\Big] = \frac{\alpha_t \hat{z}^{\mathrm{tilt}} - z_t}{\sigma_t^2}. \tag{22}$$

Proof. The proof is in Appendix A.4. □

The guided score therefore has exactly the same form as the original score, with the posterior mean $\hat{z}$ being replaced by its acquisition-weighted counterpart $\hat{z}^{\mathrm{tilt}}$. This *tilted mean difference* is a key simplification, as guidance does not require differentiating the acquisition function, but only estimating $\hat{z}^{\mathrm{tilt}}$.

In general, $\hat{z}^{\mathrm{tilt}}$ is not available in closed form because $\ell(\tau(x))$ need not factorize across sequence positions, so we estimate it from samples $x^{(n)} \sim p(\cdot \mid z_t)$ by self-normalized importance sampling. For N samples, let

$$\ell_n = \ell\big(\tau(x^{(n)})\big), \qquad h_n = \frac{\ell_n}{\sum_{m=1}^{N} \ell_m}, \qquad \hat{z}^{\mathrm{tilt}} \approx \sum_{n=1}^{N} h_n x^{(n)}. \tag{23}$$

This estimator is consistent (Appendix A.5); guidance strength can further be controlled by a scale λ (see Eq. (45)) or a temperature γ on the importance weights (see Eq. (46)).

3.4 Two Backbones, One Bridge

CDCD. CDCD's forward process in Eq. (5) is the kernel in Eq. (15) with the variance-exploding schedule $\alpha_t \equiv 1$, $\sigma_t = t$ (Karras et al., 2022), so $p_t^\star = p^\star * \mathcal{N}(0, t^2 I)$ and z_t lives in the embedding space of Eq. (4). The network provides both required quantities: the learned score Eq. (6) and the per-position posterior marginals Eq. (7), from which posterior draws are taken under a mean-field approximation (Appendix B). At sufficiently large T, the added noise dominates both p_θ and the tilt,

$$p_T^\star \approx p_T \approx \mathcal{N}(0, T^2 I), \tag{24}$$

so reversing the bridge with the PF-ODE or SDE driven by Eq. (19) yields $x \sim p^\star$, which decodes to $w = \tau(x) \sim P^\star$.

SeqFlow. SeqFlow has no built-in noising process. Although $p_\theta(x)$ is tractable, applying Eq. (15) in embedding space makes both the noised marginal p_t and the posterior $p(x \mid z_t)$ intractable. We therefore move the bridge to the flow latent $z = g_\theta^{-1}(x)$, whose untilted law is exactly $\mathcal{N}(0, I)$ by Eq. (8). Since g_θ is a diffeomorphism, transporting the tilted density Eq. (13) to the latent carries the same Jacobian factor J as transporting p_θ, so the factor cancels and only the tilt survives (Appendix A.3),

$$\pi^\star(z) \triangleq p^\star\big(g_\theta(z)\big)\,\big|\det J_{g_\theta}(z)\big| = \frac{1}{Z}\,\mathcal{N}(z; 0, I)\,\ell\big(\tau(g_\theta(z))\big), \tag{25}$$

with $\tau(g_\theta(z)) \sim P^\star$ for $z \sim \pi^\star$. As in Section 3.1, the tilt remains piecewise constant on the cells A_w of Eq. (8), so we smooth it using

$$q_t(z_t \mid z) = \mathcal{N}\big(z_t; \alpha_t z, \sigma_t^2 I\big), \tag{26}$$

with the variance-preserving schedule $\sigma_t = \sqrt{1 - \alpha_t^2}$, $\alpha_t \in (0, 1]$. Gaussian conjugacy then gives both required quantities in closed form,

$$\text{(i) } \nabla_{z_t} \log p_t(z_t) = -z_t, \qquad \text{(ii) } p(z \mid z_t) = \mathcal{N}\Big(z; \alpha_t z_t, (1 - \alpha_t^2) I\Big). \tag{27}$$

The posterior is exact and network-free; the flow enters only through the tilt, evaluated once per draw and never differentiated. Thus, Section 3.3 applies with x replaced by z, while CDCD operates in embedding space. Table 2 summarizes the correspondence between the two backbones.

3.5 Trust Region

Restricting search to a neighborhood of the incumbent, the best design observed so far, is a common strategy for improving the sample efficiency of high-dimensional BO (Eriksson et al., 2019). In AFGG, we implement this locality through the initialization of the guided sampling trajectories. Both backbones start sampling from a Gaussian, $z_T \sim \mathcal{N}(0, \sigma^2 I)$, with $\sigma = T$ for CDCD (see Eq. (24)) and $\sigma = 1$ for SeqFlow. Instead, we wish to start closer to an anchor $z^\star$ that represents

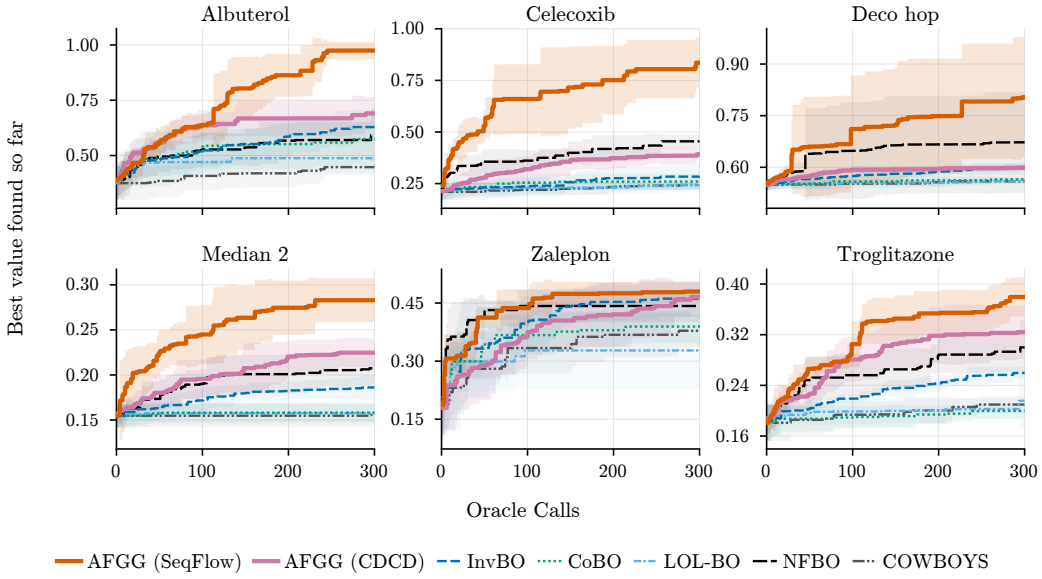

Figure 1: **Sequential setting.** Optimization trajectories on six representative objectives, with 10 initial observations followed by 300 oracle evaluations, one per BO iteration. Curves show the median over 5 seeds.

the incumbent in the same space. However, sample quality deteriorates if the model must process an input that is unlikely under the D-dimensional Gaussian it expects, which concentrates its mass on a shell of radius $\sigma\sqrt{D}$ around the origin (Blum et al., 2020, Thm. 2.9). We therefore sample

$$z_T \sim \mathcal{N}\big(z_T;\, \rho z^\star,\, (1-\rho^2)\,\sigma^2 I\big), \tag{28}$$

where $\rho \in [0,1]$ controls how tightly the trust region concentrates around $z^\star$: $\rho = 0$ recovers the unconditional initialization and $\rho = 1$ starts every trajectory at $z^\star$. Since $\mathbb{E}[\|z_T\|^2 \mid z^\star] = \rho^2\|z^\star\|^2 + (1-\rho^2)\,\sigma^2 D$, an anchor on the shell, $\|z^\star\|^2 = \sigma^2 D$, gives $\mathbb{E}[\|z_T\|^2 \mid z^\star] = \sigma^2 D$ for every ρ, so the initializations stay near the shell.

The complete AFGG optimization loop is described in Appendix D.

4 Experiments

We evaluate AFGG on the 25 molecular optimization objectives of Gao et al. (2022) and González-Duque et al. (2024), following the small-data protocol of Moss et al. (2025). We consider both a *sequential setting*, with 10 initial observations followed by 300 oracle evaluations, one per BO iteration, and a *batch setting*, with 100 initial observations followed by 500 oracle evaluations in batches of five. We instantiate AFGG with two different generative backbones, SeqFlow (Lee et al., 2025) and CDCD (Dieleman et al., 2022), as described in Section 3.4, allowing us to test the same acquisition-guided construction with two substantially different generative models. For SeqFlow, we use the released GuacaMol-trained checkpoint; since no pretrained CDCD checkpoint was available, we train one on the same dataset (Appendix B.1).

We compare against five strong recent methods that are particularly relevant to our setting: CoBO (Lee et al., 2023), LOL-BO (Maus et al., 2022), COWBOYS (Moss et al., 2025), InvBO (Chu et al., 2024), and NFBO (Lee et al., 2025). These include both the closest prior approach, COWBOYS, and NFBO, which uses the same SeqFlow generative backbone as one of our AFGG variants. For the sequential setting, Appendix E.1 (Table 6) broadens the comparison to 11 baselines, additionally including CMA-ES, differential evolution (DE), genetic algorithms (GA), hill climbing (HC), RLBO (Kirschner et al., 2019), and TuRBO (Eriksson et al., 2019). Unless stated otherwise, AFGG uses EI with the temperature tilt of Eq. (46), which performs best in our ablations (Appendix E.3).

	baselines					AFGG	
Objective	CoBO	InvBO	LOL-BO	NFBO	COWBOYS	CDCD	SeqFlow
albu_sim	0.593 ± 0.111	0.629 ± 0.079	0.509 ± 0.065	0.593 ± 0.119	0.448 ± 0.037	**0.691 ± 0.073**	**0.975 ± 0.038**
amlo_mpo	0.523 ± 0.012	0.538 ± 0.050	0.516 ± 0.022	0.563 ± 0.095	0.472 ± 0.038	0.546 ± 0.039	**0.565 ± 0.045**
cele_red	0.262 ± 0.041	0.286 ± 0.068	0.243 ± 0.035	0.455 ± 0.035	0.243 ± 0.020	0.392 ± 0.024	**0.836 ± 0.125**
deco_hop	0.565 ± 0.008	0.597 ± 0.019	0.560 ± 0.009	0.673 ± 0.147	0.559 ± 0.008	0.599 ± 0.010	**0.803 ± 0.175**
drd2	0.510 ± 0.370	0.963 ± 0.037	0.626 ± 0.184	0.971 ± 0.066	0.796 ± 0.243	0.730 ± 0.226	**1.000 ± 0.000**
fexo_mpo	0.710 ± 0.034	0.734 ± 0.027	0.716 ± 0.021	0.723 ± 0.032	0.683 ± 0.031	**0.763 ± 0.017**	**0.741 ± 0.016**
gsk3b	0.362 ± 0.069	0.500 ± 0.042	0.336 ± 0.103	0.656 ± 0.174	0.418 ± 0.093	0.518 ± 0.088	**0.950 ± 0.037**
iso_c7	0.875 ± 0.145	**1.000 ± 0.000**	0.760 ± 0.036	0.860 ± 0.092	0.919 ± 0.046	0.957 ± 0.049	0.981 ± 0.043
iso_c9	0.577 ± 0.050	0.699 ± 0.147	0.630 ± 0.082	0.758 ± 0.024	**0.834 ± 0.020**	0.722 ± 0.079	0.809 ± 0.081
jnk3	0.148 ± 0.031	0.206 ± 0.063	0.144 ± 0.035	0.412 ± 0.095	0.176 ± 0.032	0.216 ± 0.033	**0.778 ± 0.038**
med_1	0.178 ± 0.025	0.253 ± 0.037	0.175 ± 0.023	0.203 ± 0.024	0.192 ± 0.024	**0.346 ± 0.031**	**0.352 ± 0.045**
med_2	0.158 ± 0.011	0.186 ± 0.014	0.158 ± 0.011	0.207 ± 0.018	0.156 ± 0.012	**0.225 ± 0.016**	**0.283 ± 0.024**
mest_sim	0.421 ± 0.064	0.531 ± 0.054	0.384 ± 0.059	0.555 ± 0.125	0.404 ± 0.023	**0.593 ± 0.051**	**0.658 ± 0.114**
osim_mpo	0.761 ± 0.013	0.782 ± 0.024	0.767 ± 0.015	**0.803 ± 0.019**	0.749 ± 0.023	0.778 ± 0.023	0.789 ± 0.034
peri_mpo	0.487 ± 0.049	0.447 ± 0.054	0.416 ± 0.079	0.434 ± 0.015	0.344 ± 0.034	0.432 ± 0.031	**0.502 ± 0.017**
rano_mpo	0.688 ± 0.027	0.742 ± 0.033	0.684 ± 0.042	0.696 ± 0.053	0.708 ± 0.040	**0.757 ± 0.034**	**0.785 ± 0.007**
logp	**65.866 ± 10.720**	**49.799 ± 19.126**	34.741 ± 5.318	25.055 ± 8.705	15.443 ± 3.623	22.684 ± 2.667	35.734 ± 10.360
qed	0.928 ± 0.004	0.942 ± 0.006	0.934 ± 0.003	0.943 ± 0.008	0.898 ± 0.034	0.940 ± 0.006	**0.945 ± 0.002**
sa	**9.001 ± 0.347**	8.393 ± 0.543	8.214 ± 0.711	7.538 ± 0.452	8.020 ± 0.363	8.349 ± 0.606	8.560 ± 0.220
scaf_hop	0.441 ± 0.029	0.501 ± 0.030	0.430 ± 0.020	0.510 ± 0.062	0.442 ± 0.027	0.500 ± 0.028	**0.583 ± 0.070**
sita_mpo	0.255 ± 0.096	**0.330 ± 0.032**	0.228 ± 0.087	0.290 ± 0.030	**0.352 ± 0.090**	0.108 ± 0.092	0.295 ± 0.056
thio_red	0.270 ± 0.025	0.308 ± 0.017	0.255 ± 0.011	0.376 ± 0.053	0.261 ± 0.010	**0.395 ± 0.051**	**0.475 ± 0.060**
trog_red	0.205 ± 0.004	0.260 ± 0.010	0.218 ± 0.044	0.300 ± 0.044	0.210 ± 0.014	**0.324 ± 0.043**	**0.380 ± 0.031**
vals_sma	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	**0.015 ± 0.034**	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
zale_mpo	0.404 ± 0.055	0.461 ± 0.047	0.348 ± 0.065	0.442 ± 0.042	0.378 ± 0.044	**0.466 ± 0.037**	**0.480 ± 0.026**

Table 1: **Sequential benchmark (partial results).** Final best score on all 25 PMO objectives after 300 oracle calls (10 initial points, 1 point per BO iteration), mean ± sample standard deviation over 5 seeds. For readability the table shows five of the eleven baselines; Table 6 gives the full comparison. AFGG is shown in two instantiations: **CDCD** and **SeqFlow**. Every cell is judged against the opposing group. **Bold**: the cell's mean exceeds every mean in the opposing group. Green, shaded: lead exceeds the cell's own sample standard deviation; bold without shading denotes a smaller lead. Objective abbreviations are defined in Table 5.

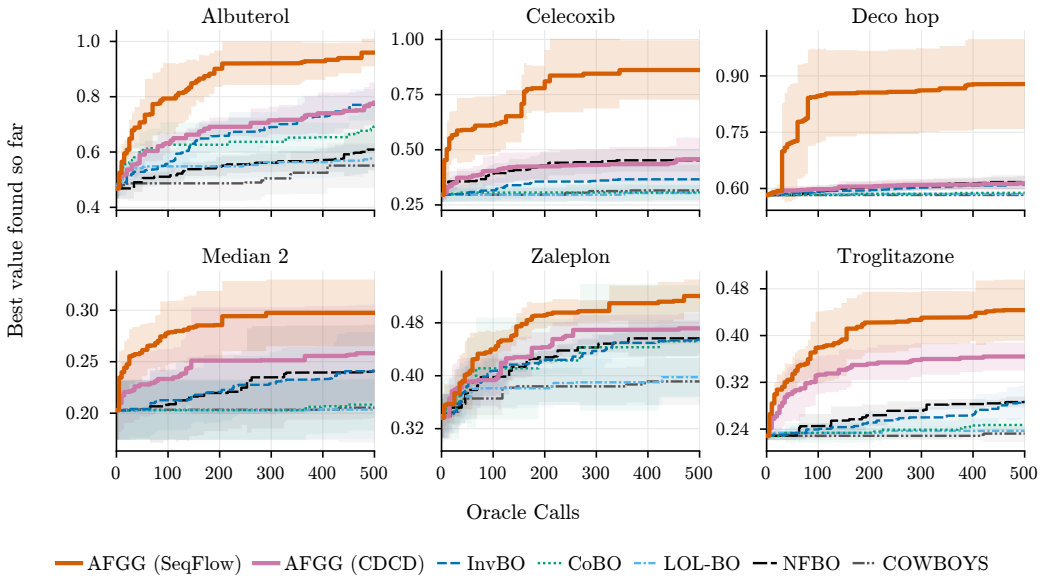

Figure 2: **Batch setting, uncoupled generation.** Optimization trajectories on the same six representative objectives, with 100 initial observations followed by 500 oracle evaluations in batches of five. Curves show the median over 5 seeds.

Fig. 1 shows the optimization trajectories on six representative objectives in the sequential setting. AFGG with SeqFlow makes particularly strong gains on several of these tasks, while the CDCD variant remains broadly competitive with the leading BO baselines. To verify that this pattern is not

specific to the selected objectives, Table 1 reports the final performance on all 25 objectives against the five closest BO baselines.

Fig. 2 repeats the comparison in the batch setting on the same six representative objectives. The advantage of the SeqFlow variant persists when $q = 5$ candidates are selected per BO round. Appendix E.1 reports results on all 25 objectives for both uncoupled and coupled batch constructions.

Across both settings, the two AFGG backbones show a clear and consistent pattern. AFGG with CDCD performs on par with the strongest BO baselines, whereas AFGG with SeqFlow achieves the best result on most objectives and remains competitive on the others. The same ordering across the sequential and batch settings suggests that the stronger generative backbone translates into substantially stronger optimization performance. **Taken together, these results demonstrate state-of-the-art performance for AFGG with SeqFlow across this benchmark suite: it achieves the best result on most objectives and remains close to the strongest baseline on the others.**

Additional analyses in Appendix E.2 isolate the contribution of the generative backbone and relate optimization performance to the quality of its unconditional samples.

5 Related Work

AFGG is related to *inference-time alignment* of pretrained generative models, which likewise samples from a distribution obtained by tilting a generative model toward a reward (Krishnamoorthy et al., 2023; Li et al., 2025; Uehara et al., 2024; 2025). The key difference is that inference-time alignment typically uses a fixed reward, whereas in Bayesian optimization the acquisition function changes after every new observation. AFGG therefore applies the same general idea of guided generation in a setting where the tilt is continually updated by the surrogate.

Another line of related work is *active generation*, where the generative model is adapted repeatedly as new data arrive, for example by conditioning or variationally fitting it to an acquisition-weighted target (Brookes et al., 2019; Oliveira et al., 2026; Steinberg et al., 2025; Yun et al., 2025); the same approach can also be applied to infilling problems, where the generated sequence has fixed length (Gruver et al., 2023; Stanton et al., 2022). AFGG takes a different route: the pretrained generator remains frozen, while only the surrogate and acquisition function are updated during BO. This allows strong pretrained generative models to be reused throughout optimization without repeated retraining or fine-tuning.

Most generative BO methods place the surrogate in the learned latent space (Chu et al., 2024; Lee et al., 2023; 2025; Maus et al., 2022; Tripp et al., 2020), which makes inductive biases defined naturally on the original designs harder to use directly. To the best of our knowledge, COWBOYS (Moss et al., 2025) is the only prior BO method that similarly combines a frozen generative model with a surrogate defined in data space. It is therefore the closest precursor to AFGG, but its construction is tied to a VAE and to probability of improvement. In contrast, AFGG only requires a generative model that supports guided sampling and a non-negative acquisition function that can be evaluated pointwise. The acquisition function need neither be differentiable nor define a probability, allowing, for example, EI to be used directly.

6 Conclusion

We introduced AFGG, which replaces acquisition-function maximization with sampling from an acquisition-tilted generative distribution. A diffusion bridge makes this target amenable to guided sampling while leaving the surrogate and acquisition function in data space. The generator remains frozen throughout BO, and the acquisition function only needs to be non-negative and evaluable pointwise. On the molecular benchmarks studied here, AFGG with CDCD is competitive with strong recent BO methods, while AFGG with SeqFlow achieves the best result on most objectives and remains close to the strongest baselines on the rest. The gap between the two variants further shows that the quality of the generative backbone matters directly for optimization performance. This makes stronger pretrained generators a natural next target for AFGG. In particular, protein foundation models (Abramson et al., 2024; Passaro et al., 2025; Watson et al., 2023) could potentially serve as frozen backbones. Applying AFGG to such models will mainly require efficient approximations to the posterior $p(x \mid z_t)$ used for guidance.

7 AI USE STATEMENT

Large language models were used as an assistive tool during the preparation of this work, under author supervision, in three roles:

- **Writing.** Copy-editing of author-written text for clarity and concision, and suggestions for reorganizing existing material. No section was drafted from scratch by a model; all technical claims, derivations and experimental conclusions are the authors'.
- **Related work.** Surfacing candidate references and drafting initial summaries of them. Every cited work was subsequently read by the authors, and all attributions and comparisons were verified against the primary sources.
- **Code.** Inspecting the released implementations of the baselines. Standardizing the implementation and evaluation protocol across methods required modifications to the original code, and an AI assistant was used to help locate and implement them. Every deviation from an original implementation is documented, with the corresponding line numbers.

The authors take full responsibility for the entire content of this paper, including any text, code or references produced with AI assistance.

8 REPRODUCIBILITY STATEMENT

An anonymized implementation is available at `https://anonymous.4open.science/r/afgg`. It covers the guided samplers of Algorithm 1 and Algorithm 2, the BO loop, and our modified baseline code, together with the configuration files and random seeds needed to reproduce every table and figure reported here. The guidance construction is stated in full in Section 3 and proved in Appendix A. Hyperparameters and training details for the CDCD generator are given in Table 4 and Appendix B.1; the SeqFlow backbone is the pretrained model of Lee et al. (2025), used without modification.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with AlphaFold 3. In *Nature*, volume 630, pp. 493–500. Nature Publishing Group, 2024.

Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of Data Science*. Cambridge University Press, 2020.

Erik Bodin, Alexandru I. Stere, Dragos D Margineantu, Carl Henrik Ek, and Henry Moss. Linear combinations of latents in generative models: subspaces and beyond. In *The Thirteenth International Conference on Learning Representations*, 2025.

David H. Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International Conference on Machine Learning*, 2019.

Jaewon Chu, Jinyoung Park, Seunghun Lee, and Hyunwoo J. Kim. Inversion-based latent Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Aryan Deshwal and Jana Doppa. Combining latent space and structured kernels for bayesian optimization over combinatorial spaces. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

David Eriksson, Michael Pearce, Jacob R. Gardner, Ryan D. Turner, and Matthias Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor W. Coley. Sample efficiency matters: A benchmark for practical molecular optimization. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2022.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Miguel González-Duque, Richard Michael, Simon Bartels, Yevgen Zainchkovskyy, Søren Hauberg, and Wouter Boomsma. A survey and benchmark of high-dimensional Bayesian optimization of discrete sequences. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Nate Gruver, Samuel Stanton, Nathan C. Frey, Tim G. J. Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein design with guided discrete diffusion. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Johannes Kirschner, Mojmír Mutný, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3429–3438. PMLR, 2019.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alán Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Siddarth Krishnamoorthy, Satvik Mashkaria, and Aditya Grover. Diffusion models for black-box optimization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Seunghun Lee, Jaewon Chu, Sihyeon Kim, Juyeon Ko, and Hyunwoo J. Kim. Advancing bayesian optimization via learning correlated latent space. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23. Curran Associates Inc., 2023.

Seunghun Lee, Jinyoung Park, Jaewon Chu, Minseo Yoon, and Hyunwoo J. Kim. Latent bayesian optimization via autoregressive normalizing flows. In *The Thirteenth International Conference on Learning Representations*, 2025.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. In *International Conference on Machine Learning*, 2025.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020.

Natalie Maus, Haydn T. Jones, Juston S. Moore, Matt J. Kusner, John Bradshaw, and Jacob R. Gardner. Local latent space Bayesian optimization over structured inputs. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Henry B. Moss, Daniel Beck, Javier González, David S. Leslie, and Paul Rayson. Boss: Bayesian optimization over string spaces. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc., 2020.

Henry B. Moss, Sebastian W. Ober, and Tom Diethe. Return of the latent space COWBOYS: Rethinking the use of VAEs for Bayesian optimisation of structured spaces. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 44956–44970. PMLR, 2025.

Sebastian W. Ober, Carl E. Rasmussen, and Mark van der Wilk. The promises and pitfalls of deep kernel learning. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1206–1216. PMLR, 2021.

Rafael Oliveira, Daniel M. Steinberg, and Edwin V. Bonilla. Generative Bayesian optimization: Generative models as acquisition functions. In *International Conference on Learning Representations*, 2026.

Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stärk, David Kwabi-Addo, Dominique Beaini, Tommi Jaakkola, and Regina Barzilay. Boltz-2: Towards accurate and efficient binding affinity prediction. In *bioRxiv*, 2025.

Liva Ralaivola, Sanjay J. Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110, 2005.

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

Francesco Sacco. Mini continuous diffusion from categorical data. `https://github.com/Francesco215/text-diffusion`, 2024. GitHub repository.

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, 2016.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

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. In *International Conference on Machine Learning*, 2022.

Daniel M. Steinberg, Rafael Oliveira, Cheng Soon Ong, and Edwin V. Bonilla. Variational search distributions. In *International Conference on Learning Representations*, 2025.

Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Austin Tripp, Sergio Bacallado, Sukriti Singh, and José Miguel Hernández-Lobato. Tanimoto random features for scalable molecular machine learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M. Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *International Conference on Learning Representations*, 2024.

Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xingyu Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Reward-guided controlled generation for inference-time alignment in diffusion models: Tutorial and review. *arXiv preprint arXiv:2501.09685*, 2025.

Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with RFdiffusion. In *Nature*, volume 620, pp. 1089–1100. Nature Publishing Group, 2023.

Jason Yang, Ravi G. Lal, James C. Bowden, Raul Astudillo, Mikhail A. Hameedi, Sukhvinder Kaur, Matthew Hill, Yisong Yue, and Frances H. Arnold. Active learning-assisted directed evolution. *Nature Communications*, 2025. doi: 10.1038/s41467-025-55987-8.

Taeyoung Yun, Kiyoung Om, Jaewoo Lee, Jinkyoo Yun, and Jinkyoo Park. Posterior inference with diffusion models for high-dimensional black-box optimization. In *International Conference on Machine Learning*, 2025.

A Proofs

A.1 Tilting Commutes with Decoding

Proposition 3 (Lifting)**.** *Let $p^\star$ be the tilted density Eq. (13) and $P^\star$ the tilted target of Definition 3.1. Then a draw $x \sim p^\star$ returns $\tau(x) \sim P^\star$.*

Proof. Write $Z = \int_{\mathbb{R}^{L\times d}} p_\theta(x)\,\ell(\tau(x))\,\mathrm{d}x$ for the normalizing constant of $p^\star$. On the cell $\tau^{-1}(w)$ the decode map is constant by definition, $\tau(x) = w$, so $\ell(\tau(x)) = \ell(w)$ factors out of the integral:

$$\begin{aligned}\Pr_{x\sim p^\star}\big[\tau(x) = w\big] &= \int_{\tau^{-1}(w)} p^\star(x)\,\mathrm{d}x = \frac{1}{Z}\int_{\tau^{-1}(w)} p_\theta(x)\,\ell(\tau(x))\,\mathrm{d}x \\ &= \frac{\ell(w)}{Z}\int_{\tau^{-1}(w)} p_\theta(x)\,\mathrm{d}x = \frac{P_\theta(w)\,\ell(w)}{Z}.\end{aligned} \tag{29}$$

The same decomposition identifies the constant. Since the cells $\{\tau^{-1}(w)\}_{w\in\mathcal{X}}$ partition $\mathbb{R}^{L\times d}$,

$$Z = \sum_{w\in\mathcal{X}} \int_{\tau^{-1}(w)} p_\theta(x)\,\ell(\tau(x))\,\mathrm{d}x = \sum_{w\in\mathcal{X}} P_\theta(w)\,\ell(w), \tag{30}$$

which is exactly the normalizing constant of $P^\star$ in Eq. (11). Hence $\tau(x) \sim P^\star$ for $x \sim p^\star$. □

Note that only the constancy of $\ell \circ \tau$ on cells was used; no property of p_θ and no regularity of τ beyond measurability is required. This is what licenses tilting in the continuous space and decoding afterwards, and for SeqFlow it is applied a second time with τ replaced by $\tau \circ g_\theta$ (Appendix A.3).

A.2 Proof of Proposition 1

Statement. With $\ell_t(z_t) \triangleq \mathbb{E}_{p(x|z_t)}\big[\ell(\tau(x))\big]$, Proposition 1 asserts that the tilted marginal and the tilted posterior factor as

$$p_t^\star(z_t) \;\propto\; p_t(z_t)\,\ell_t(z_t), \qquad p^\star(x \mid z_t) \;\propto\; p(x \mid z_t)\,\ell(\tau(x)),$$

the first being Eq. (17).

Proof. Expand $p_t^\star$ using Definition 3.1 and Definition 3.2:

$$p_t^\star(z_t) = \int q_t(z_t \mid x)\, p^\star(x)\,\mathrm{d}x \propto \int q_t(z_t \mid x)\, p_\theta(x)\,\ell(\tau(x))\,\mathrm{d}x. \tag{31}$$

By Bayes' rule, $q_t(z_t \mid x)\, p_\theta(x) = p(x \mid z_t)\, p_t(z_t)$, so

$$p_t^\star(z_t) \propto p_t(z_t) \int p(x \mid z_t)\,\ell(\tau(x))\,\mathrm{d}x = p_t(z_t)\,\ell_t(z_t), \tag{32}$$

with $\ell_t(z_t) = \mathbb{E}_{p(x|z_t)}[\ell(\tau(x))]$, which is Eq. (17).

For the second claim, apply Bayes' rule to the tilted pair $\big(p^\star, q_t\big)$ and substitute both expansions above:

$$p^\star(x \mid z_t) = \frac{p^\star(x)\, q_t(z_t \mid x)}{p_t^\star(z_t)} \propto \frac{p_\theta(x)\,\ell(\tau(x))\, q_t(z_t \mid x)}{p_t(z_t)\,\ell_t(z_t)} = \underbrace{\frac{p_\theta(x)\, q_t(z_t \mid x)}{p_t(z_t)}}_{= p(x|z_t)} \cdot \frac{\ell(\tau(x))}{\ell_t(z_t)}. \tag{33}$$

Since $\ell_t(z_t)$ does not depend on x, it is absorbed into the normalizing constant, giving $p^\star(x \mid z_t) \propto p(x \mid z_t)\,\ell(\tau(x))$, which is Eq. (18). □

A.3 Change of Variables to the Flow Latent

Section 3.4 runs SeqFlow in its latent $z = g_\theta^{-1}(x)$ rather than in x. This requires two facts: the tilted target Eq. (13), pushed through g_θ^{-1}, is again a Gaussian tilted by the acquisition, now composed with the decoder, Eq. (25); and decoding a draw of it recovers $P^\star$. Both follow from the change-of-variables formula, because the Jacobian of the flow enters the tilted and the untilted density in the same way.

Proposition 4 (Flow-latent tilt)**.** *Let g_θ be a diffeomorphism of $\mathbb{R}^{L\times d}$ such that $z \sim \mathcal{N}(0, I)$ implies $g_\theta(z) \sim p_\theta$, as in Eq. (8), and let $\pi^\star(z) \triangleq p^\star\big(g_\theta(z)\big)\,|J(z)|$, with $J(z)$ the Jacobian determinant of g_θ at z. Then*

$$\pi^\star(z) = \frac{1}{Z}\,\mathcal{N}(z; 0, I)\;\ell\big(\tau(g_\theta(z))\big),$$

which is Eq. (25), and $z \sim \pi^\star$ implies $\tau(g_\theta(z)) \sim P^\star$.

Proof. Since g_θ is a diffeomorphism, the change-of-variables formula applies to $p^\star$ and to p_θ with the *same* Jacobian factor. Writing $J(z)$ for the Jacobian determinant of g_θ at z,

$$\pi^\star(z) = p^\star\big(g_\theta(z)\big)\,|J(z)|, \qquad \mathcal{N}(z; 0, I) = p_\theta\big(g_\theta(z)\big)\,|J(z)|, \tag{34}$$

the second because g_θ transports $\mathcal{N}(0, I)$ to p_θ, as in Eq. (8). Substituting $p^\star = \frac{1}{Z}\, p_\theta \cdot (\ell \circ \tau)$ from Eq. (13) into the first and recognizing the second,

$$\pi^\star(z) = \frac{1}{Z}\,\underbrace{p_\theta\big(g_\theta(z)\big)\,|J(z)|}_{= \mathcal{N}(z;0,I)}\;\ell\big(\tau(g_\theta(z))\big), \tag{35}$$

which is Eq. (25): the Jacobian is common to both densities and cancels in the ratio $\pi^\star/\mathcal{N}(0, I) = (p^\star/p_\theta) \circ g_\theta$. Finally, $z \sim \pi^\star$ gives $g_\theta(z) \sim p^\star$ by the first identity of Eq. (34), so $\tau(g_\theta(z)) \sim P^\star$ by Appendix A.1. □

A.4 Proof of Proposition 2

Statement. For the Gaussian kernel Eq. (15) with $\alpha_t, \sigma_t > 0$, and with $\hat{z}, \hat{z}^{\mathrm{tilt}}$ as in Definition 3.3, Proposition 2 asserts

$$\nabla_{z_t} \log \ell_t(z_t) = \frac{\alpha_t\big(\hat{z}^{\mathrm{tilt}} - \hat{z}\big)}{\sigma_t^2}, \qquad \nabla_{z_t} \log p_t(z_t) = \frac{\alpha_t\, \hat{z} - z_t}{\sigma_t^2},$$

so that the guided score Eq. (19) is $\big(\alpha_t \hat{z}^{\mathrm{tilt}} - z_t\big)/\sigma_t^2$, i.e. Eq. (21) and Eq. (22).

Proof. We differentiate $\log \ell_t$ directly and exhibit the two means whose z_t cancels. Write

$$I(z_t) \triangleq \int \ell(\tau(x))\, q_t(z_t \mid x)\, p_\theta(x)\, \mathrm{d}x. \tag{36}$$

By Bayes' rule, $p(x \mid z_t) = q_t(z_t \mid x)\, p_\theta(x)/p_t(z_t)$, so the definition of ℓ_t in Proposition 1 reads

$$\ell_t(z_t) = \int \ell(\tau(x))\, p(x \mid z_t)\, \mathrm{d}x = \frac{1}{p_t(z_t)} \int \ell(\tau(x))\, q_t(z_t \mid x)\, p_\theta(x)\, \mathrm{d}x = \frac{I(z_t)}{p_t(z_t)}, \tag{37}$$

and therefore

$$\nabla_{z_t} \log \ell_t(z_t) = \nabla_{z_t} \log I(z_t) - \nabla_{z_t} \log p_t(z_t). \tag{38}$$

Since q_t is the Gaussian kernel Eq. (15), $\nabla_{z_t} \log q_t(z_t \mid x) = (\alpha_t\, x - z_t)/\sigma_t^2$, so by the log-derivative rule $\nabla_{z_t} q_t = q_t\, (\alpha_t\, x - z_t)/\sigma_t^2$. Consequently

$$\begin{aligned}
\nabla_{z_t} \log I(z_t) = \frac{\nabla_{z_t} I(z_t)}{I(z_t)} &= \frac{\displaystyle\int \ell(\tau(x))\, \frac{\alpha_t\, x - z_t}{\sigma_t^2}\, q_t(z_t \mid x)\, p_\theta(x)\, \mathrm{d}x}{\displaystyle\int \ell(\tau(x))\, q_t(z_t \mid x)\, p_\theta(x)\, \mathrm{d}x} \\
&= \frac{\displaystyle\int \ell(\tau(x))\, \frac{\alpha_t\, x - z_t}{\sigma_t^2}\, p(x \mid z_t)\, \mathrm{d}x}{\displaystyle\int \ell(\tau(x))\, p(x \mid z_t)\, \mathrm{d}x} \\
&= \frac{1}{\sigma_t^2} \cdot \frac{\alpha_t \displaystyle\int x\, \ell(\tau(x))\, p(x \mid z_t)\, \mathrm{d}x \;-\; z_t\, \ell_t(z_t)}{\ell_t(z_t)} \\
&= \frac{1}{\sigma_t^2} \left(\alpha_t \int x\, \frac{\ell(\tau(x))\, p(x \mid z_t)}{\ell_t(z_t)}\, \mathrm{d}x \;-\; z_t \right) \\
&= \frac{1}{\sigma_t^2} \left(\alpha_t \int x\, p^\star(x \mid z_t)\, \mathrm{d}x \;-\; z_t \right) = \frac{\alpha_t\, \hat{z}^{\mathrm{tilt}} - z_t}{\sigma_t^2}.
\end{aligned} \tag{39}$$

The second term of Eq. (38) follows from the same steps applied to $p_t(z_t) = \int q_t(z_t \mid x)\, p_\theta(x)\, \mathrm{d}x$. Now the denominator is $p_t(z_t)$ itself, so dividing by it gives $p(x \mid z_t)$ at once, and the second member carries $\int p(x \mid z_t)\, \mathrm{d}x = 1$ in place of $\ell_t(z_t)$:

$$\begin{aligned}
\nabla_{z_t} \log p_t(z_t) = \frac{\nabla_{z_t} p_t(z_t)}{p_t(z_t)} &= \int \frac{\alpha_t\, x - z_t}{\sigma_t^2}\, \frac{q_t(z_t \mid x)\, p_\theta(x)}{p_t(z_t)}\, \mathrm{d}x \\
&= \frac{1}{\sigma_t^2} \left(\alpha_t \int x\, p(x \mid z_t)\, \mathrm{d}x \;-\; z_t \int p(x \mid z_t)\, \mathrm{d}x \right) = \frac{\alpha_t\, \hat{z} - z_t}{\sigma_t^2},
\end{aligned} \tag{40}$$

where $\hat{z} = \mathbb{E}_{p(\cdot \mid z_t)}[x]$; this is Tweedie's formula for this kernel and is the untilted score quoted in Proposition 2.

Substituting Eq. (39) and Eq. (40) into Eq. (38), the z_t terms — identical in both — cancel:

$$\nabla_{z_t} \log \ell_t(z_t) = \frac{\alpha_t\, \hat{z}^{\mathrm{tilt}} - z_t}{\sigma_t^2} - \frac{\alpha_t\, \hat{z} - z_t}{\sigma_t^2} = \frac{\alpha_t\big(\hat{z}^{\mathrm{tilt}} - \hat{z}\big)}{\sigma_t^2}, \tag{41}$$

which is Eq. (21). Adding Eq. (40) back recovers Eq. (22). □

A.5 Estimating the Tilted Mean

By Proposition 2, the guidance term depends on the frozen model only through the tilted posterior mean $\hat{z}^{\text{tilt}}$ of Definition 3.3. Its untilted counterpart $\hat{z}$ comes with the score (i), but $\hat{z}^{\text{tilt}}$ is a ratio of two expectations under $p(\cdot \mid z_t)$ with ℓ a black box, and has no closed form. This section gives the estimator used in Section 3.3, how its draws are obtained for each backbone, and the two parameters that set the strength of the tilt.

Proposition 5 (Self-normalized importance-sampling (SNIS) estimator)**.** *Draw N i.i.d. samples $x^{(n)} \sim p(x \mid z_t)$ and write $\ell_n \triangleq \ell(\tau(x^{(n)}))$. Then*

$$\hat{z}^{\text{tilt}} \approx \sum_{n=1}^{N} h_n \, x^{(n)}, \qquad h_n = \frac{\ell_n}{\sum_m \ell_m}, \qquad \sum_n h_n = 1. \tag{42}$$

Proof. We start by rewriting Definition 3.3. By Eq. (18), $p^\star(x \mid z_t) = p(x \mid z_t)\, \ell(\tau(x))/\ell_t(z_t)$, hence

$$\hat{z}^{\text{tilt}} = \frac{1}{\ell_t(z_t)} \int x \, p(x \mid z_t)\, \ell(\tau(x)) \, \mathrm{d}x = \frac{\mathbb{E}_{p(\cdot \mid z_t)}\big[\ell(\tau(x))\, x\big]}{\mathbb{E}_{p(\cdot \mid z_t)}\big[\ell(\tau(x))\big]}, \tag{43}$$

the last step by the definition of ℓ_t in Proposition 1. Neither expectation admits a closed form: ℓ is a scalar function of the whole decoded object and does not factorize over the coordinates of x, so numerator and denominator both couple all L positions. Estimating the two expectations of Eq. (43) by Monte Carlo, using the same N draws $x^{(n)} \sim p(\cdot \mid z_t)$ in both, gives

$$\hat{z}^{\text{tilt}} \approx \frac{\frac{1}{N}\sum_{n=1}^{N} \ell_n \, x^{(n)}}{\frac{1}{N}\sum_{m=1}^{N} \ell_m} = \sum_{n=1}^{N} \underbrace{\frac{\ell_n}{\sum_m \ell_m}}_{h_n} x^{(n)}, \qquad \sum_n h_n = 1. \tag{44}$$

The estimator is a ratio of two unbiased estimators sharing the same draws, hence consistent but biased at finite N, with bias $O(1/N)$; the self-normalization in h_n is what removes the unknown $\ell_t(z_t)$. □

Two knobs, and they are not the same operation. In Eq. (11) the acquisition enters with unit exponent; in practice the strength of the tilt is a tuning parameter. It can be adjusted at two points of the estimator: after the tilted mean has been formed, or inside the weights. The first, λ, scales the guidance term of Eq. (19) after the tilted mean has been formed, $\lambda \nabla_{z_t} \log \ell_t = \nabla_{z_t} \log \ell_t^{\lambda}$:

$$\nabla_{z_t} \log p_t + \lambda \, \nabla_{z_t} \log \ell_t = \frac{\alpha_t\big(\hat{z} + \lambda(\hat{z}^{\text{tilt}} - \hat{z})\big) - z_t}{\sigma_t^2} = \frac{\alpha_t \, \hat{z}^{\text{tilt},\gamma} - z_t}{\sigma_t^2}. \tag{45}$$

This leaves the draws and the weights of Eq. (42) untouched and moves only the target, along the segment from $\hat{z}$ to $\hat{z}^{\text{tilt}}$. Powering ℓ first and marginalizing second changes only the weights,

$$h_n^{(\gamma)} = \mathrm{softmax}_n\Big(\tfrac{1}{\gamma} \log \ell_n\Big) = \frac{\ell_n^{1/\gamma}}{\sum_m \ell_m^{1/\gamma}}, \qquad \ell \mapsto \ell^{1/\gamma}, \qquad \ell_t^{(\gamma)} = \mathbb{E}_{p(\cdot \mid z_t)}\big[\ell^{1/\gamma}\big]. \tag{46}$$

The target keeps the form of Eq. (22) and the weight vector is deformed instead: $\gamma \to 0$ collapses h onto the best sampled candidate, while $\gamma \to \infty$ flattens it to uniform and recovers the untilted $\hat{z}$.

B CDCD

Algorithm 1 covers the guided-sampling step when the backbone is a diffusion model.

Instantiation. CDCD's forward process Eq. (5) is the kernel Eq. (15) of Definition 3.2 at $(\alpha_t, \sigma_t) = (1, t)$, so $q_t(z_t \mid x) = \mathcal{N}(z_t; x, t^2 I)$ with $x = E(w)$. Proposition 2 then reads

$$\nabla_{z_t} \log p_t^\star(z_t) = \frac{1}{t^2}\big[(\hat{z} - z_t) + (\hat{z}^{\text{tilt}} - \hat{z})\big] = \frac{\hat{z}^{\text{tilt}} - z_t}{t^2}, \tag{47}$$

Table 2: What each backbone supplies to the guided score Eq. (19). Quantities (i) and (ii) are backbone-specific; the guidance term $\nabla_{z_t} \log \ell_t$ is estimated from the draws of (ii) (Section 3.3).

	CDCD	**SeqFlow**
noised variable	embedding x	flow latent $g_\theta^{-1}(x)$
schedule (α_t, σ_t)	$(1,\, t)$, VE	$\big(\rho_t,\ \sqrt{1-\rho_t^2}\big)$, VP Eq. (51)
(i) $\nabla_{z_t} \log p_t$	learned Eq. (6)	$-z_t$, exact
(ii) draws from $p(\cdot \mid z_t)$	mean-field $\tilde{p}$ Eq. (49), one network pass	exact Gaussian Eq. (27), N passes of g_θ
$\nabla_{z_t} \log \ell_t$	SNIS Eq. (42) or REINFORCE Eq. (50)	SNIS Eq. (42)
noise end	$p_T^\star \approx \mathcal{N}(0, T^2 I)$	$\pi_T^\star \approx \mathcal{N}(0, I)$
anchor scale σ	$\sigma = T$	$\sigma = 1$

Algorithm 1 GUIDEDSAMPLE — CDCD. Instantiates Section 3.4 at $(\alpha_t, \sigma_t) = (1, t)$. CDCD's code was never released, so we adapted a public reimplementation (Sacco, 2024).

Require: z_{init}; frozen net_θ and embedding table E; α, γ, λ, N, $\mathcal{D}$

1: $z_T \leftarrow z_{\text{init}}$
2: **for** $t \in [T, \ldots, 0]$, $\delta = t_{\text{curr}} - t_{\text{next}}$ **do**
3: $\quad$ logits, $\hat{z} \leftarrow \text{net}_\theta(z_t, t)$ ▷ *Eq. (6)*
4: $\quad$ $w^{(n)} \sim p_\theta(\cdot \mid z_t, t),\ n = 1..N$ ▷ *Eq. (7)*
5: $\quad$ $h_n \leftarrow \text{softmax}_n\big(\frac{1}{\gamma} \log \alpha(w^{(n)}; \mathcal{D})\big)$ ▷ *Eq. (46)*
6: $\quad$ $\hat{z}^{\text{tilt}} \leftarrow \sum_n h_n\, E(w^{(n)})$ ▷ *Proposition 5 or Proposition 6*
7: $\quad$ $z_{t-\delta} \leftarrow z_t + \delta\, t\, \dfrac{\hat{z} + \lambda(\hat{z}^{\text{tilt}} - \hat{z}) - z_t}{t^2}$ ▷ *Eq. (45)*
8: **end for**
9: **return** z_0

whose first term is CDCD's own score Eq. (6): guidance replaces $\hat{z}$ by $\hat{z}^{\text{tilt}}$ and leaves the sampler otherwise untouched. The sampler integrates Eq. (47) by an Euler step (Algorithm 1, line 7), which carries a discretization error per step — a cost SeqFlow does not pay (Appendix C).

With $x = E(w)$, Definition 3.3 reads

$$\hat{z}^{\text{tilt}} = \frac{\sum_{w \in \mathcal{V}^L} p_\theta(w \mid z_t)\, \ell(w)\, E(w)}{\sum_{w \in \mathcal{V}^L} p_\theta(w \mid z_t)\, \ell(w)}. \tag{48}$$

Both sums range over $\mathcal{V}^L$, and neither decomposes: ℓ is a scalar function of the whole decoded molecule, $\ell\big((w_1, \ldots, w_L)\big)$, so it couples all L positions. Note the contrast with $\hat{z}$, which is an expectation of a quantity that is additive across positions and is therefore determined by the marginals Eq. (7) alone; $\hat{z}^{\text{tilt}}$ is not, and must be estimated by Proposition 5.

Sampling the denoising posterior. This is where CDCD incurs its one structural approximation, and it is worth being precise about whose approximation it is. Proposition 5 needs draws from $p(\cdot \mid z_t)$. For a diffusion model, drawing from the true denoising posterior means running the reverse process from z_t all the way to the data end, once per sample: at N samples per guidance step and K steps per trajectory, that is NK rollouts, which is prohibitive. What is available for free is Eq. (7), the per-position marginals, since they come from the same forward pass that already produced $\hat{z}$ and drawing from a factorized categorical costs nothing. We therefore adopt the mean-field approximation

$$p(x \mid z_t) \;\approx\; \tilde{p}(x \mid z_t) \triangleq \textstyle\prod_{i=1}^{L} p_\theta(w_i \mid z_t, t), \qquad x = E(w), \tag{49}$$

and sample positions independently.

Two things should be said about it. First, it is *our* approximation, introduced in the estimator, not an assumption CDCD makes: the model never represents the joint because the score does not require

Table 3: The mean-field approximation Eq. (49) against rollouts of the reverse process. K is the number of inner denoising steps used to draw each of the N SNIS samples: $K = 0$ is the factorized draw from the per-position marginals Eq. (7), i.e. Eq. (49), and is what Algorithm 1 uses; $K > 0$ replaces it by a K-step ancestral rollout, which couples the positions through the network and resolves the joint. Final objective value after 300 BO epochs on six GuacaMol/TDC objectives, mean $\pm$ sd over 3 seeds; higher is better. Δ is K=32 minus K=0, and the last column expresses it in units of the pooled seed-to-seed sd. At $K = 32$ — beyond the point where proposal fidelity saturates, and at $\approx 24\times$ the wall clock (8.77 against 0.36 h per arm) — no objective improves beyond the seed spread and one degrades.

objective	$K = 0$	$K = 32$	Δ	Δ/sd
med2	0.1637 ± 0.0193	0.1694 ± 0.0243	$+0.0058$	$+0.26$
osmb	0.7094 ± 0.0578	0.7055 ± 0.0611	-0.0039	-0.07
adip	0.4335 ± 0.0297	0.4483 ± 0.0029	$+0.0148$	$+0.70$
zale	0.3811 ± 0.0935	0.2842 ± 0.0612	-0.0969	-1.23
rano	0.7503 ± 0.0361	0.7088 ± 0.0570	-0.0416	-0.87
qed	0.8992 ± 0.0370	0.9106 ± 0.0387	$+0.0114$	$+0.30$

it, $\hat{z}$ being determined by the marginals alone. Second, it is empirically benign — we compared Eq. (49) against exact rollouts of the reverse process and found no degradation in optimization performance, at a small fraction of the cost (Table 3). The SeqFlow version needs neither: its posterior Eq. (53) is exactly Gaussian and fully correlated, so a draw from it is a draw from the true posterior.

REINFORCE estimator. Proposition 5 estimates $\hat{z}^{\mathrm{tilt}}$ from draws of a posterior that is already available in closed form, so the conditional score is analytic and Monte Carlo is needed only for the sequence-level expectation of ℓ. That estimator works directly on Definition 3.3. A second, more direct route starts from the observation that Proposition 2 can be rearranged as $\hat{z}^{\mathrm{tilt}} = \hat{z} + t^2 \nabla_{z_t} \log \ell_t(z_t)$. Since $\hat{z}$ is available from the forward pass of the model, the only remaining obstacle is $\nabla_{z_t} \log \ell_t(z_t)$.

Proposition 6 (REINFORCE estimator)**.** *Given $\hat{z}^{\mathrm{tilt}} = \hat{z} + t^2 \nabla_{z_t} \log \ell_t(z_t)$, its estimate may be written as*

$$\hat{z}^{\mathrm{tilt}} \approx \hat{z} + t^2 \sum_{n=1}^{N} h_n \nabla_{z_t} \log p_\theta(w^{(n)} \mid z_t), \qquad h_n = \frac{\ell(w^{(n)})}{\sum_m \ell(w^{(m)})}. \tag{50}$$

Proof. By Eq. (17), $\ell_t(z_t) = \int \ell(\tau(x))\, p(x \mid z_t)\, \mathrm{d}x = \sum_{w \in \mathcal{V}^L} \ell(w)\, p_\theta(w \mid z_t)$. Applying the log-derivative trick, i.e. multiplying and dividing by p_θ,

$$\nabla_{z_t} \ell_t = \sum_{w \in \mathcal{V}^L} \ell(w)\, p_\theta \frac{\nabla_{z_t} p_\theta}{p_\theta} = \mathbb{E}_{p_\theta}\big[\ell\, \nabla_{z_t} \log p_\theta\big]$$

$$\nabla_{z_t} \log \ell_t = \frac{\mathbb{E}_{p_\theta}\big[\ell\, \nabla_{z_t} \log p_\theta\big]}{\mathbb{E}_{p_\theta}\big[\ell\big]} = \frac{\displaystyle\sum_{w \in \mathcal{V}^L} \nabla_{z_t} \log p_\theta(w \mid z_t)\, \ell(w)\, p_\theta(w \mid z_t)}{\displaystyle\sum_{w \in \mathcal{V}^L} p_\theta(w \mid z_t)\, \ell(w)}.$$

This is again a ratio of two expectations under the same p_θ, with the same intractable denominator ℓ_t, so Monte Carlo applies as in Proposition 5:

$$\nabla_{z_t} \log \ell_t \approx \frac{\frac{1}{N}\sum_{n=1}^{N} \ell_n \nabla_{z_t} \log p_\theta(w^{(n)} \mid z_t)}{\frac{1}{N}\sum_{m=1}^{N} \ell_m} = \sum_{n=1}^{N} \underbrace{\frac{\ell_n}{\sum_m \ell_m}}_{h_n} \nabla_{z_t} \log p_\theta(w^{(n)} \mid z_t), \qquad \sum_n h_n = 1.$$

Substituting into $\hat{z}^{\mathrm{tilt}} = \hat{z} + t^2 \nabla_{z_t} \log \ell_t(z_t)$ gives Eq. (50). □

The per-sample score required by Eq. (50) is itself available in closed form.

Table 4: Hyperparameters of the CDCD generator.

Hyperparameter	Value
Architecture	
Embedding dimension	16
Hidden dimension	512
Query/key/value dimension	512
Time embedding dimension	16
Layers	8
Attention heads	8
Conditioning	
Self-conditioning probability	0.4
Mask-conditioning probability	0.1
Masking probability	1.0
Loss	
Objective	Focal (Lin et al., 2020)
`<nop>` token weight	1.0
`<eos>` token weight	1.0
`<pad>` token weight	1.0
Noise schedule	
Minimum time	0.01
Maximum time	200.0
Location	3.0
Scale	2.0
Height	6.0
Offset	-1.0
Update frequency	100
Optimization	
Batch size	512
Learning rate	5×10^{-6}
Maximum epochs	30
Maximum steps	450,000
Gradient clipping	1.0
Gradient accumulation steps	1
Sequence and sampling	
Maximum sequence length	110
Integration steps	200

Cost. Both $\hat{z}$ and the categorical $p_\theta(w \mid z_t, t)$ come from a single forward pass of net_θ, so the N draws of Proposition 5 are sampled from an object already in hand and cost no additional network evaluations. Each step is therefore one network evaluation, N decodes, and one GP prediction over N molecules.

B.1 Training CDCD

No pretrained diffusion model was available off the shelf, so we trained one from scratch. Table 4 lists the hyperparameter configuration we used. Dieleman et al. (2022) never released the original code, so we started from a public reimplementation (Sacco, 2024) and adapted it to our needs.

C SeqFlow

Algorithm 2 covers the guided-sampling step when the backbone is a normalizing flow.

Instantiation. SeqFlow supplies no noising process of its own, so the kernel of Definition 3.2 is imposed rather than inherited. We work in the flow latent $z = g_\theta^{-1}(x)$, whose law is exactly $\mathcal{N}(0, I)$ while the tilt is the pullback $\ell\big(\tau(g_\theta(z))\big)$, Eq. (25). Set

$$q_t(z_t \mid z) = \mathcal{N}\big(z_t;\, \rho_t\, z,\, (1-\rho_t^2)I\big), \qquad \rho_t \in (0, 1], \tag{51}$$

Algorithm 2 GUIDEDSAMPLE — SeqFlow. Instantiates Section 3.4 in the flow latent with the variance-preserving schedule Eq. (51).

Require: z_{init}; frozen flow g_θ and readout τ; schedule $\rho_T < \cdots < \rho_0$; α, γ, λ, N, $\mathcal{D}$

$z_T \leftarrow z_{\text{init}}$
for $t = T, \dots, 0,\ t'$ the next level **do**
  $\hat{z} \leftarrow \rho_t\, z_t$ ▷ *closed form; no network eval, Eq. (53)*
  $z^{(n)} \sim \mathcal{N}\big(\hat{z}, (1-\rho_t^2)I\big),\ n = 1..N$ ▷ *exact posterior*
  $h_n \leftarrow \mathrm{softmax}_n\big(\frac{1}{\gamma} \log \alpha(\tau(g_\theta(z^{(n)})); \mathcal{D})\big)$ ▷ *Eq. (46)*
  $\hat{z}^{\text{tilt}} \leftarrow \sum_n h_n\, z^{(n)}$ ▷ *Proposition 5*
  $z_{t'} \leftarrow \rho_{t'}\big(\hat{z} + \lambda(\hat{z}^{\text{tilt}} - \hat{z})\big) + \sqrt{1-\rho_{t'}^2}\,\xi,\ \xi \sim \mathcal{N}(0, I)$
end for
return z_0, decoded as $\tau(g_\theta(z_0))$

i.e. $(\alpha_t, \sigma_t) = \big(\rho_t, \sqrt{1-\rho_t^2}\big)$ in Eq. (26). Two closed forms follow. Since the latent law is $\mathcal{N}(0, I)$, the untilted marginal is

$$p_t(z_t) = \int \mathcal{N}\big(z_t; \rho_t z, (1-\rho_t^2)I\big)\, \mathcal{N}(z; 0, I)\, dz = \mathcal{N}(z_t; 0, I), \tag{52}$$

so the bridge is variance preserving and every level carries the same law as the flow's own base; and by Gaussian conjugacy the posterior of Proposition 1 is exact,

$$p(z \mid z_t) = \mathcal{N}\big(z;\ \rho_t\, z_t,\ (1-\rho_t^2)I\big), \qquad \hat{z} = \rho_t\, z_t, \tag{53}$$

which is Eq. (27) with $\alpha_t = \rho_t$. Neither requires a network evaluation. Proposition 2 then reads

$$\nabla_{z_t} \log p_t^\star(z_t) = \frac{\rho_t\, \hat{z}^{\text{tilt}} - z_t}{1-\rho_t^2}. \tag{54}$$

Nothing about g_θ enters any of Eq. (52)–Eq. (54): the learned content of the flow is confined to the tilt, where it is evaluated pointwise by Proposition 5 and never differentiated. This is why no analogue of Proposition 6 is needed here — there is no learned posterior whose score would have to be estimated.

Why tilting the latent is legitimate. Definition 3.1 is stated over the object that ℓ scores, whereas Eq. (51) tilts z. The two agree by the argument of Appendix A.1, applied with τ replaced by the composite readout $\tau \circ g_\theta$: writing Π for the distribution over molecules induced by $\pi^\star(z) \propto \mathcal{N}(z; 0, I)\, \ell(\tau(g_\theta(z)))$ of Eq. (25), and using that ℓ is constant on each cell A_w of Eq. (8),

$$\Pi(w) = \frac{1}{Z} \int_{A_w} \mathcal{N}(z; 0, I)\, \ell\big(\tau(g_\theta(z))\big)\, dz = \frac{\ell(w)}{Z} \int_{A_w} \mathcal{N}(z; 0, I)\, dz = \frac{P_\theta(w)\, \ell(w)}{Z}, \tag{55}$$

the intended target. Tilting the latent and decoding is therefore the same distribution over molecules as tilting the molecules directly; the constancy of ℓ on cells is what licenses pulling it out of the integral.

What remains approximate. Three things, none of them the posterior. First, the N-sample estimate of $\hat{z}^{\text{tilt}}$ in Proposition 5, which is common to both backbones. Second, Algorithm 2 substitutes the tilted posterior *mean* $\hat{z}^{\text{tilt}}$ for a *draw* from $p^\star(\cdot \mid z_t)$, which is under-dispersed and mode-seeking; resampling a single $z^{(n)}$ with probability h_n instead recovers a transition that is asymptotically exact in N, at the cost of variance. Third, the rollout is initialized at $z_T \sim p_T = \mathcal{N}(0, I)$ rather than at $\pi_T^\star$, which by Proposition 1 differs from p_T by the tilt ℓ_T; the discrepancy vanishes as $\rho_T \to 0$ and is the price of $\pi_T^\star$ not being samplable.

Cost. Each step draws N latents from Eq. (53) and decodes each through g_θ and τ, so the flow is evaluated N times per step. CDCD amortizes a single forward pass across all N draws, since Eq. (7) is already available from it. This is the price of the exact posterior: the flow has no amortized categorical to sample from, and every draw must be pushed through the network to be scored. The decode is latency-bound rather than throughput-bound, however, so N is nearly free and the step count is the binding constraint — the opposite of CDCD's economics.

Algorithm 3 AFGG: full BO loop

Require: frozen generator; acquisition α; temperature γ; guidance scale λ; inner samples N; batch size q; oracle budget B; initial dataset $\mathcal{D}_0$; trust-region coefficient ρ; anchor scale σ

$\mathcal{D} \leftarrow \mathcal{D}_0$
for $r = 1, 2, \ldots, \lfloor B/q \rfloor$ **do**
  fit GP on $\mathcal{D}$ ▷ *in design space, Section 2.1*
  $z^\star \leftarrow$ anchor from the incumbent
  $z_{\text{init}}^{(1:q)} \leftarrow \rho\, z^\star + \sqrt{1-\rho^2}\,\varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ ▷ *Eq. (28)*
  $z^{(1:q)} \leftarrow \textsc{GuidedSample}\big(z_{\text{init}}^{(1:q)}\big)$ ▷ *Algorithm 1 (CDCD) or Algorithm 2 (SeqFlow)*
  $w^{(i)} \leftarrow \tau(z^{(i)})$ (CDCD) or $\tau\big(g_\theta(z^{(i)})\big)$ (SeqFlow), $i = 1..q$ ▷ *decode to design space*
  $y^{(i)} \leftarrow$ evaluate oracle on $w^{(i)}$, $i = 1..q$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(w^{(i)}, y^{(i)})\}_{i=1}^{q}$
end for
return $\arg\max_{(w,y)\in\mathcal{D}} y$

Abbreviation	PMO objective	Abbreviation	PMO objective
albu_sim	albuterol_similarity	osim_mpo	osimertinib_mpo
amlo_mpo	amlodipine_mpo	peri_mpo	perindopril_mpo
cele_red	celecoxib_rediscovery	rano_mpo	ranolazine_mpo
deco_hop	deco_hop	logp	rdkit_logp
drd2	drd2_docking	qed	rdkit_qed
fexo_mpo	fexofenadine_mpo	sa	sa_tdc
gsk3b	gsk3_beta	scaf_hop	scaffold_hop
iso_c7	isomer_c7h8n2o2	sita_mpo	sitagliptin_mpo
iso_c9	isomer_c9h10n2o2pf2cl	thio_red	thiothixene_rediscovery
jnk3	jnk3	trog_red	troglitazone_rediscovery
med_1	median_1	vals_sma	valsartan_smarts
med_2	median_2	zale_mpo	zaleplon_mpo
mest_sim	mestranol_similarity		

Table 5: Abbreviations of the 25 PMO objectives, used in all tables.

C.1 Training SeqFlow

We used the pretrained model available in the authors' repository. For details of how it was trained and which hyperparameter configuration was used, we refer to Lee et al. (2025).

D AFGG Algorithm

Algorithm 3 summarizes the complete AFGG optimization loop. At each BO iteration, the surrogate is updated in data space, the guided sampler is initialized around the current incumbent using the trust region above, and the resulting candidate is decoded and evaluated. The overall procedure is shared across both generative backbones; they differ only in the continuous variable in which the bridge and guidance are applied. For CDCD, this variable is the token-embedding representation, whereas for SeqFlow it is the Gaussian flow latent. The complete guided samplers are given in Appendix B and Appendix C (Algorithm 1 and Algorithm 2), and Table 2 summarizes the correspondence between the two instantiations.

E Additional Material

E.1 Additional Results on GuacaMol

This section reports the full benchmark behind Section 4: all 25 PMO objectives, in the sequential regime and in both batch regimes, as final scores and as complete trajectories. Objective names are abbreviated as in Table 5.

	baselines											AFGG	
Objective	CoBO	InvBO	LOL-BO	NFBO	CMA-ES	COWBOYS	DE	GA	HC	RLBO	TuRBO	CDCD	SeqFlow
albu_sim	0.593 ± 0.111	0.629 ± 0.079	0.509 ± 0.065	0.593 ± 0.119	0.500 ± 0.030	0.448 ± 0.037	0.488 ± 0.035	0.385 ± 0.074	0.454 ± 0.070	0.483 ± 0.091	0.475 ± 0.047	**0.691 ± 0.073**	**0.975 ± 0.038**
amlo_mpo	0.523 ± 0.012	0.538 ± 0.050	0.516 ± 0.022	0.563 ± 0.095	0.459 ± 0.038	0.472 ± 0.038	0.438 ± 0.143	0.419 ± 0.082	0.460 ± 0.048	0.491 ± 0.030	0.475 ± 0.024	0.546 ± 0.039	**0.565 ± 0.045**
cele_red	0.262 ± 0.041	0.286 ± 0.068	0.243 ± 0.035	0.455 ± 0.035	0.227 ± 0.032	0.243 ± 0.020	0.216 ± 0.036	0.210 ± 0.040	0.228 ± 0.032	0.224 ± 0.031	0.221 ± 0.033	0.392 ± 0.024	**0.836 ± 0.125**
deco_hop	0.565 ± 0.008	0.597 ± 0.019	0.560 ± 0.009	0.673 ± 0.147	0.563 ± 0.010	0.559 ± 0.008	0.571 ± 0.018	0.549 ± 0.015	0.553 ± 0.012	0.556 ± 0.015	0.552 ± 0.012	0.599 ± 0.010	**0.803 ± 0.175**
drd2	0.510 ± 0.370	0.963 ± 0.037	0.626 ± 0.184	0.971 ± 0.066	0.184 ± 0.068	0.796 ± 0.243	0.862 ± 0.156	0.154 ± 0.051	0.183 ± 0.045	0.380 ± 0.246	0.556 ± 0.337	0.730 ± 0.226	**1.000 ± 0.000**
fexo_mpo	0.710 ± 0.034	0.734 ± 0.027	0.716 ± 0.021	0.723 ± 0.032	0.672 ± 0.014	0.683 ± 0.031	0.724 ± 0.045	0.611 ± 0.063	0.680 ± 0.033	0.682 ± 0.034	0.688 ± 0.026	**0.763 ± 0.017**	**0.741 ± 0.016**
gsk3b	0.362 ± 0.069	0.500 ± 0.042	0.336 ± 0.103	0.656 ± 0.174	0.371 ± 0.042	0.418 ± 0.093	0.478 ± 0.108	0.248 ± 0.081	0.320 ± 0.047	0.343 ± 0.084	0.328 ± 0.047	0.518 ± 0.088	**0.950 ± 0.037**
iso_c7	0.875 ± 0.145	**1.000 ± 0.000**	0.760 ± 0.036	0.860 ± 0.092	0.972 ± 0.051	0.919 ± 0.046	0.808 ± 0.051	0.808 ± 0.051	0.895 ± 0.090	0.898 ± 0.072	0.898 ± 0.072	0.957 ± 0.049	0.981 ± 0.043
iso_c9	0.577 ± 0.050	0.699 ± 0.147	0.630 ± 0.082	0.758 ± 0.024	0.751 ± 0.074	**0.834 ± 0.020**	0.672 ± 0.146	0.643 ± 0.103	0.687 ± 0.106	0.714 ± 0.035	0.778 ± 0.049	0.722 ± 0.079	0.809 ± 0.081
jnk3	0.148 ± 0.031	0.206 ± 0.063	0.144 ± 0.035	0.412 ± 0.095	0.150 ± 0.025	0.176 ± 0.032	0.190 ± 0.053	0.142 ± 0.029	0.144 ± 0.025	0.150 ± 0.049	0.158 ± 0.029	0.216 ± 0.033	**0.778 ± 0.038**
med_1	0.178 ± 0.025	0.253 ± 0.037	0.175 ± 0.023	0.203 ± 0.024	0.146 ± 0.012	0.192 ± 0.024	0.179 ± 0.038	0.152 ± 0.042	0.154 ± 0.015	0.150 ± 0.023	0.181 ± 0.030	**0.346 ± 0.031**	**0.352 ± 0.045**
med_2	0.158 ± 0.011	0.186 ± 0.014	0.158 ± 0.011	0.207 ± 0.018	0.155 ± 0.013	0.156 ± 0.012	0.168 ± 0.017	0.155 ± 0.013	0.155 ± 0.013	0.155 ± 0.013	0.155 ± 0.013	**0.225 ± 0.016**	**0.283 ± 0.024**
mest_sim	0.421 ± 0.064	0.531 ± 0.054	0.384 ± 0.059	0.555 ± 0.125	0.362 ± 0.062	0.404 ± 0.023	0.382 ± 0.104	0.280 ± 0.090	0.358 ± 0.058	0.372 ± 0.046	0.395 ± 0.057	**0.593 ± 0.051**	**0.658 ± 0.114**
osim_mpo	0.761 ± 0.013	0.782 ± 0.024	0.767 ± 0.015	**0.803 ± 0.019**	0.750 ± 0.020	0.749 ± 0.023	0.756 ± 0.067	0.724 ± 0.060	0.744 ± 0.028	0.749 ± 0.027	0.748 ± 0.022	0.778 ± 0.023	0.789 ± 0.034
peri_mpo	0.487 ± 0.049	0.447 ± 0.054	0.416 ± 0.079	0.434 ± 0.015	0.345 ± 0.053	0.344 ± 0.034	0.355 ± 0.067	0.332 ± 0.055	0.344 ± 0.034	0.356 ± 0.023	0.342 ± 0.038	0.432 ± 0.031	**0.502 ± 0.017**
rano_mpo	0.688 ± 0.027	0.742 ± 0.033	0.684 ± 0.042	0.696 ± 0.053	0.690 ± 0.071	0.708 ± 0.040	0.729 ± 0.068	0.484 ± 0.081	0.710 ± 0.043	0.672 ± 0.103	0.728 ± 0.025	**0.757 ± 0.034**	**0.785 ± 0.007**
logp	**65.866 ± 10.720**	**49.799 ± 19.126**	34.741 ± 5.318	25.055 ± 8.705	13.285 ± 1.014	15.443 ± 3.623	7.589 ± 3.102	6.138 ± 0.827	11.939 ± 1.801	10.491 ± 2.039	12.128 ± 1.918	22.684 ± 2.667	35.734 ± 10.360
qed	0.928 ± 0.004	0.942 ± 0.006	0.934 ± 0.003	0.943 ± 0.008	0.888 ± 0.032	0.898 ± 0.034	0.888 ± 0.034	0.857 ± 0.047	0.889 ± 0.038	0.882 ± 0.045	0.896 ± 0.021	0.940 ± 0.006	**0.945 ± 0.002**
sa	**9.001 ± 0.347**	8.393 ± 0.543	8.214 ± 0.711	7.538 ± 0.452	8.035 ± 0.069	8.020 ± 0.363	8.492 ± 0.742	**8.853 ± 0.151**	8.053 ± 0.225	8.035 ± 0.128	7.971 ± 0.035	8.349 ± 0.606	8.560 ± 0.220
scaf_hop	0.441 ± 0.029	0.501 ± 0.030	0.430 ± 0.020	0.510 ± 0.062	0.431 ± 0.008	0.442 ± 0.027	0.441 ± 0.027	0.407 ± 0.023	0.413 ± 0.018	0.417 ± 0.027	0.427 ± 0.029	0.500 ± 0.028	**0.583 ± 0.070**
sita_mpo	0.255 ± 0.096	**0.330 ± 0.032**	0.228 ± 0.087	0.290 ± 0.030	**0.367 ± 0.037**	**0.352 ± 0.090**	0.275 ± 0.095	0.094 ± 0.050	**0.400 ± 0.101**	**0.319 ± 0.118**	0.275 ± 0.063	0.108 ± 0.092	0.295 ± 0.056
thio_red	0.270 ± 0.025	0.308 ± 0.017	0.255 ± 0.011	0.376 ± 0.053	0.250 ± 0.015	0.261 ± 0.010	0.272 ± 0.039	0.240 ± 0.021	0.241 ± 0.019	0.245 ± 0.014	0.250 ± 0.014	**0.395 ± 0.051**	**0.475 ± 0.060**
trog_red	0.205 ± 0.004	0.260 ± 0.010	0.218 ± 0.044	0.300 ± 0.044	0.206 ± 0.017	0.210 ± 0.014	0.205 ± 0.033	0.182 ± 0.028	0.194 ± 0.016	0.196 ± 0.016	0.190 ± 0.017	**0.324 ± 0.043**	**0.380 ± 0.031**
vals_sma	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	**0.015 ± 0.034**	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
zale_mpo	0.404 ± 0.055	0.461 ± 0.047	0.348 ± 0.065	0.442 ± 0.042	0.390 ± 0.037	0.378 ± 0.044	0.361 ± 0.123	0.191 ± 0.068	0.365 ± 0.037	0.357 ± 0.083	0.355 ± 0.041	**0.466 ± 0.037**	**0.480 ± 0.026**

Table 6: Final best score on all 25 PMO objectives after 300 oracle calls (10 initial points, 1 point per BO iteration), mean ± sample standard deviation over 5 seeds. AFGG is shown in two instantiations: **CDCD** and **SeqFlow**. Every cell is judged against the opposing group: an AFGG cell against the 11 baselines, a baseline cell against the two AFGG columns. **Bold**: the cell's mean exceeds every mean in the opposing group. Green : that lead is significant, exceeding the cell's own standard deviation; bold without green therefore means ahead but within noise.

All available generative-model checkpoints of the baselines were trained on GuacaMol; COWBOYS provides no checkpoint, so we retrain it on GuacaMol from the released code.

One accounting point applies throughout. CoBO (Lee et al., 2023) and LOL-BO (Maus et al., 2022) spend oracle calls realigning a latent with the molecule it decodes to, and their released code does not charge those calls to the budget. Following Chu et al. (2024) we modified both to count them, so every method below is compared at the same number of oracle evaluations.

Fig. 3 and Table 6 give the sequential regime ($q = 1$, 10 initial points, 300 oracle calls) against all eleven baselines, of which Table 1 in the main text shows five. Each cell is judged against the opposing group, so a bold AFGG cell beats every baseline on that objective and a bold baseline cell beats both AFGG columns; green marks a lead exceeding the cell's own standard deviation. Two objectives are worth reading separately from the rest. `logp` and `sa` are not scaled to be in $[0, 1]$, and reward molecules that are large or synthetically implausible, so a high score there measures how far a method will leave the training distribution rather than how well it optimizes; the LSBO baselines lead on both.

The batch regime (100 initial points, then 100 epochs of $q = 5$) admits two ways of turning a per-candidate tilt into a batch, differing in where the interaction between members enters. *Uncoupled* generation guides each trajectory by its own $\ell = \alpha^{\mathrm{EI}}$ and lets the points "meet" only at selection, where the q points are chosen from the generated pool by joint qLogEI in design space. *Coupled* generation uses one joint tilt per batch, ℓ being qLogEI evaluated on the partial batch, so each member is generated aware of those already fixed. Coupled generation is the more faithful instantiation of batch BO — it is the one that targets the joint acquisition the batch is eventually scored by — but it is not the one that performs better here, so we report both. Fig. 4 and Table 7 give the uncoupled variant, the one used in Section 4; Fig. 5 and Table 8 the coupled one. The baseline columns are the same runs in both tables, since only the AFGG variants differ.

E.2 Effect of the Generative Prior

AFGG (CDCD) and AFGG (SeqFlow) share the same acquisition-guided machinery, but differ in the generative model being steered: Table 9 isolates that component: we draw 5000 unconditional samples from each model and score them against a 30k-molecule subset of GuacaMol. Validity, uniqueness and novelty do not separate the two models: all six values exceed 0.96 and the residual differences, below two points, are in CDCD's favor. FCD does separate them, by a factor of roughly 3.5: SeqFlow reaches 6.02 against CDCD's 20.99, i.e. its samples sit far closer to the reference distribution in ChemNet feature space. Guidance tilts the prior rather than extending its support, so what the acquisition function can propose is confined to the region the prior already covers, and a prior whose mass is better aligned with the data distribution gives the search a better-conditioned region to act on.

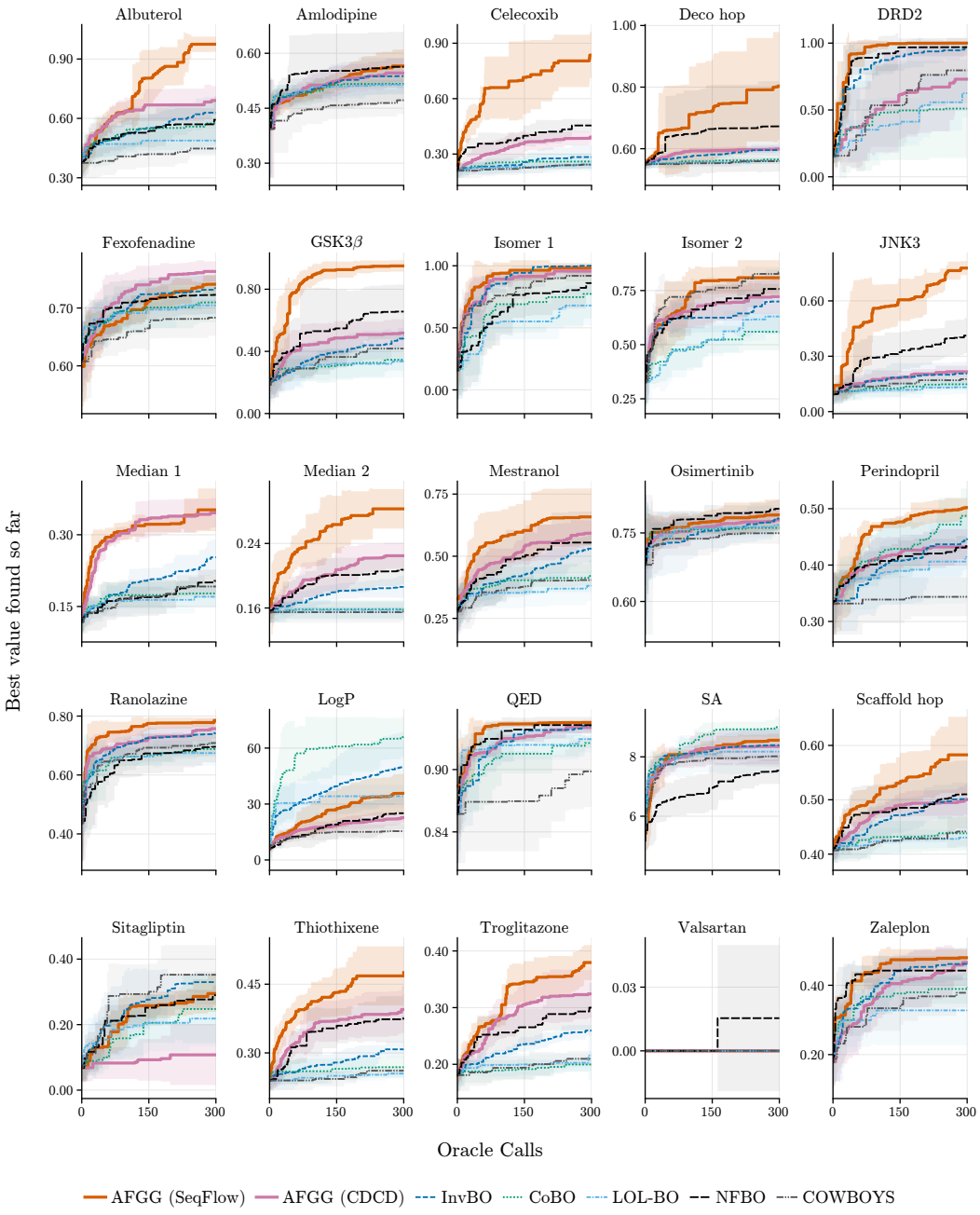

Figure 3: **Sequential setting**: full trajectories on all 25 PMO objectives (10 initial points, 300 oracle calls, 1 point per BO iteration).

E.3 Hyperparameter Tuning for CDCD

Every column reports the final best score after 300 oracle calls, mean ± sample standard deviation over 5 seeds. All ablations use the CDCD backbone in the sequential setting (10 initial points, 300 oracle calls, 1 point per epoch). Every objective is scored on $[0, 1]$; bold marks the best mean in each row (ties at the printed precision are all bolded), and *Wins* counts only rows with a unique winner.

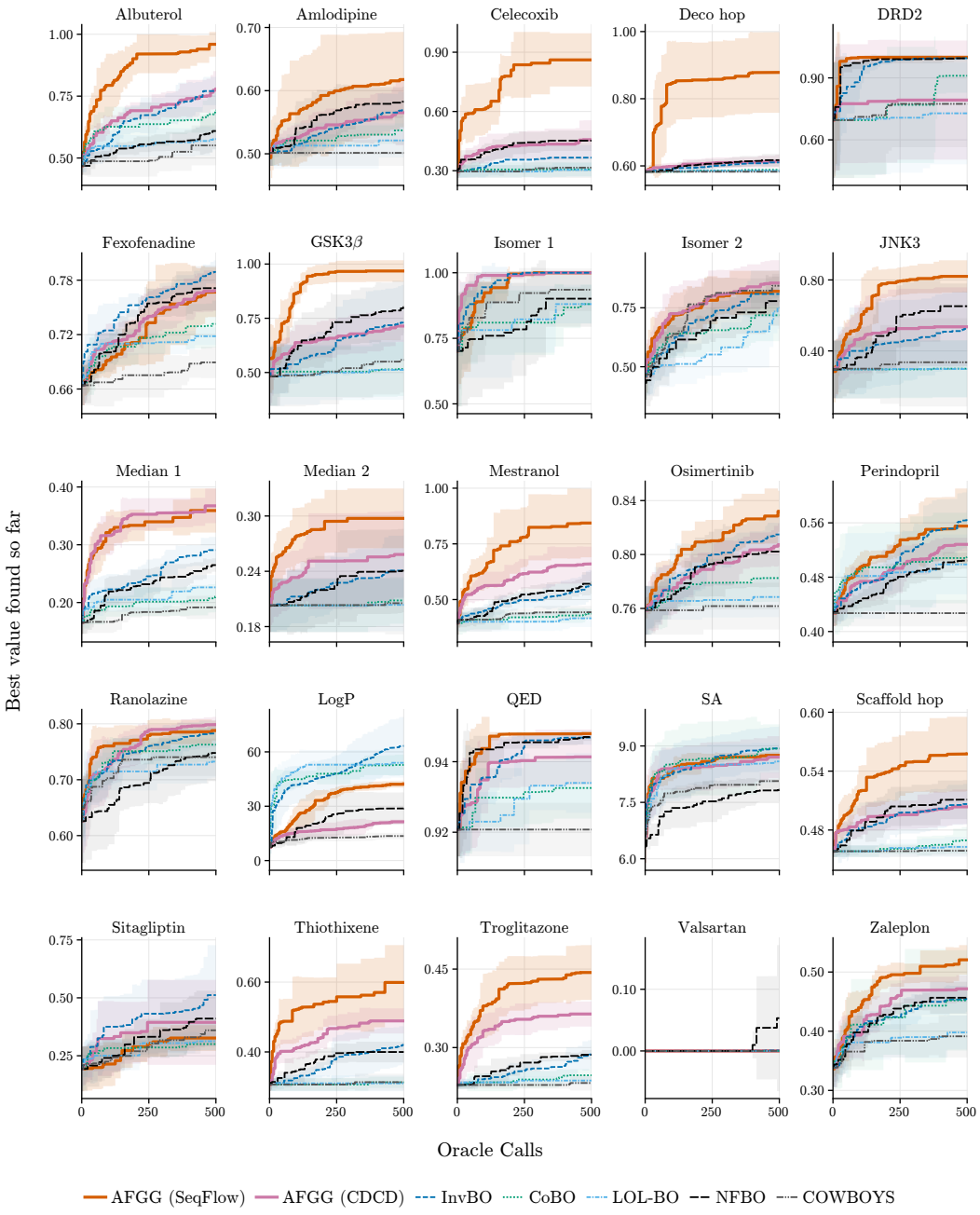

Figure 4: **Batch regime, uncoupled generation**: 500 acquisitions (100 initial points, then 100 epochs of 5), all 25 objectives.

Guidance on vs. off. By setting $\lambda = 0$, it has been proven that the generative model is not the main driver for optimizing the function. Having an additional force, represented by the acquisition function, has a dramatic effect on performance (see Table 10 for full results).

Guidance estimator. SNIS and REINFORCE target the same quantity, $\hat{z}^{\text{tilt}}$ (Proposition 6 and Proposition 5). SNIS forms it directly as a weighted average of the sampled embeddings, $\sum_n h_n E(w^{(n)})$: a closed-form combination with no gradient involved Eq. (42). REINFORCE instead routes through Eq. (50), a score-function estimator — these are well known to carry substan-

	baselines					AFGG	
Objective	CoBO	InvBO	LOL-BO	NFBO	COWBOYS	CDCD	SeqFlow
albu_sim	0.692 ± 0.095	0.771 ± 0.057	0.579 ± 0.043	0.609 ± 0.073	0.551 ± 0.081	**0.779 ± 0.073**	**0.960 ± 0.050**
amlo_mpo	0.540 ± 0.021	0.569 ± 0.029	0.522 ± 0.026	0.582 ± 0.024	0.501 ± 0.021	0.565 ± 0.052	**0.618 ± 0.075**
cele_red	0.307 ± 0.035	0.366 ± 0.064	0.304 ± 0.036	0.452 ± 0.052	0.316 ± 0.052	**0.457 ± 0.098**	**0.861 ± 0.135**
deco_hop	0.588 ± 0.005	0.610 ± 0.007	0.586 ± 0.004	0.617 ± 0.018	0.583 ± 0.004	0.614 ± 0.019	**0.878 ± 0.120**
drd2	0.930 ± 0.090	0.996 ± 0.006	0.728 ± 0.247	0.994 ± 0.012	0.774 ± 0.210	0.792 ± 0.288	**1.000 ± 0.000**
fexo_mpo	0.732 ± 0.025	**0.789 ± 0.013**	0.718 ± 0.010	**0.771 ± 0.025**	0.689 ± 0.017	0.768 ± 0.022	0.767 ± 0.028
gsk3b	0.516 ± 0.125	0.730 ± 0.174	0.512 ± 0.138	0.802 ± 0.120	0.558 ± 0.151	0.714 ± 0.073	**0.968 ± 0.050**
iso_c7	0.880 ± 0.074	1.000 ± 0.000	0.880 ± 0.074	0.901 ± 0.107	0.935 ± 0.081	1.000 ± 0.000	1.000 ± 0.000
iso_c9	0.718 ± 0.110	0.808 ± 0.100	0.758 ± 0.074	0.778 ± 0.104	0.843 ± 0.051	**0.855 ± 0.097**	0.820 ± 0.087
jnk3	0.300 ± 0.161	0.524 ± 0.211	0.298 ± 0.161	0.652 ± 0.128	0.336 ± 0.248	0.536 ± 0.222	**0.820 ±0.091**
med_1	0.212 ± 0.026	0.291 ± 0.023	0.228 ± 0.022	0.265 ± 0.012	0.192 ± 0.020	**0.367 ± 0.030**	**0.359 ± 0.038**
med_2	0.211 ± 0.024	0.241 ± 0.037	0.204 ± 0.028	0.241 ± 0.045	0.205 ± 0.034	**0.258 ± 0.047**	**0.297 ± 0.032**
mest_sim	0.439 ± 0.020	0.565 ± 0.068	0.428 ± 0.038	0.577 ± 0.052	0.443 ± 0.028	**0.660 ± 0.077**	**0.844 ± 0.154**
osim_mpo	0.786 ± 0.019	0.815 ± 0.009	0.768 ± 0.015	0.803 ± 0.027	0.762 ± 0.017	0.807 ± 0.016	**0.832 ± 0.015**
peri_mpo	0.509 ± 0.069	**0.567 ± 0.041**	0.499 ± 0.044	0.504 ± 0.012	0.427 ± 0.019	0.528 ± 0.010	0.555 ± 0.055
rano_mpo	0.763 ± 0.031	0.784 ± 0.025	0.732 ± 0.028	0.748 ± 0.051	0.740 ± 0.033	**0.799 ± 0.014**	**0.788 ± 0.018**
logp	**52.790 ± 5.679**	**63.296 ± 16.073**	**53.962 ± 8.208**	29.014 ± 8.537	13.554 ± 1.710	21.420 ± 1.219	42.342 ± 1.350
qed	0.934 ± 0.008	0.947 ± 0.002	0.934 ± 0.008	0.947 ± 0.001	0.921 ± 0.008	0.941 ± 0.008	**0.948 ± 0.000**
sa	**8.952 ± 0.668**	**8.947 ± 0.581**	8.590 ± 0.473	7.841 ± 0.310	8.066 ± 0.225	8.703 ± 0.557	8.752 ± 0.083
scaf_hop	0.479 ± 0.027	0.506 ± 0.021	0.463 ± 0.010	0.511 ± 0.020	0.459 ± 0.005	0.504 ± 0.018	**0.557 ± 0.038**
sita_mpo	0.310 ± 0.036	**0.512 ± 0.216**	0.319 ± 0.032	**0.410 ± 0.121**	0.360 ± 0.047	0.394 ± 0.183	0.327 ± 0.056
thio_red	0.327 ± 0.035	0.420 ± 0.053	0.311 ± 0.022	0.400 ± 0.030	0.313 ± 0.024	**0.489 ± 0.036**	**0.599 ± 0.107**
trog_red	0.250 ± 0.009	0.287 ± 0.028	0.237 ± 0.008	0.286 ± 0.009	0.232 ± 0.014	**0.364 ± 0.024**	**0.443 ± 0.052**
vals_sma	0.000 ± 0.000	**0.001 ± 0.002**	0.000 ± 0.000	**0.053 ± 0.118**	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
zale_mpo	0.454 ± 0.083	0.455 ± 0.022	0.401 ± 0.027	0.457 ± 0.027	0.391 ± 0.036	**0.472 ± 0.023**	**0.521 ± 0.025**

Table 7: **Batch regime, uncoupled generation.** Final best score on all 25 PMO objectives after 500 acquisitions (100 initial points, then 100 epochs of 5), mean ± sample standard deviation over 5 seeds. AFGG generates each batch *uncoupled* here: every molecule is guided by its own acquisition value, and the batch only forms at selection. Every cell is judged against the opposing group: an AFGG cell against the five baselines, a baseline cell against the two AFGG columns. **Bold**: the cell's mean exceeds every mean in the opposing group. Green : that lead is significant, exceeding the cell's own standard deviation; bold without green therefore means ahead but within noise.

tially higher variance than a direct, closed-form average for a fixed sample budget, and at $N = 10$ trajectories per epoch that variance shows: SNIS wins all seven objectives, several by a wide margin (zale_mpo: 0.420 vs. 0.250), Table 11 for full results.

Acquisition function. Table 12 shows AFGG's versatility: it can be applied with any acquisition function. As expected, EI yields higher performance than PI or UCB.

Candidate pool size. Moving from 1 to 10 trajectories captures essentially all the achievable gain (mean $0.470 \rightarrow 0.480$); doubling again to 20 does not clearly help (mean 0.477, within a single row's typical standard deviation of the 10-trajectory result), while doubling the per-epoch generation cost. We fix 10 as the point where that curve has flattened, Table 13 for full results.

Where the temperature acts. Power tilt reshapes the SNIS weights, $h_n \propto \ell_n^{1/\gamma}$ (see Eq. (46)) — always a convex combination of the sampled embeddings, so the guided target cannot leave their span. Scale drift instead rescales the drift after the tilted mean is formed (see Eq. (45)): at $\lambda > 1$ this extrapolates past $\hat{z}^{\text{tilt}}$, pushing the implied target outside the convex hull of what was actually sampled and decoded — a region the frozen denoiser never saw in training. On this seven-objective slice the two mechanisms land close together (means 0.474–0.486, all within a row's typical standard deviation), so we are not claiming a large empirical gap here; the convex-hull argument, not this table, is why we default to power tilt at higher temperatures; Table 14 for full results.

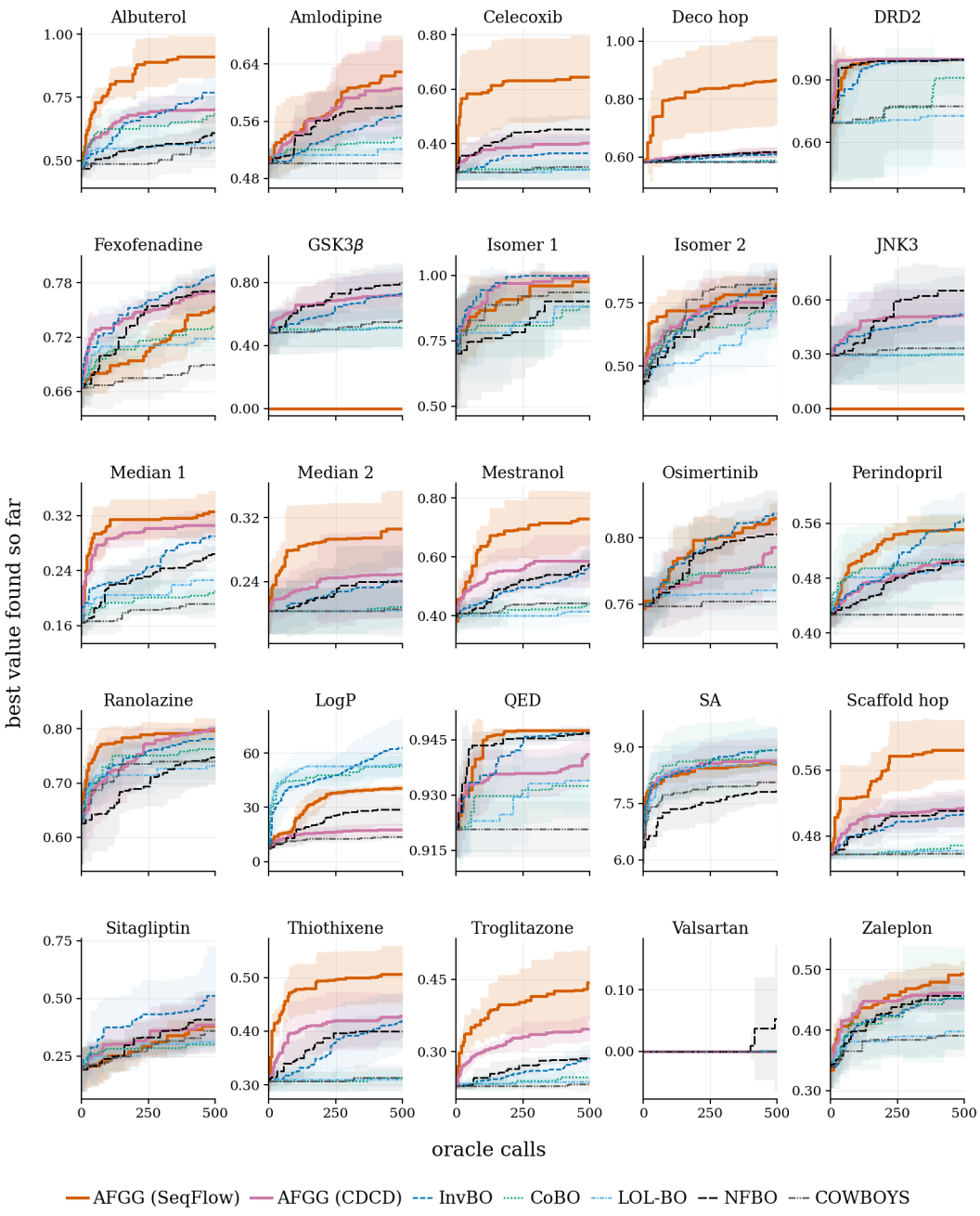

Figure 5: **Batch regime, coupled generation**: 500 acquisitions (100 initial points, then 100 epochs of 5), all 25 objectives.

	baselines					AFGG	
Objective	CoBO	InvBO	LOL-BO	NFBO	COWBOYS	CDCD	SeqFlow
albu_sim	0.692 ± 0.095	0.771 ± 0.057	0.579 ± 0.043	0.609 ± 0.073	0.551 ± 0.081	0.703 ± 0.057	**0.910 ± 0.083**
amlo_mpo	0.540 ± 0.021	0.569 ± 0.029	0.522 ± 0.026	0.582 ± 0.024	0.501 ± 0.021	**0.606 ± 0.068**	**0.630 ± 0.051**
cele_red	0.307 ± 0.035	0.366 ± 0.064	0.304 ± 0.036	0.452 ± 0.052	0.316 ± 0.052	0.402 ± 0.034	**0.645 ± 0.153**
deco_hop	0.588 ± 0.005	0.610 ± 0.007	0.586 ± 0.004	0.617 ± 0.018	0.583 ± 0.004	0.615 ± 0.018	**0.866 ± 0.153**
drd2	0.930 ± 0.090	0.996 ± 0.006	0.728 ± 0.247	0.994 ± 0.012	0.774 ± 0.210	**1.000 ± 0.000**	**1.000 ± 0.000**
fexo_mpo	0.732 ± 0.025	**0.789 ± 0.013**	0.718 ± 0.010	**0.771 ± 0.025**	0.689 ± 0.017	0.770 ± 0.017	0.753 ± 0.029
gsk3b	0.516 ± 0.125	0.730 ± 0.174	0.512 ± 0.138	0.802 ± 0.120	0.558 ± 0.151	0.718 ± 0.144	**0.950 ± 0.044**
iso_c7	0.880 ± 0.074	**1.000 ± 0.000**	0.880 ± 0.074	0.901 ± 0.107	0.935 ± 0.081	0.995 ± 0.011	0.995 ± 0.011
iso_c9	0.718 ± 0.110	0.808 ± 0.100	0.758 ± 0.074	0.778 ± 0.104	**0.843 ± 0.051**	0.777 ± 0.127	0.825 ± 0.052
jnk3	0.300 ± 0.161	0.524 ± 0.211	0.298 ± 0.161	0.652 ± 0.128	0.336 ± 0.248	0.514 ± 0.140	**0.704 ± 0.110**
med_1	0.212 ± 0.026	0.291 ± 0.023	0.228 ± 0.022	0.265 ± 0.012	0.192 ± 0.020	**0.306 ± 0.022**	**0.326 ± 0.031**
med_2	0.211 ± 0.024	0.241 ± 0.037	0.204 ± 0.028	0.241 ± 0.045	0.205 ± 0.034	**0.249 ± 0.054**	**0.306 ± 0.048**
mest_sim	0.439 ± 0.020	0.565 ± 0.068	0.428 ± 0.038	0.577 ± 0.052	0.443 ± 0.028	**0.587 ± 0.089**	**0.730 ± 0.093**
osim_mpo	0.786 ± 0.019	**0.815 ± 0.009**	0.768 ± 0.015	0.803 ± 0.027	0.762 ± 0.017	0.794 ± 0.021	0.812 ± 0.009
peri_mpo	0.509 ± 0.069	**0.567 ± 0.041**	0.499 ± 0.044	0.504 ± 0.012	0.427 ± 0.019	0.506 ± 0.029	0.551 ± 0.021
rano_mpo	0.763 ± 0.031	0.784 ± 0.025	0.732 ± 0.028	0.748 ± 0.051	0.740 ± 0.033	**0.800 ± 0.007**	**0.797 ± 0.023**
logp	**52.790 ± 5.679**	**63.296 ± 16.073**	**53.962 ± 8.208**	29.014 ± 8.537	13.554 ± 1.710	17.799 ± 3.424	40.594 ± 2.014
qed	0.934 ± 0.008	0.947 ± 0.002	0.934 ± 0.008	0.947 ± 0.001	0.921 ± 0.008	0.941 ± 0.007	**0.948 ± 0.000**
sa	**8.952 ± 0.668**	**8.947 ± 0.581**	8.590 ± 0.473	7.841 ± 0.310	8.066 ± 0.225	8.645 ± 0.597	8.570 ± 0.163
scaf_hop	0.479 ± 0.027	0.506 ± 0.021	0.463 ± 0.010	0.511 ± 0.020	0.459 ± 0.005	**0.514 ± 0.023**	**0.584 ± 0.036**
sita_mpo	0.310 ± 0.036	**0.512 ± 0.216**	0.319 ± 0.032	**0.410 ± 0.121**	0.360 ± 0.047	0.387 ± 0.126	0.381 ± 0.025
thio_red	0.327 ± 0.035	0.420 ± 0.053	0.311 ± 0.022	0.400 ± 0.030	0.313 ± 0.024	**0.429 ± 0.040**	**0.506 ± 0.053**
trog_red	0.250 ± 0.009	0.287 ± 0.028	0.237 ± 0.008	0.286 ± 0.009	0.232 ± 0.014	**0.348 ± 0.026**	**0.444 ± 0.076**
vals_sma	0.000 ± 0.000	**0.001 ± 0.002**	0.000 ± 0.000	**0.053 ± 0.118**	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
zale_mpo	0.454 ± 0.083	0.455 ± 0.022	0.401 ± 0.027	0.457 ± 0.027	0.391 ± 0.036	**0.462 ± 0.027**	**0.493 ± 0.021**

Table 8: **Batch regime, coupled generation.** Final best score on all 25 PMO objectives after 500 acquisitions (100 initial points, then 100 epochs of 5), mean ± sample standard deviation over 5 seeds. AFGG generates each batch *coupled* here: one joint qLogEI tilt per candidate batch, so its members are generated aware of each other. Every cell is judged against the opposing group: an AFGG cell against the five baselines, a baseline cell against the two AFGG columns. **Bold**: the cell's mean exceeds every mean in the opposing group. Green : that lead is significant, exceeding the cell's own standard deviation; bold without green therefore means ahead but within noise.

Metric	CDCD	SeqFlow
Validity ↑	**1.0000**	0.9972
Uniqueness ↑	**0.9932**	0.9862
Novelty ↑	**0.9956**	0.9838
FCD ↓	20.99	**6.02**

Table 9: Unconditional sample quality. 5000 molecules drawn from each generative model and scored against a 30k-molecule subset of GuacaMol.

	Guidance on vs off	
Objective	on	off (λ=0)
osim_mpo	0.714 ± 0.064	**0.722 ± 0.048**
med_2	**0.199 ± 0.013**	0.173 ± 0.012
amlo_mpo	**0.486 ± 0.053**	0.408 ± 0.018
zale_mpo	0.420 ± 0.054	**0.431 ± 0.030**
rano_mpo	**0.788 ± 0.012**	0.745 ± 0.026
peri_mpo	**0.443 ± 0.018**	0.405 ± 0.073
trog_red	**0.308 ± 0.020**	0.238 ± 0.016
MEAN	0.480	0.446
Wins	5	2

Table 10: Guidance on vs. off; *off* sets $\lambda = 0$, flattening the SNIS weights to uniform so the tilted mean equals the untilted one.

	Guidance estimator	
Objective	SNIS (closed form)	REINFORCE
osim_mpo	**0.714 ± 0.064**	0.671 ± 0.033
med_2	**0.199 ± 0.013**	0.170 ± 0.010
amlo_mpo	**0.486 ± 0.053**	0.410 ± 0.062
zale_mpo	**0.420 ± 0.054**	0.250 ± 0.085
rano_mpo	**0.788 ± 0.012**	0.687 ± 0.036
peri_mpo	**0.443 ± 0.018**	0.382 ± 0.055
trog_red	**0.308 ± 0.020**	0.294 ± 0.013
MEAN	0.480	0.409
Wins	7	0

Table 11: Guidance estimator: closed-form SNIS vs. REINFORCE (score-function estimator with a leave-one-out baseline) in place of the mean-swap identity.

	Acquisition function		
Objective	EI	PI	UCB
osim_mpo	**0.714 ± 0.064**	0.708 ± 0.065	0.696 ± 0.041
med_2	**0.199 ± 0.013**	0.183 ± 0.015	0.170 ± 0.009
amlo_mpo	0.486 ± 0.053	**0.495 ± 0.029**	0.490 ± 0.013
zale_mpo	0.420 ± 0.054	0.439 ± 0.045	**0.445 ± 0.055**
rano_mpo	**0.788 ± 0.012**	0.753 ± 0.026	0.746 ± 0.037
peri_mpo	0.443 ± 0.018	**0.469 ± 0.053**	0.391 ± 0.044
trog_red	**0.308 ± 0.020**	0.300 ± 0.016	0.235 ± 0.015
MEAN	0.480	0.478	0.453
Wins	4	2	1

Table 12: Acquisition function driving both the guidance tilt and candidate selection; UCB uses $\beta = 0.1$.

	Candidate pool size			
Objective	1	5	10	20
osim_mpo	**0.736 ± 0.010**	0.712 ± 0.053	0.714 ± 0.064	0.711 ± 0.063
med_2	0.186 ± 0.012	0.192 ± 0.008	0.199 ± 0.013	**0.204 ± 0.007**
amlo_mpo	0.486 ± 0.045	**0.504 ± 0.038**	0.486 ± 0.053	0.496 ± 0.029
zale_mpo	**0.432 ± 0.048**	0.411 ± 0.084	0.420 ± 0.054	0.406 ± 0.060
rano_mpo	0.764 ± 0.008	0.768 ± 0.019	**0.788 ± 0.012**	0.771 ± 0.015
peri_mpo	0.407 ± 0.045	0.448 ± 0.047	0.443 ± 0.018	**0.453 ± 0.031**
trog_red	0.274 ± 0.018	0.288 ± 0.015	**0.308 ± 0.020**	0.294 ± 0.011
MEAN	0.470	0.475	0.480	0.477
Wins	2	1	2	2

Table 13: Candidate pool size: guided trajectories rolled out per epoch, of which the best by acquisition value is evaluated.

	Where the temperature acts			
Objective	power tilt γ=2	scale drift λ=0.5	scale drift λ=2	scale drift λ=5
osim_mpo	**0.714 ± 0.064**	0.702 ± 0.034	0.694 ± 0.040	0.682 ± 0.026
med_2	**0.199 ± 0.013**	0.194 ± 0.010	0.191 ± 0.013	0.182 ± 0.007
amlo_mpo	0.486 ± 0.053	0.474 ± 0.029	0.423 ± 0.029	**0.496 ± 0.051**
zale_mpo	0.420 ± 0.054	**0.449 ± 0.026**	0.402 ± 0.041	0.401 ± 0.029
rano_mpo	**0.788 ± 0.012**	0.750 ± 0.031	0.782 ± 0.023	0.773 ± 0.023
peri_mpo	0.443 ± 0.018	**0.458 ± 0.043**	0.455 ± 0.056	0.407 ± 0.029
trog_red	**0.308 ± 0.020**	0.278 ± 0.014	0.300 ± 0.022	0.302 ± 0.027
MEAN	0.480	0.472	0.464	0.463
Wins	4	2	0	1

Table 14: Different temperature parameters and how they affect the final generation process.