GPA: Generative Population Annealing for Test-Time Sequence Design with Pretrained Generative Models

Anirban Sarkar [1] **Alejandra Duran** [1] **Peter Koo** [1]

Abstract

Oracle-guided biological sequence design must improve predicted function without moving outside the sequence distribution where the oracle is trustworthy. We introduce Generative Population Annealing (GPA), a test-time sampler for sequence design with a frozen generator and frozen oracle. GPA instantiates annealed Sequential Monte Carlo at population scale: particles are initialized from a pretrained sequence prior, reweighted by an oracle reward tilt, selectively upsampled as effective sample size falls, mutated with the pretrained generator, and returned as the design pool. The base sampler targets a reward-tilted prior; practical variants shape the proposal or modify the objective to trade activity, specificity, fidelity, and diversity. Across enhancer and promoter benchmarks, GPA scales to thousands of sequences and is competitive with inference-time samplers, gradient-based editors, tree-search diffusion, and RL-fine-tuned generators. The same inference loop is evaluated with masked discrete-diffusion and autoregressive DNA generators. GPA reaches high predicted activity while preserving strong motif fidelity and model-based likelihood, although specialized baselines retain higher diversity or stronger k-mer fidelity in some settings. Cross-oracle and sequence-level audits suggest fewer obvious reward-hacking pathologies, including shorter homopolymer runs than CTRL-DNA in the HepG2 audit (5.1 bp mean maximum run versus 20.9 bp), but all validation remains computational.

Accepted at the 2026 Workshop on Generative and Agentic AI for Biology (ICML 2026). [1]Cold Spring Harbor Laboratory, Quantitative Biology, NY, USA. Correspondence to: Anirban Sarkar <asarkar@cshl.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

1. Introduction

Designing biological sequences with specified functions is a central problem in synthetic biology, spanning regulatory DNA, proteins, and other molecular substrates. In this paper, we focus on cell-type-specific regulatory DNA, including enhancers and promoters, where the goal is to design sequences that drive activity in a chosen cellular context while remaining biologically plausible. This setting is a useful testbed for oracle-guided sequence design because the sequence-to-function map is high-dimensional, combinatorial, and only partially observed experimentally. Regulatory activity depends on transcription-factor motifs, local sequence context, and chromatin state, so computational methods typically rely on an oracle: a sequence-to-function model trained on experimental measurements and used to score, guide, or reward candidate sequences during search (De Winter et al., 2025). Existing approaches include *in silico* evolution (Vaishnav et al., 2022; Taskiran et al., 2024), gradient-based editing (Linder & Seelig, 2021; Schreiber et al., 2025), genetic algorithms (Li et al., 2025), conditional generation (Avdeyev et al., 2023; Stark et al., 2024; Sarkar et al., 2024; DaSilva et al., 2026; Awasthi et al., 2026), RL fine-tuning of generative priors (Wang et al., 2024; Chen et al., 2025b), and reward-tilted sampling (Lee et al., 2025; Pani et al., 2025; Skreta et al., 2025). Across these methods, the central challenge is that aggressive optimization can move designs away from the oracle's training distribution, where predicted activity becomes unreliable. A useful design sampler must therefore balance oracle-directed improvement against a prior over natural sequences, while returning a sufficiently diverse population of candidates for downstream experimental screening.

Oracle optimization and reward hacking. A direct approach is to start from a sequence, evaluate the oracle, and iteratively edit the sequence toward higher predicted activity. This is the regime in which reward hacking is most apparent. Because the oracle is a learned model trained on a finite experimental dataset, sequences that maximize its prediction can exploit model-specific artifacts instead of regulatory logic that transfers beyond the training oracle. In regulatory DNA, this can appear as excessive motif tiling, low-complexity sequence patterns, or other designs

that score highly under the optimization oracle but fail under held-out models or biological plausibility checks. This failure mode is not specific to regulatory genomics. It reflects a broader problem in reward-driven generation, where optimizing a learned proxy can move samples outside the region in which the proxy is reliable (Skalse et al., 2022; Clark et al., 2023; Gao et al., 2023). A pretrained generative model fills this gap by assigning high probability to sequences resembling natural regulatory DNA. The design objective is therefore to find sequences that are high-scoring under the oracle while remaining likely under a generative prior.

Fine-tuning generative priors. One way to combine these objectives is to fine-tune a pretrained generator with reinforcement learning, shifting its sample distribution toward high oracle reward while regularizing against the original prior (Ouyang et al., 2022). Recent biological sequence design methods instantiate this strategy with different backbones and optimization schemes. DRAKES (Wang et al., 2024) backpropagates a Gumbel-softmax-relaxed reward through the denoising trajectory of a pretrained masked-diffusion model, while CTRL-DNA (Chen et al., 2025b) applies constrained proximal policy optimization (PPO) to a HyenaDNA backbone with explicit specificity constraints. These methods can produce high-activity designs, but each new oracle, cell type, or constraint requires another optimization run. The generator is also updated directly against the oracle reward. When that update is strong, the resulting distribution can lose diversity or drift toward degenerate sequence patterns, weakening the prior constraint that motivated the pretrained generator in the first place.

Frozen-prior test-time sampling. A second approach leaves the pretrained generator fixed and steers sampling only at inference time. Sequential Monte Carlo (SMC) provides a natural framework for this setting: a population of candidate sequences is reweighted by oracle reward, selectively upsampled to retain high-reward candidates, and mutated using the pretrained generator's transition kernel. This approximates a reward-tilted target distribution that combines the prior and reward without modifying the generator. Existing twisted-SMC and reward-guided diffusion samplers have this structure (Lee et al., 2025; Pani et al., 2025), but have largely been evaluated with small particle populations or outside population-scale biological sequence design. DNA-CRAFT (Awasthi et al., 2026) extends this direction for enhancer design using conditional Monte Carlo tree diffusion and achieves strong sequence-fidelity metrics, but its search procedure can trade target-cell activity against motif and k-mer fidelity. These results suggest that the main practical issue is how to explore the prior-reward distribution at sufficient scale while retaining a diverse pool of high-scoring candidates.

We introduce GPA, an annealed SMC sampler for biological sequence design that keeps both the pretrained generator and the oracle fixed. The base sampler targets a reward-tilted prior, but the main practical point is the population: GPA returns thousands of particles as the design pool, rather than using particles only to estimate an expectation. Starting from prior samples, GPA reweights sequences by an incremental reward tilt, selectively upsamples when effective sample size falls, and mutates sequences with the pretrained generator. The resulting pool can be selected post hoc for activity, specificity, diversity, or motif-grounded criteria. Our contributions are:

- We instantiate ESS-adaptive annealed SMC as a population-output, frozen-generator test-time sampler for oracle-guided biological sequence design.
- We scale the sampler to thousands of particles and return the final population directly as the design pool.
- We define a generator-interface view of GPA and evaluate the same inference loop with masked discrete-diffusion and autoregressive DNA backbones.
- We evaluate GPA across enhancer and promoter benchmarks against inference-time samplers, gradient-based editors, tree-search diffusion, and RL-fine-tuned generators.
- We test reward overoptimization using held-out cross-oracle evaluation and sequence-level naturalness diagnostics, showing that GPA avoids some oracle-specific and low-complexity pathologies observed in greedy or fine-tuned alternatives.

This interface is deliberately small: GPA needs a way to sample initial sequences, a generator-defined mutation proposal, and an evaluable scalar reward. This is what lets the same inference loop be reused across generator classes and design objectives without retraining the generator.

2. Background

Pretrained generative models over biological sequences. Let Σ denote a discrete vocabulary, such as $\{A, C, G, T\}$ for DNA or the amino-acid alphabet for proteins, and let $\mathcal{X} = \Sigma^L$ be the space of length-L sequences. A pretrained generative model defines a prior distribution p_θ over $\mathcal{X}$, assigning higher probability to sequences that resemble its training corpus. In biological sequence design, this prior is useful because useful designs occupy a small and structured subset of an exponentially large discrete space, while purely reward-driven search can move rapidly into regions with little support under the biological training distribution.

Different model classes provide different ways to sample or locally perturb sequences. Discrete diffusion models define a corruption process and learn the reverse denoising distribution, enabling partial corruption followed by resampling of masked positions (Austin et al., 2021; Lou et al., 2023; Sahoo et al., 2024). Autoregressive models factorize $p_\theta(x) = \prod_t p_\theta(x_t \mid x_{<t})$ and can resample tokens under a partial context (Nguyen et al., 2023; Brixi et al., 2026). Continuous diffusion models operate through relaxed sequence representations and require an additional discretization or decoding step before returning discrete sequences. In this work, we treat the generator through the operation required by GPA: a model-defined mutation kernel that proposes sequence variants while keeping search anchored to the pretrained prior. The theoretical analysis in Sec. 3.4 states the standard SMC assumptions explicitly; the implementation uses generator-defined proposals to restore diversity while keeping mutations tied to the pretrained prior.

Reward-tilted sequence design. A design task specifies a desired property through an oracle reward $r : \mathcal{X} \to \mathbb{R}$, typically a learned sequence-to-function model trained on experimental measurements (De Winter et al., 2025). In our experiments, this reward represents enhancer or promoter activity, with multi-cell-type settings handled by combining target-cell activity with penalties for off-target activity (Chen et al., 2025b; Awasthi et al., 2026; Wang et al., 2024). The same formulation can also accommodate other learned or computed sequence-level objectives, such as binding affinity or structural compatibility, provided they can be evaluated as rewards over candidate sequences (Goel et al., 2024; Geffner et al., 2025; Tang et al., 2025). Sampling from p_θ alone preserves the pretrained sequence distribution but does not target the desired function. Maximizing r alone can exploit the oracle outside the region where its predictions are reliable (Skalse et al., 2022; Gao et al., 2023; Clark et al., 2023). A standard way to combine the pretrained prior and oracle reward is to sample from the reward-tilted distribution (Neal, 2001; Del Moral et al., 2006; Wu et al., 2023; Pani et al., 2025)

$$\pi_\beta(x) \propto p_\theta(x) \exp(\beta r(x)), \tag{1}$$

where $\beta \geq 0$ controls the strength of selection toward high-reward sequences.

This target has a useful variational interpretation: π_β optimizes expected reward minus $\beta^{-1}\mathrm{KL}(\pi \| p_\theta)$, up to the normalizing constant in Eq. 1 (Boyd & Vandenberghe, 2004; Yang et al., 2025). Thus, β sets the trade-off between oracle reward and deviation from the pretrained prior. This is the same basic objective used in reward-regularized generator fine-tuning (Ouyang et al., 2022), but here it is used as a sampling target with the generator fixed. Direct sampling from Eq. 1 is intractable for nontrivial sequence lengths, motivating annealed SMC.

Sequential Monte Carlo and population annealing. Sequential Monte Carlo (SMC) approximates a difficult target distribution through a sequence of easier intermediate targets (Del Moral et al., 2006; Naesseth et al., 2019). For the reward-tilted target in Eq. 1, this corresponds to an annealing path

$$\pi_{\beta_0}, \pi_{\beta_1}, \ldots, \pi_{\beta_T}, \qquad 0 = \beta_0 < \beta_1 < \cdots < \beta_T = \beta_*.$$

The sampler maintains a population of N weighted sequences $\{(x_t^i, w_t^i)\}_{i=1}^N$. In standard SMC, this population is often used to approximate expectations under the target distribution (Doucet & Johansen, 2009; Chopin, 2002). In sequence design, the sampled population itself is the output: a pool of candidate sequences for downstream selection or experimental screening.

At each step, weights are updated by the incremental reward tilt,

$$w_{t+1}^i \propto w_t^i \exp\{(\beta_{t+1} - \beta_t) r(x_t^i)\}.$$

The population is selectively upsampled when the effective sample size,

$$\mathrm{ESS}_t = \frac{(\sum_i w_t^i)^2}{\sum_i (w_t^i)^2},$$

falls below a chosen threshold αN for $\alpha \in (0, 1)$ (Liu & Chen, 1995). Mutation steps are then used to restore diversity after selection. Under standard annealed-SMC assumptions, including controlled incremental weights and appropriate mutation kernels, the weighted empirical distribution converges to the target at rate $\mathcal{O}(1/\sqrt{N})$ (Del Moral et al., 2006).

The annealing schedule can also be chosen adaptively. Rather than fixing all temperatures in advance, one can choose the next β_{t+1} by bisection so that the effective sample size remains above a prescribed fraction of the population (Del Moral et al., 2012). This is useful because the reward distribution changes during sampling: early populations may tolerate large temperature steps, while later populations may require smaller steps to avoid collapse. The same structure appears in statistical physics as population annealing (Hukushima & Iba, 2003; Machta, 2010; Wang et al., 2015). Appendices D and F compare related method families.

Closest SMC-style sequence-design methods. Closest prior methods use related machinery for different purposes. $\mathrm{SMC}_{\mathrm{amot}}$ and debiasing-guidance methods use SMC mainly as an inference correction for guided diffusion, typically with small particle counts rather than as a population-scale candidate generator. SVDD uses per-step proposal branching but not ESS-adaptive annealing over a large returned population. FK-style correctors provide useful reward-tilting language, but have not, to our knowledge, been evaluated as frozen-prior, population-output samplers for

regulatory DNA across both discrete-diffusion and autoregressive generators. GPA combines ESS-adaptive annealing, generator-defined mutation proposals, and thousands-particle design pools, with the returned population evaluated directly as the design object.

3. The GPA Algorithm

3.1. Core algorithm

GPA is an annealed SMC sampler for the reward-tilted target in Eq. 1. Starting from prior samples, it increases the reward tilt through an adaptive temperature schedule, selectively upsamples high-weight particles, and mutates sequences with the pretrained generator. For diffusion backbones, mutation is partial corruption followed by re-denoising with p_θ (Uehara et al., 2025); for autoregressive backbones, it is partial-context resampling. These generator-defined proposals restore diversity while keeping mutations tied to the pretrained prior. Figure 1 summarizes the loop; Appendix B gives pseudocode and Appendix E gives a worked example.

Population as the design output. Prior SMC-based diffusion samplers have typically used smaller populations, treating particles as inference samples rather than as a design pool (Wu et al., 2023; Pani et al., 2025). In GPA, the returned population is the output. Scaling N to thousands of sequences changes the role of SMC from estimating a low-dimensional quantity to producing candidates for post hoc selection by activity, specificity, diversity, or motif-grounded criteria.

3.2. Practical extensions

Branch factor K. A practical way to increase local selection pressure is to draw K proposals for each sequence and retain the one with the highest reward. This changes the proposal distribution rather than leaving the base target in Eq. 1 unchanged. Under a symmetric proposal, this is the standard maximum order-statistic proposal, which adds a monotone reward-dependent shaping term (Appendix C.3). Empirically, increasing K has a similar qualitative effect to increasing selection pressure, at extra oracle-evaluation cost.

Gumbel-softmax oracle gradient. For differentiable oracles, we optionally bias proposals using gradients of r with respect to a relaxed sequence representation, implemented with a straight-through Gumbel-softmax estimator and strength η (Maddison et al., 2017; Jang et al., 2017). This is a proposal heuristic, not part of the base SMC target, and is unavailable for strictly black-box or API-only oracles.

Diversity regularization. To reduce population collapse, we optionally add a population-conformity penalty $\lambda c(x; S_t)$, where c is the fraction of positions matching the current population majority token. This changes the target to

$$\pi_{\beta,\lambda}(x) \propto p_\theta(x) \exp\{\beta r(x) - \lambda c(x; S_t)\}, \qquad (2)$$

and we report when $\lambda > 0$.

The distinction between the base sampler and the experimental recipes is important. Base GPA is the clean annealed-SMC reference for Eq. 1. The variants used in the benchmark recipes should be read more narrowly: K-branch selection, DPS, and GC-centering shape the proposal; λ and pw modify the objective; and DPS/GC-centering require oracle gradients. Thus, the empirical claim is not that every recipe is an exact sampler for Eq. 1, but that these proposal and objective controls give a practical frozen-prior design procedure.

3.3. Multi-objective design via pw

For multi-cell-type design, we combine target-cell activity with an off-target penalty using the linear fitness

$$r_{\mathrm{pw}}(x) = r_{\mathrm{target}}(x) - \mathrm{pw} \cdot \frac{1}{|\mathcal{O}|} \sum_{c \in \mathcal{O}} r_c(x), \qquad (3)$$

where $\mathcal{O}$ is the set of off-target cell types and $\mathrm{pw} \geq 0$ is the penalty weight. Sweeping pw explores the activity–specificity trade-off without re-training the generator or oracle. This is a linear scalarization of the multi-objective trade-off, with pw playing the role of a Lagrange multiplier; Appendix C.4 gives the corresponding formal statement.

3.4. Standard SMC facts used by GPA

The base sampler uses standard annealed-SMC and adaptive-SMC constructions; we do not claim a new SMC convergence theorem. Under the usual assumptions on incremental weights, resampling, and mutation kernels, the weighted empirical distribution converges to the terminal reward-tilted target at rate $\mathcal{O}(N^{-1/2})$ (Del Moral et al., 2006; Naesseth et al., 2019). ESS-adaptive temperature selection is likewise standard and consistent in the large-N limit (Del Moral et al., 2012). These statements apply to the base sampler; the experimental controls in Sec. 3.2 modify the proposal or objective as described above and in Appendix C.

4. Results

We evaluate GPA across four comparisons: the DNA-CRAFT enhancer benchmark (§4.1), a cross-oracle reward-inflation test (§4.2), the Gosai HepG2 sampler benchmark (§4.3), and a CTRL-DNA promoter-design comparison using HyenaDNA (§4.4). Together, these comparisons ask

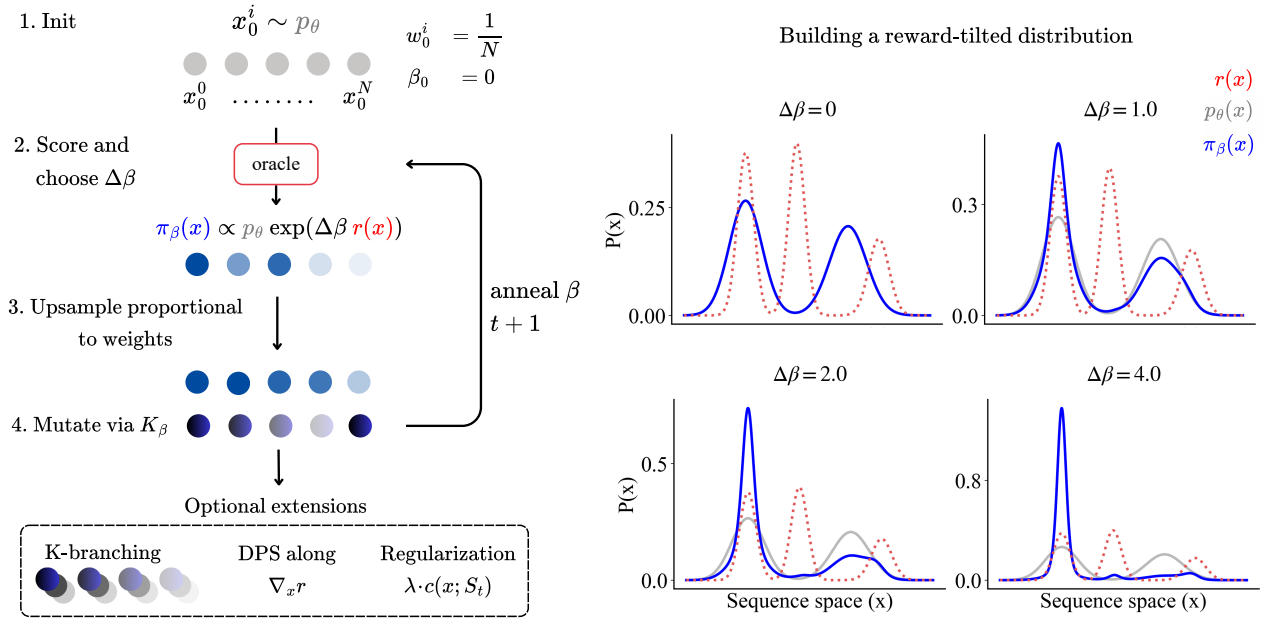


Figure 1. GPA overview. **Left:** Starting from prior samples, GPA adapts β by ESS, reweights, upsamples, and mutates with the pretrained generator; the full final population is returned as the design pool. Optional controls shape the proposal or modify the target. **Right:** Increasing β shifts mass toward higher-reward regions while remaining modulated by the pretrained prior.

whether population-scale frozen-prior sampling can improve predicted function while preserving prior- or benchmark-based evidence of sequence fidelity and reducing oracle-specific optimization artifacts. Ablations for population size, branch factor, DPS, GC-centering, runtime, and naturalness are in the appendix. Across benchmarks, GPA runs in minutes to tens of minutes per seed on a single H100 without per-task policy optimization; on the K562 enhancer runtime comparison, it is $3.4\times$ faster than LEDIDI and $9.3\times$ faster than ISM for a matched 5,000-sequence pool.

4.1. Cell-type-specific enhancer design on the 3-cell benchmark

We first evaluate GPA on the DNA-CRAFT enhancer-design benchmark. The task is to design 200 bp regulatory sequences with high predicted activity in a target cell type while preserving motif and k-mer statistics of high-activity natural enhancers. The benchmark uses the Gosai MPRA dataset (Gosai et al., 2024), which contains approximately 798,000 enhancer sequences measured in HepG2, K562, and SK-N-SH cells. We follow the DNA-CRAFT protocol (Awasthi et al., 2026): a 50/50 graph-cluster split is used to train separate design-time and evaluation Enformer oracles, and outputs are evaluated by MinGap to a real top-99.9% anchor, motif correlation, 3-mer correlation, and pairwise Hamming diversity. GPA runs at inference time on the publicly available MDLM backbone of Sahoo et al. (2024); DNA-CRAFT reports its strongest results with DiMamba, so the backbone is not matched.

We interpret MinGap jointly with motif and 3-mer correlation, since high activity is only meaningful here if the selected pool remains close to the benchmark's high-activity natural enhancers.

For the main comparison, we use GPA with $K = 8$ and DPS, without MCTS-style planning, and select 64 sequences from each 5,000-sequence pool using DNA-CRAFT's greedy pool-composite rule. GPA improves MinGap over the reported DNA-CRAFT values in all three cell types and matches or exceeds motif correlation, while DNA-CRAFT retains a consistent 3-mer advantage. This is the relevant operating point for this benchmark: higher target-cell activity than the strongest sequence-fidelity baseline, while retaining motif and 3-mer statistics close to high-activity natural enhancers. Because DNA-CRAFT reports its strongest values with DiMamba whereas GPA uses publicly available MDLM, Table 1 should be read as benchmark competitiveness rather than an isolated comparison of search procedures. The relevant claim is therefore conditional: under the published benchmark and matched top-64 selection rule, GPA occupies a favorable activity–fidelity point, not that its search procedure dominates DNA-CRAFT independent of backbone.

GPA does not reach the diversity ceiling of this benchmark: several baselines report Diversity $= 1.98$, while GPA reaches 1.83–1.86. Appendix N audits the HepG2 top-128 pools and finds long homopolymer tracts in CTRL-DNA designs (20.9 bp mean maximum run) but not in GPA (5.1

Table 1. **DNA-CRAFT enhancer benchmark.** Top-64 sequences were selected by `by_pool_composite`, matched to DNA-CRAFT's G^* at $N_{\max} = 64$. GPA values are means across 3 runs using a 5,000-particle pool. Baselines are from Awasthi et al. (2026); standard deviations are in Appendix Table 18.

Cell	Metric	SMC	CG	TDS	DRAKES	D3	Ledidi	Ctrl-DNA	DNA-CRAFT	**GPA**
HepG2	MinGap ↑	1.61	−0.23	0.40	−1.40	0.05	5.77	**7.79**	4.35	6.91
	Motif ↑	0.55	0.86	0.40	0.06	0.87	0.58	0.63	**0.92**	**0.92**
	3-mer ↑	0.81	0.97	0.74	−0.36	**0.98**	0.76	0.49	**0.98**	0.95
	Diversity ↑	0.83	**1.98**	0.96	1.86	**1.98**	**1.98**	1.90	**1.98**	1.86
K562	MinGap ↑	4.12	0.00	1.62	−0.20	0.18	7.66	**9.07**	5.69	8.13
	Motif ↑	0.45	0.85	0.51	0.14	0.86	0.65	0.63	0.93	**0.94**
	3-mer ↑	0.66	0.94	0.65	−0.35	0.96	0.69	0.41	**0.98**	0.95
	Diversity ↑	0.31	**1.98**	0.64	1.96	**1.98**	**1.98**	1.90	**1.98**	1.83
SK-N-SH	MinGap ↑	0.56	−0.28	0.19	0.09	−0.01	3.03	3.72	3.23	**4.98**
	Motif ↑	0.52	0.86	0.48	0.23	0.84	0.38	0.48	0.88	**0.92**
	3-mer ↑	0.78	0.95	0.72	−0.38	0.93	0.37	0.20	**0.97**	0.94
	Diversity ↑	1.27	**1.98**	0.92	1.83	1.97	**1.98**	1.86	**1.98**	1.86

bp, close to the real Gosai value of 5.4 bp).

Population size is the main budget parameter. With the recipe and top-64 rule fixed, increasing N from 128 to 20,000 improves MinGap by +1.03 in HepG2, +1.08 in K562, and +2.98 in SK-N-SH, with gains in motif correlation, 3-mer correlation, and diversity (Appendix J). We use $N = 5{,}000$ as a runtime-benefit compromise; moving to 20,000 improves every metric but gives modest average gains for a 4× larger pool.

4.2. Cross-oracle evaluation of reward inflation

We next test whether population-scale sampling is less vulnerable to oracle-specific optimization than greedy single-sequence editing. On the LentiMPRA K562 task, all methods optimize LegNet K562 (LN), while outputs are evaluated by a held-out AlphaGenome K562 oracle (AG) that is never used during optimization (Appendix K). Because AG is also a learned predictor, this is a cross-model transfer test rather than experimental validation.

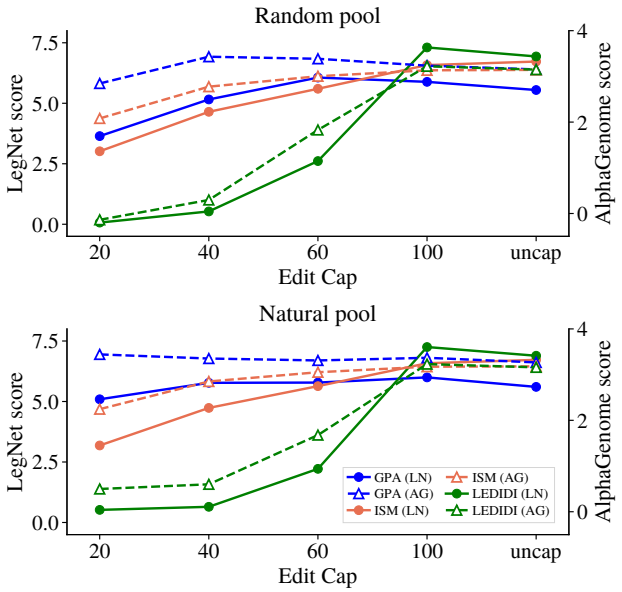
We compare GPA against ISM (Vaishnav et al., 2022) and LEDIDI (Schreiber et al., 2025) on random and natural 5,000-sequence seed pools under edit-budget caps of 20, 40, 60, and 100, plus each method's best final checkpoint. Figure 2 reports LN/AG ratio and a joint-rank score that rewards high AG with low LN/AG inflation; we use joint rank as the primary summary because the ratio is unstable when AG is near zero.

On the random seed pool, GPA has the best joint-rank score at every edit budget. On the natural seed pool, LEDIDI has LN/AG ratios closer to 1 at small edit budgets, reflecting limited movement from the natural seeds rather than high held-out activity. The joint-rank score is similar at 20 and 40 edits and favors GPA at caps of 60 and above. At larger edit budgets, LEDIDI continues to increase LN while its joint-rank score decreases, consistent with optimization-oracle inflation that does not transfer to AG. This is the failure mode motivating the sampler: additional optimization pressure can improve the design oracle while reducing the quality of the transferred design pool under an independently trained evaluator. Full wall-time results are reported in Appendix L.

4.3. Activity–naturalness trade-off on Gosai HepG2 enhancers

We next compare GPA to reward-guided and SMC-based samplers on Gosai HepG2 using the protocol of Pani et al. (2025). The benchmark reports held-out gReLU activity, ATAC accessibility, k-mer and motif correlations to high-expression references, and approximate log-likelihood under the pretrained model. GPA uses the same MDLM backbone as above, the DRAKES split-oracle as the design-time activity head, three random seeds, top-640-by-composite selection, and the same recipe as Table 1 with no off-target penalty ($\mathrm{pw} = 0$), since this is a single-cell (HepG2) task. The GC-centering term is also omitted for this comparison, letting the frozen prior set base composition rather than an external target. Without it, GPA's GC (0.41) already matches the natural Gosai reference (0.44, Appendix N), and the term's effect is ablated in Appendix H. We report $\beta_* = 5$, 10, and 15 as inference-time operating points of the same frozen generator.

This benchmark should be read as a trade-off rather than a single-metric ranking. Sweeping β_* from 5 to 15 moves GPA along the activity-fidelity frontier with a single knob: at $\beta_* = 5$ it matches the strongest fidelity baselines (highest 3-mer 0.91; JASPAR 0.81) while already exceeding all but SGDD and SVDD (N=16) on Pred-Activity; at $\beta_* = 10$ it reaches Pred-Activity 8.04 and 93.5% ATAC. At $\beta_* = 15$ it also attains the highest App-Log-Lik in the



	LN/AG ratio ↓			Joint rank ↑		
edits	GPA	ISM	LEDIDI	GPA	ISM	LEDIDI
RANDOM POOL						
20	**1.281**	1.450	—	**1.042**	0.956	1.002
40	**1.504**	1.675	1.793	**1.089**	0.907	1.004
60	1.791	1.864	**1.429**	**1.055**	0.928	1.017
100	**1.819**	2.100	2.269	**1.281**	0.968	0.751
uncap	**1.760**	2.140	2.203	**1.271**	0.902	0.827
NATURAL POOL						
20	1.479	1.419	**1.047**	0.999	0.997	**1.005**
40	1.722	1.661	**1.081**	0.992	**1.006**	1.001
60	1.747	1.847	**1.322**	**1.058**	0.933	1.009
100	**1.781**	2.074	2.245	**1.271**	0.957	0.772
uncap	**1.715**	2.113	2.182	**1.318**	0.881	0.801

Figure 2. **Joint LegNet/AlphaGenome behavior across editing budgets.** All methods optimize LegNet K562 (LN) and are evaluated by a held-out AlphaGenome-derived K562 oracle (AG). Curves show per-pool behavior across edit budgets; tables report LN/AG ratio and joint rank. LN/AG measures optimization-oracle inflation when AG is positive; higher joint rank indicates designs that transfer better to AG without excessive LN/AG inflation. Best per row is bolded; the ratio is omitted when mean AG is negative.

Table 2. **Gosai HepG2 sequence-design benchmark.** The table reports trade-offs between predicted activity, accessibility, marginal-statistic fidelity, and model likelihood. Pred-Activity and App-Log-Lik are medians; ATAC-Acc, 3-mer Corr, and JASPAR Corr are means across 3 seeds. Baselines are from Pani et al. (2025), originally from DRAKES (Wang et al., 2024). Standard deviations are in Appendix Table 19.

Method	Pred-Activity ↑	ATAC-Acc ↑ (%)	3-mer Corr ↑	JASPAR Corr ↑	App-Log-Lik ↑
Pretrained	0.17	1.5	−0.061	0.249	−261
CG	3.30	0.0	−0.065	0.212	−266
CFG	5.04	92.1	0.746	0.864	−265
DRAKES w/o KL	6.44	82.5	0.307	0.557	−281
DRAKES	5.61	92.5	0.887	**0.911**	−264
SGDD ($\beta = 30$)	8.85	90.9	0.470	0.466	−263
SGDD ($\beta = 50$)	**9.32**	96.4	0.370	0.398	−269
SVDD ($N = 8$)	6.57	67.4	0.813	0.753	−258
SVDD ($N = 16$)	6.89	84.3	0.891	0.834	−260
SMC_{amot} ($N = 1$)	5.40	82.1	0.653	0.778	−259
SMC_{amot} ($N = 8$)	6.35	95.8	0.736	0.845	−261
SMC_{amot} ($N = 16$)	6.68	**97.6**	0.796	0.886	−261
GPA ($\beta = 5$)	6.82	92.3	**0.912**	0.813	−274
GPA ($\beta = 10$)	8.04	93.5	0.794	0.711	−262
GPA ($\beta = 15$)	8.45	95.3	0.563	0.551	**−257**

Table 3. **Cell-type-specific promoter design on the Reddy MPRA benchmark.** Both methods select the top 128 sequences by predicted target activity. CTRL-DNA uses the R200 PPO policy; GPA uses the universal inference-time recipe. Values are mean(std) across 5 seeds.

Cell	Method	Seeds	Target	Composite	Shannon	Motif Corr
JURKAT	CTRL-DNA R200	5	5.68 (0.48)	4.04 (0.44)	**1.64 (0.07)**	0.78 (0.06)
	GPA universal	5	**8.37 (0.09)**	**6.32 (0.10)**	1.53 (0.10)	**0.91 (0.01)**
K562	CTRL-DNA R200	5	6.02 (0.06)	**5.01 (0.08)**	**1.69 (0.01)**	0.80 (0.04)
	GPA universal	5	**7.00 (0.01)**	4.76 (0.03)	1.63 (0.05)	**0.84 (0.03)**
THP1	CTRL-DNA R200	5	4.16 (0.26)	2.15 (0.18)	**1.79 (0.03)**	0.70 (0.09)
	GPA universal	5	**4.52 (0.04)**	**2.44 (0.03)**	1.60 (0.16)	**0.82 (0.04)**

table (-257). SGDD reaches the highest Pred-Activity, and SMC_{amot} reaches the highest ATAC accessibility.

The trade-off is explicit within one method: pushing β_* toward 15 raises activity and accessibility but lowers marginal-statistic fidelity (3-mer 0.56, JASPAR 0.55), whereas $\beta_* = 5$ recovers fidelity comparable to the strongest baselines at lower activity. GPA thus does not dominate at any single β_*, but its one inference-time knob spans operating points that separate baselines reach only via distinct training or value functions. Because GPA reports top-selected sequences while the Pretrained row reports unfiltered samples, App-Log-Lik should be interpreted as a relative diagnostic of whether design methods remain in high-likelihood regions of the MDLM prior, not as evidence that selected designs are more natural than unfiltered prior samples.

4.4. Cell-type-specific promoter design on the 3-cell benchmark

Finally, we compare GPA to RL fine-tuning on the Reddy MPRA promoter benchmark, with 250 bp promoters measured in JURKAT, K562, and THP1 cells. CTRL-DNA fine-tunes a separate constrained PPO policy for each cell type on a HyenaDNA backbone. GPA uses the same pretrained HyenaDNA backbone with one universal inference-time recipe across all three cells ($\text{pw} = 0.35$, $\lambda = 1.0$; Appendix G). Following CTRL-DNA, candidate pools are ranked by predicted target activity, the top 128 sequences are selected, and performance is reported using target activity, composite activity-specificity score, Shannon diversity, and motif correlation.

GPA achieves higher target activity and motif correlation than CTRL-DNA in all three cell types, and higher composite score in JURKAT and THP1. This comparison is useful because CTRL-DNA trains a separate PPO policy for each target cell type, whereas GPA uses one frozen HyenaDNA prior and one inference-time recipe across all three cells. The result therefore supports the backbone-interface claim: the same population-annealing loop can be reused with an autoregressive generator, rather than being tied to masked discrete diffusion. The remaining gaps are also clear. CTRL-DNA achieves higher Shannon diversity in all three cells and higher K562 composite score, indicating better off-target suppression at that operating point. On K562, GPA reaches higher target activity with very low seed-to-seed variance, so the composite-score gap is mainly a specificity/diversity issue rather than failure to activate the target cell. We use the same pw across all three cells to test a universal inference-time recipe rather than per-cell hyperparameter tuning.

5. Conclusion

We presented GPA, a test-time annealed SMC sampler for oracle-guided biological sequence design with frozen generative priors. GPA uses ESS-adaptive reward annealing, selective upsampling, and generator-defined mutation proposals to return a population-scale design pool without fine-tuning the generator or oracle. Across enhancer and promoter benchmarks, GPA is competitive with inference-time samplers, editors, tree-search diffusion, and RL-fine-tuned generators. The main practical benefit is that a user can change the oracle, target cell type, or scalar reward terms without retraining the generator, while still obtaining a large candidate pool for downstream filtering or experimental screening. This shifts design from training a new generator for each objective to reusing a fixed prior as a reusable proposal mechanism.

The results should be interpreted within several limits. All validation is computational: the held-out AlphaGenome experiment tests cross-model transfer, not wet-lab activity. The DNA-CRAFT comparison is not backbone-matched, so it establishes benchmark competitiveness rather than isolated search-algorithm superiority. Practical variants are not exact samplers for the base reward-tilted target: K-branch selection, DPS, and GC-centering shape the proposal, while λ and pw modify the objective. The strongest evidence for this framing is not single-metric dominance, but the ability to return large design pools that remain competitive across activity, fidelity, transfer, and runtime diagnostics. GPA still does not uniformly dominate specialized baselines: diversity, k-mer fidelity, and K562 promoter specificity remain gaps, and future work should pair oracle-based evaluation with experimental validation.

References

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Avdeyev, P., Shi, C., Tan, Y., Dudnyk, K., and Zhou, J. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pp. 1276–1301. PMLR, 2023.

Awasthi, A., Bednarsky, R., Schaefer, M., and Bock, C. Conditional monte carlo tree diffusion for designing cell-type-specific and biologically faithful regulatory dna. *arXiv preprint arXiv:2604.20488*, 2026.

Blickle, T. and Thiele, L. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary computation*, 4(4):361–394, 1996.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004.

Brixi, G., Durrant, M. G., Ku, J., Naghipourfar, M., Poli, M., Sun, G., Brockman, G., Chang, D., Fanton, A., Gonzalez, G. A., et al. Genome modelling and design across all domains of life with evo 2. *Nature*, pp. 1–13, 2026.

Chen, T., Zhang, Y., Tang, S., and Chatterjee, P. Multi-objective-guided discrete flow matching for controllable biological sequence design. *arXiv preprint arXiv:2505.07086*, 2025a.

Chen, X., Ma, S., Lin, R., Lin, J., and Wang, B. Ctrl-dna: Controllable cell-type-specific regulatory dna design via constrained rl. *arXiv preprint arXiv:2505.20578*, 2025b.

Chopin, N. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

Clark, K., Vicol, P., Swersky, K., and Fleet, D. J. Directly fine-tuning diffusion models on differentiable rewards. *arXiv preprint arXiv:2309.17400*, 2023.

Dang, M., Han, J., Xu, M., Xu, K., Srivastava, A., and Ermon, S. Inference-time scaling of diffusion language models with particle gibbs sampling. *arXiv preprint arXiv:2507.08390*, 2025.

DaSilva, L. F., Senan, S., Kribelbauer-Swietek, J. F., Patel, Z. M., Louis, L. K., Reddy, A. J., Gabbita, S., Rosen, J. D., Nussbaum, Z., Córdova, C. M. V., et al. Designing synthetic regulatory elements using the generative ai framework dna-diffusion. *Nature Genetics*, 58(1):180–194, 2026.

Dau, H.-D. and Chopin, N. Waste-free sequential monte carlo. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):114–148, 2022.

De Winter, S., Konstantakos, V., and Aerts, S. Modelling and design of transcriptional enhancers. *Nature Reviews Bioengineering*, 3(5):374–389, 2025.

Del Bono, L. M., Ricci-Tersenghi, F., and Zamponi, F. Demonstrating real advantage of machine-learning-enhanced monte carlo for combinatorial optimization. *arXiv preprint arXiv:2510.19544*, 2025.

Del Moral, P., Doucet, A., and Jasra, A. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

Del Moral, P., Doucet, A., and Jasra, A. An adaptive sequential monte carlo method for approximate bayesian computation. *Statistics and computing*, 22(5):1009–1020, 2012.

Doucet, A. and Johansen, A. M. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 2009.

Gao, L., Schulman, J., and Hilton, J. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Geffner, T., Didi, K., Zhang, Z., Reidenbach, D., Cao, Z., Yim, J., Geiger, M., Dallago, C., Kucukbenli, E., Vahdat, A., et al. Proteina: Scaling flow-based protein structure generative models. *arXiv preprint arXiv:2503.00710*, 2025.

Goel, S., Schray, P. M., Zhang, Y., Vincoff, S., Kratochvil, H. T., and Chatterjee, P. Token-level guided discrete diffusion for membrane protein design. *arXiv preprint arXiv:2410.16735*, 2024.

Gosai, S. J., Castro, R. I., Fuentes, N., Butts, J. C., Mouri, K., Alasoadura, M., Kales, S., Nguyen, T. T. L., Noche, R. R., Rao, A. S., Joy, M. T., Pardis, C. S., Reilly, S. K., and Tewhey, R. Machine-guided design of cell-type-targeting cis-regulatory elements. *Nature*, 634(8036):1211–1220, 2024.

Gruver, N., Stanton, S., Frey, N., Rudner, T. G., Hotzel, I., Lafrance-Vanasse, J., Rajpal, A., Cho, K., and Wilson, A. G. Protein design with guided discrete diffusion. *Advances in neural information processing systems*, 36: 12489–12517, 2023.

Hartman, E., Wallin, J., Malmström, J., and Olsson, J. Controllable protein design through feynman-kac steering. *arXiv preprint arXiv:2511.09216*, 2025.

Hasan, M., Ohanesian, V., Gazizov, A., Bengio, Y., Aspuru-Guzik, A., Bondesan, R., Skreta, M., and Neklyudov, K. Discrete feynman-kac correctors. *arXiv preprint arXiv:2601.10403*, 2026.

He, J., Jeha, P., Potaptchik, P., Zhang, L., Hernández-Lobato, J. M., Du, Y., Syed, S., and Vargas, F. Crepe: Controlling diffusion with replica exchange. *arXiv preprint arXiv:2509.23265*, 2025.

Hukushima, K. and Iba, Y. Population annealing and its application to a spin glass. *AIP Conference Proceedings*, 690(1):200–206, 11 2003. ISSN 0094-243X. doi: 10.1063/1.1632130. URL https://doi.org/10.1063/1.1632130.

Jain, M., Bengio, E., Hernandez-Garcia, A., Rector-Brooks, J., Dossou, B. F. P., Ekbote, C. A., Fu, J., Zhang, T., Kilgour, M., Zhang, D., Simine, L., Das, P., and Bengio, Y. Biological sequence design with GFlowNets. In *International Conference on Machine Learning (ICML)*, 2022. arXiv:2203.04115.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax, 2017. URL https://arxiv.org/abs/1611.01144.

Jiang, K., Yan, Z., Di Bernardo, M., Sgrizzi, S. R., Villiger, L., Kayabolen, A., Kim, B., Carscadden, J. K., Hiraizumi, M., Nishimasu, H., et al. Rapid in silico directed evolution by a protein language model with evolvepro. *Science*, 387 (6732):eadr6006, 2024.

Kim, S., Kim, M., and Park, D. Test-time alignment of diffusion models without reward over-optimization. *arXiv preprint arXiv:2501.05803*, 2025.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M. (eds.), *Machine Learning: ECML 2006*, pp. 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Lee, C. K., Jeha, P., Frellsen, J., Lió, P., Albergo, M., and Vargas, F. Debiasing guidance for discrete diffusion with SMC. *arXiv preprint arXiv:2502.06079*, 2025. ICLR 2025.

Lew, A. K., Zhi-Xuan, T., Grand, G., and Mansinghka, V. K. Sequential Monte Carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023.

Li, J., Zhang, P., Xi, X., Liu, L., Wei, L., and Wang, X. Modeling and designing enhancers by introducing and harnessing transcription factor binding units. *Nature Communications*, 16(1):1469, 2025.

Li, X., Zhao, Y., Wang, C., Scalia, G., Eraslan, G., Nair, S., Biancalani, T., Regev, A., Levine, S., and Uehara, M. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding (SVDD). *arXiv preprint arXiv:2408.08252*, 2024.

Linder, J. and Seelig, G. Fast activation maximization for molecular sequence design. *BMC bioinformatics*, 22(1): 510, 2021.

Liu, J. S. and Chen, R. Blind deconvolution via sequential imputations. *Journal of the american statistical association*, 90(430):567–576, 1995.

Lou, A., Meng, C., and Ermon, S. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Loula, J., LeBrun, B., Du, L., Lipkin, B., Pasti, C., Grand, G., Liu, T., Emara, Y., Freedman, M., Eisner, J., et al. Syntactic and semantic control of large language models via sequential monte carlo. *arXiv preprint arXiv:2504.13139*, 2025.

Luo, Z., Jin, Z., Wang, L., Bing, L., and Schön, T. B. Self-rewarding sequential monte carlo for masked diffusion language models. *arXiv preprint arXiv:2602.01849*, 2026.

Machta, J. Population annealing with weighted averages: A monte carlo method for rough free-energy landscapes. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 82(2):026704, 2010.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables, 2017. URL https://arxiv.org/abs/1611.00712.

Naesseth, C. A., Lindsten, F., and Schön, T. B. Elements of sequential monte carlo. *Foundations and Trends® in Machine Learning*, 12(3):187–306, 2019.

Neal, R. M. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

Nguyen, E., Poli, M., Faizi, M., Thomas, A., Wornow, M., Birch-Sykes, C., Massaroli, S., Patel, A., Rabideau, C., Bengio, Y., et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36: 43177–43201, 2023.

Nisonoff, H., Xiong, J., Allenspach, S., and Listgarten, J. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv preprint arXiv:2406.01572*, 2024.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Pani, C., Ou, Z., and Li, Y. Test-time alignment of discrete diffusion models with sequential monte carlo. 2025.

Pitt, M. K. and Shephard, N. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

Ren, Y., Gao, W., Ying, L., Rotskoff, G. M., and Han, J. Driftlite: Lightweight drift control for inference-time scaling of diffusion models. *arXiv preprint arXiv:2509.21655*, 2025.

Sahoo, S. S., Arriola, M., Schiff, Y., Gokaslan, A., Marroquin, E., Chiu, J. T., Rush, A., and Kuleshov, V. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37: 130136–130184, 2024.

Sarkar, A., Duran, A., Yu, Y., Lin, D.-W., Kang, Y., Somia, N., Mantilla, P., Zhou, J., Nagai, M., Tang, Z., et al. Designing dna with tunable regulatory activity using discrete diffusion. *bioRxiv*, pp. 2024–05, 2024.

Schiff, Y., Sahoo, S. S., Phung, H., Wang, G., Boshar, S., Dalla-torre, H., de Almeida, B. P., Rush, A., Pierrot, T., and Kuleshov, V. Simple guidance mechanisms for discrete diffusion models. *arXiv preprint arXiv:2412.10193*, 2024.

Schreiber, J., Lorbeer, F. K., Heinzl, M., Reiter, F., Rafanel, B., Lu, Y. Y., Stark, A., and Noble, W. S. Programmatic design and editing of cis-regulatory elements. *bioRxiv*, pp. 2025–04, 2025.

Sinai, S., Wang, R., Whatley, A., Slocum, S., Locane, E., and Kelsic, E. D. Adalead: A simple and robust adaptive greedy search algorithm for sequence design, 2020. URL https://arxiv.org/abs/2010.02141.

Singhal, R., Horvitz, Z., Teehan, R., Ren, M., Yu, Z., McKeown, K., and Ranganath, R. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

Skalse, J., Howe, N., Krasheninnikov, D., and Krueger, D. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

Skreta, M., Akhound-Sadegh, T., Ohanesian, V., Bondesan, R., Aspuru-Guzik, A., Doucet, A., Brekelmans, R., Tong, A., and Neklyudov, K. Feynman-kac correctors in diffusion: Annealing, guidance, and product of experts. *arXiv preprint arXiv:2503.02819*, 2025.

Stanton, S., Maddox, W., Gruver, N., Maffettone, P., Delaney, E., Greenside, P., and Wilson, A. G. Accelerating bayesian optimization for biological sequence design with denoising autoencoders. In *International conference on machine learning*, pp. 20459–20478. PMLR, 2022.

Stark, H., Jing, B., Wang, C., Corso, G., Berger, B., Barzilay, R., and Jaakkola, T. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024.

Syed, S., Bouchard-Côté, A., Chern, K., and Doucet, A. Optimised annealed sequential monte carlo samplers. *arXiv preprint arXiv:2408.12057*, 2024.

Tang, S., Zhang, Y., and Chatterjee, P. Peptune: De novo generation of therapeutic peptides with multi-objective-guided discrete diffusion. *ArXiv*, pp. arXiv–2412, 2025.

Taskiran, I. I., Spanier, K. I., Dickmänken, H., Kempynck, N., Pančíková, A., Ekşi, E. C., Hulselmans, G., Ismail, J. N., Theunis, K., Vandepoel, R., et al. Cell-type-directed design of synthetic enhancers. *Nature*, 626(7997):212–220, 2024.

Uehara, M., Su, X., Zhao, Y., Li, X., Regev, A., Ji, S., Levine, S., and Biancalani, T. Reward-guided iterative refinement in diffusion models at test-time with applications to protein and dna design. *arXiv preprint arXiv:2502.14944*, 2025.

Vaishnav, E. D., de Boer, C. G., Molinet, J., Yassour, M., Fan, L., Adiconis, X., Thompson, D. A., Levin, J. Z., Cubillos, F. A., and Regev, A. The evolution, evolvability and engineering of gene regulatory dna. *Nature*, 603 (7901):455–463, 2022.

Wang, C., Uehara, M., He, Y., Wang, A., Biancalani, T., Lal, A., Jaakkola, T., Levine, S., Wang, H., and Regev, A. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. *arXiv preprint arXiv:2410.13643*, 2024.

Wang, W., Machta, J., and Katzgraber, H. G. Comparing monte carlo methods for finding ground states of ising spin glasses: Population annealing, simulated annealing, and parallel tempering. *Physical Review E*, 92(1):013303, 2015.

Wu, L., Trippe, B., Naesseth, C., Blei, D., and Cunningham, J. P. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023.

Yang, Z., Su, B., Cao, C., and Wen, J.-R. Regulatory dna sequence design with reinforcement learning. *arXiv preprint arXiv:2503.07981*, 2025.

Yao, Y., Pan, Y., Li, J., Tsang, I., and Yao, X. Proud: Pareto-guided diffusion model for multi-objective generation. *Machine Learning*, 113(9):6511–6538, 2024.

Ye, H., Lin, H., Han, J., Xu, M., Liu, S., Liang, Y., Ma, J., Zou, J., and Ermon, S. Tfg: Unified training-free guidance for diffusion models. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22370–22417. Curran Associates, Inc., 2024. doi: 10.52202/079017-0704. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2818054fc6de6dacdda0f142a3475933-Paper-Conference.pdf.

Yoon, T., Min, Y., Yeo, K., and Sung, M. Psi-sampler: Initial particle sampling for smc-based inference-time reward alignment in score models. *arXiv preprint arXiv:2506.01320*, 2025.

A. GPA variant summary

The base sampler is the clean annealed-SMC reference. The variants used in experiments are practical controls; their interpretation is narrower because some modify the proposal and others modify the target or scalarized objective.

B. GPA algorithm pseudocode

Algorithm 1 gives the core ESS-adaptive annealed SMC loop used by GPA. The main text summarizes the loop in Fig. 1; the pseudocode is placed here to keep the main methods section focused on the modeling choices and empirical claims.

C. Standard SMC statements and proof sketches

This appendix collects the formal statements used in Sec. 3.4. These are standard facts from annealed SMC, adaptive SMC, order statistics, and convex duality. The paper does not claim a new SMC theorem. The contribution is the way these pieces are used for fixed-prior, oracle-guided biological sequence design, where the SMC population is itself the returned design pool. We separate the base annealed-SMC target from practical modifications, including K-branch proposal selection, gradient-biased proposals, and population-level penalties.

Throughout, p_θ denotes a pretrained generative model over sequences, $r : \mathcal{X} \to \mathbb{R}$ is an evaluable reward, and the reward-tilted target is

$$\pi_\beta(x) \propto p_\theta(x) \exp(\beta r(x)).$$

For diffusion backbones, the mutation proposal is implemented by partial-mask $\circ$ denoise; for autoregressive backbones, by partial-context resampling. The exact convergence statements below use standard annealed-SMC assumptions on the mutation kernels. The implemented proposals should be read as prior-anchored approximations whose empirical behavior is evaluated in the Results.

C.1. Convergence of the base annealed-SMC sampler

Known result being used. Annealed SMC approximates a difficult target distribution by moving through a sequence of intermediate targets and maintaining a weighted particle approximation at each stage. We use the following theorem as the idealized reference point for the base GPA sampler.

Theorem C.1 (Base annealed-SMC convergence; standard)**.** *Consider Algorithm 1 with $K = 1$, $\lambda = 0$, and no gradient-biased proposal. Given:*

(A1) The incremental importance weights have finite second moments along the annealing path.

(A2) The mutation kernels satisfy the standard annealed-SMC assumptions for the intermediate targets $\{\pi_{\beta_t}\}_{t=0}^{T}$.

(A3) The resampling rule is unbiased.

Then for any bounded test function φ,

$$\sum_{i=1}^{N} w_T^i \varphi(x_T^i) \xrightarrow[N\to\infty]{\text{prob}} \mathbb{E}_{\pi_{\beta_*}}[\varphi],$$

with Monte Carlo error of order $\mathcal{O}(N^{-1/2})$ under the usual regularity conditions (Del Moral et al., 2006; Naesseth et al., 2019).

Proof sketch. This is the standard convergence result for SMC samplers (Del Moral et al., 2006; Naesseth et al., 2019). $(A1)$ controls the variance of the incremental importance weights along the annealing path. By $(A2)$ and $(A3)$, the weighted particle system gives a consistent empirical approximation to each intermediate target, and the final weighted empirical average converges to the corresponding expectation under π_{β_*} at the standard Monte Carlo rate. □

GPA-specific use. The GPA sampler proposes a generator-defined mutation kernel: partial-mask$\circ$denoise for diffusion backbones or partial-context resampling for autoregressive backbones. We therefore use Theorem C.1 as an idealized SMC reference point. The implemented proposal is not claimed to be an exact π_β-invariant transition in all settings; its role is to restore diversity while keeping mutations tied to the pretrained prior.

What this gives us in practice. The $\mathcal{O}(1/\sqrt{N})$ rate formalizes the population-as-budget intuition: under the idealized SMC assumptions, more particles reduce Monte Carlo error in empirical averages over the reward-tilted target. In sequence design, the same population is also the returned candidate pool, so larger N gives more candidates for post-hoc selection. The N-axis ablation in Appendix J tests how this scaling appears in the empirical design metrics.

C.2. ESS-adaptive schedule

Known result being used. Adaptive SMC can choose the next temperature using the post-reweighting effective sample size instead of fixing the full annealing path in advance (Del Moral et al., 2012). We use this standard construction to set the reward tilt in GPA.

The effective sample size after an incremental tilt Δ is

$$\mathrm{ESS}(\Delta) = \frac{\left(\sum_i w_t^i e^{\Delta r(x_t^i)}\right)^2}{\sum_i \left(w_t^i e^{\Delta r(x_t^i)}\right)^2}.$$

Table 4. What each GPA variant changes. The base sampler is the clean annealed-SMC reference. The variants used in experiments are practical controls, and the claim attached to each variant is correspondingly narrower.

Variant	Used	Proposal	Target	Grad.	Correct interpretation
Base GPA	reference	no	no	no	Annealed SMC for the reward-tilted prior under the assumptions of Sec. 3.4.
GPA-K	yes	yes	no	no	Reward-shaped proposal selection; higher local oracle pressure at extra evaluation cost.
GPA-DPS	yes	yes	no	yes	Gradient-biased proposal heuristic for differentiable oracles.
GPA-GC	yes	yes	no	yes	Proposal-level GC control used in the enhancer benchmark recipe.
GPA-λ	yes	possible	yes	no	Diversity-regularized modified target.
GPA-pw	yes	no	yes	no	Scalarized activity-specificity objective.

Algorithm 1 GPA core: ESS-adaptive annealed SMC

Input: pretrained generator p_θ; reward r; target temperature β_*; population size N; ESS threshold $\alpha \in (0,1)$; max steps T
Sample $\{x_0^i\}_{i=1}^N \overset{\text{iid}}{\sim} p_\theta$; set $w_0^i = 1/N$ and $\beta_0 = 0$
for $t = 0, \ldots, T-1$ **do**
  Choose $\Delta\beta_t$ by bisection so that the ESS after reweighting at $\beta_t + \Delta\beta_t$ is at least αN
  $\beta_{t+1} \leftarrow \min(\beta_t + \Delta\beta_t, \beta_*)$
  $\tilde{w}_{t+1}^i \leftarrow w_t^i \exp\{(\beta_{t+1} - \beta_t) r(x_t^i)\}$; normalize $\tilde{w}_{t+1}$
  if $\mathrm{ESS}(\tilde{w}_{t+1}) < \alpha N$ **then**
    Upsample ancestors proportional to $\tilde{w}_{t+1}^i$ to obtain $\{\bar{x}_t^i\}_{i=1}^N$; set $w_{t+1}^i = 1/N$
  else
    Set $\bar{x}_t^i = x_t^i$ and $w_{t+1}^i = \tilde{w}_{t+1}^i$
  end if
  Mutate: $x_{t+1}^i \sim M_\theta(\cdot \mid \bar{x}_t^i)$ {e.g., partial mask + denoise}
  if $\beta_{t+1} = \beta_*$ **then break**
  end if
end for
Return: $\{(x_T^i, w_T^i)\}_{i=1}^N$ =0

As Δ increases, the weights concentrate on higher-reward particles and the ESS decreases. The adaptive schedule chooses the largest step that keeps the post-tilt ESS above a prescribed threshold αN. This matters in sequence design because the reward distribution evolves during sampling: early populations admit larger temperature steps, while later populations require smaller steps once high-reward particles have concentrated. The adaptive schedule avoids choosing a global temperature path that is too conservative early in sampling and too aggressive later.

Lemma C.2 (ESS-adaptive schedule; standard adaptive SMC)**.** *Let $\Delta\beta_t$ be chosen by bisection so that the post-tilt ESS is at least αN, for $\alpha \in (0,1)$. Under the assumptions of Theorem C.1 and the regularity conditions of adaptive SMC, the resulting random temperature schedule is consistent in the large-N limit (Del Moral et al., 2012).*

Proof sketch. The ESS is monotone non-increasing in the temperature increment under the standard adaptive-SMC conditions. Bisection therefore identifies the largest admissible step up to numerical tolerance. As N grows, the empirical reward distribution converges to its population counterpart along the annealing path, so the adaptive schedule converges to the corresponding deterministic limiting schedule. The SMC consistency conclusion then follows from the adaptive-SMC results of Del Moral et al. (2012). □

GPA-specific use. We use $\alpha = 0.5$ throughout, a common default in SMC implementations (Liu & Chen, 1995). This setting allows the weights to concentrate between resampling events while limiting collapse to a small number of

dominant particles. Sensitivity to $\alpha \in [0.3, 0.8]$ is reported in Appendix G.

C.3. Branch factor K: order-statistic proposal shaping

Known result being used. Selecting the highest-reward sample among K iid proposals induces the standard maximum order-statistic distribution. In GPA, this gives a simple way to describe the effect of K-branching: it biases the mutation proposal toward higher-reward candidates.

Proposition C.3 (Order-statistic proposal)**.** *Let $q(x' \mid x)$ be a proposal kernel on $\mathcal{X}$ and let $r : \mathcal{X} \to \mathbb{R}$ be an evaluable reward. At state x, draw $Y_1, \ldots, Y_K \sim q(\cdot \mid x)$ iid and retain the proposal with the highest reward. Assuming no ties for simplicity, the induced proposal density is*

$$q_K(x' \mid x) = Kq(x' \mid x)\,[F_q(r(x') \mid x)]^{K-1}, \qquad (4)$$

where

$$F_q(\rho \mid x) = \Pr_{Y \sim q(\cdot \mid x)}[r(Y) \le \rho].$$

For $K = 1$, $q_K = q$. For $K > 1$, the proposal is biased toward states with higher reward under the local proposal distribution.

Proof. This is the standard density of the maximum order statistic applied to the scalar random variable $r(Y)$. A retained proposal at x' requires one proposal to land at x' and the remaining $K - 1$ proposals to have reward at most $r(x')$. Summing over which of the K proposals is retained gives the factor K, yielding Eq. 4. □

GPA-specific interpretation. In GPA, K is a proposal-selection parameter. Drawing multiple proposals and keeping the highest-reward one reduces wasted local proposals and increases local reward pressure. Operationally, K trades additional oracle evaluations for a more selective local mutation step. However, this changes the proposal distribution; it does not preserve the base target π_β without an additional correction. We therefore report recipes with their chosen K and treat $K > 1$ as a deliberate proposal modification. This is why the main text describes K as proposal shaping, not as a target-preserving reparameterization of β.

Empirical signature. The K-sweep in Appendix I tests whether increased proposal branching improves the matched-output-budget metrics. Because K changes the proposal distribution, its effect is empirical and benchmark-dependent.

C.4. pw as a Lagrangian-style scalarization

Known result being used. Linear penalties are the standard Lagrangian form for constrained optimization. We use this interpretation to connect the off-target penalty weight pw to a relaxed activity–specificity trade-off.

Proposition C.4 (Lagrangian form of the relaxed off-target constraint)**.** *Consider the relaxed problem over distributions on $\mathcal{X}$ that maximizes expected target reward subject to an upper bound on expected average off-target reward:*

$$\max_{\mu} \mathbb{E}_{x\sim\mu}[r_{\mathrm{target}}(x)] \quad \textit{subject to} \quad \mathbb{E}_{x\sim\mu}\left[\frac{1}{|\mathcal{O}|}\sum_{c\in\mathcal{O}} r_c(x)\right] \le \tau.$$

The Lagrangian contains the scalarized reward

$$r_{\mathrm{pw}}(x) = r_{\mathrm{target}}(x) - \mathrm{pw} \cdot \frac{1}{|\mathcal{O}|}\sum_{c\in\mathcal{O}} r_c(x),$$

where $\mathrm{pw} \ge 0$ is the multiplier on the off-target constraint. As τ varies over the feasible range, the corresponding multipliers $\mathrm{pw}(\tau)$ define a family of scalarized objectives along the activity–specificity trade-off, under the usual convex-duality assumptions for the relaxed problem (Boyd & Vandenberghe, 2004).

Proof sketch. The result follows by forming the Lagrangian for the relaxed constrained problem. The multiplier on the off-target constraint appears exactly as the coefficient on the average off-target reward. Standard convex-duality results apply to the relaxed problem under the usual feasibility and regularity assumptions (Boyd & Vandenberghe, 2004). □

GPA-specific use. In the experiments, we do not claim to solve the full constrained discrete optimization problem. We use pw as an inference-time scalarization parameter that can be swept without retraining the generator or oracle. This interpretation explains why pw is a useful knob: it indexes a family of scalarized objectives that trade target activity against average off-target activity. In practice, this lets us sweep specificity pressure at inference time while holding the generator, oracle, and sampling procedure fixed.

Empirical signature. The pw sweep tests whether the scalarized objective moves designs along the observed activity–specificity trade-off. Monotonicity is expected for the relaxed convex formulation, but the empirical sequence-design problem is discrete and model-dependent; we therefore report the observed trade-off directly.

C.5. Backbone and oracle interface

Known result being used. The standard SMC statement is agnostic to where the proposal kernel and reward function come from, once the assumptions of the theorem are satisfied. Thus, changing the generator or oracle does not require a new derivation if the new proposal satisfies the same mutation-kernel assumptions and the reward remains evaluable with stable incremental weights.

Remark C.5 (Backbone and oracle interface). The algorithmic interface consists of a generator-defined mutation proposal and an evaluable scalar reward. The formal convergence statement continues to apply under the assumptions of Theorem C.1. In practice, changing backbones changes the proposal distribution and can change mixing, runtime, and empirical design quality. Our experiments evaluate this interface across the specific backbones and oracles reported in the main text.

GPA-specific use. This interface is useful because GPA does not require retraining the generator when the reward changes. It also allows different generator classes to be used when they expose an appropriate mutation or resampling operation. This is an empirical portability claim, not a guarantee that every sequence generator will mix well under the same hyperparameters.

D. Method-family comparison

Table 5 gives a qualitative comparison of method families under the design requirements emphasized in this paper. These axes correspond to practical choices a user faces when applying a design method: whether the method returns a pool, whether it stays tied to a learned prior, whether it can use black-box rewards, whether selection pressure can be adjusted at inference time, whether constraints can be added at inference time, and whether the same procedure can be reused across generator classes. The axes were selected to surface trade-offs we observed in practice; they are not intended as a model-independent specification of what every sequence-design sampler must support. Because these are method-family summaries, the entries should be read as typical behavior rather than definitive statements about every possible implementation. The distinguishing feature of GPA is the combination of these properties in a single frozen-generator inference loop, rather than any one axis in isolation.

We classify methods along six axes: *population output* (does the method naturally return a large pool of designs from one run?), *prior-anchored proposals* (are mutations or samples tied to a pretrained sequence generator?), *black-box reward* (can the method use rewards without oracle gradients?), *test-time trade-off control* (can selection pressure or scalar objective terms be changed at inference time without retraining?), *composable constraints* (can additional scalar reward terms be added without retraining?), and *backbone interface* (can the method be reused across generator classes, such as discrete diffusion or autoregressive models, without re-deriving the algorithm?).

Interface combinations. A practical advantage of GPA is that the mutation proposal, reward oracle, and scalar reward terms are specified separately. This modularity is a design property of the interface; it does not imply that all compatible combinations have been benchmarked or that they will behave similarly. Changing one component does not require retraining the generator, but empirical performance still depends on the generator proposal, the oracle, and the reward terms used in a given benchmark. Table 6 separates combinations evaluated in this paper from combinations that are compatible with the same interface but not tested here.

Table 5. Qualitative method-family comparison. ✓ = typical support, ✗ = typically absent, ∼ = partial or implementation-dependent. Entries are not impossibility claims. For GPA, black-box compatibility refers to the base sampler; optional gradient-biased proposals require oracle gradients. GFlowNets are listed separately because they train an amortized reward-proportional sampler rather than fine-tuning a generator by RL.

	Pop. out.	Prior anch.	BB compat.	TT ctrl.	Compos.	Backbone
RL fine-tuning (DRAKES, CTRL-DNA)	∼	∼	✓	✗	∼	✗
GFlowNets (Jain et al., 2022)	✓	∼	✓	✗	∼	∼
Beam/tree search (EVO 2, Proteina (Geffner et al., 2025), DNA-CRAFT)	∼	✓	✓	✗	∼	∼
MCMC/score-guidance (DDRM, Langevin)	∼	∼	∼	✗	∼	∼
Genetic alg. (CMA-ES, neuroevolution)	✓	✗	✓	✗	✓	✓
GPA (ours)	✓	✓	∼	✓	✓	∼

Table 6. Examples of mutation proposals, reward oracles, and scalar reward terms under the GPA interface. ✓ = evaluated in this paper; ∼ = compatible with the interface but not evaluated here. Compatibility requires a usable mutation or resampling operation from the generator and an evaluable scalar reward.

Component	**Example**	**Use in this paper**	**Status**
Mutation proposal	MDLM partial-mask ∘denoise	Enhancer benchmarks	✓
Mutation proposal	HyenaDNA partial-context resampling	Promoter benchmark	✓
Mutation proposal	DiMamba-style discrete diffusion	Not run; checkpoints unavailable	∼
Mutation proposal	SEDD / D3PM-style discrete diffusion	Not evaluated here	∼
Mutation proposal	Causal LM resampling, e.g. EVO 2	Not evaluated here	∼
Reward oracle	LegNet / split-oracle activity model	LentiMPRA and design-time scoring	✓
Reward oracle	Enformer evaluation oracle	DNA-CRAFT benchmark protocol	✓
Reward oracle	gReLU MSE oracle	Gosai HepG2 / promoter comparisons	✓
Reward oracle	AlphaGenome evaluation oracle	Held-out cross-oracle audit	✓
Reward oracle	Other evaluable black-box scorer	Not evaluated here	∼
Scalar reward term	Off-target penalty pw	Multi-cell-type specificity	✓
Scalar reward term	Population conformity penalty λ	Diversity regularization	✓
Scalar reward term	GC penalty / GC filter	DNA-CRAFT and naturalness ablations	✓
Scalar reward term	Edit-budget constraint	Cross-oracle edit-budget analysis	✓

E. Worked example: SMC walk-through with $N = 8$

To make Algorithm 1 concrete, we walk through one GPA update on a toy population of $N = 8$ particles. The example is illustrative: the scores and sequences are chosen for clarity, and the purpose is to show how reweighting, ESS adaptation, resampling, and mutation interact in a single SMC step.

E.1. Score and reweight

Let the eight particles $\{x^1, \ldots, x^8\}$ have oracle scores r_i:

i	1	2	3	4	5	6	7	8
r_i	-0.45	-0.38	-0.52	-0.31	-0.48	-0.25	-0.41	-0.35

Suppose higher scores are preferred (true in our case for activity prediction) and take $\Delta\beta = 15$. The unnormalized weights are $\tilde{w}_i = \exp(\Delta\beta r_i)$ and the normalized weights that introduce a tilt to the distribution (Fig 3) are: $w_i = \tilde{w}_i / \sum_j \tilde{w}_j$:

i	1	2	3	4	5	6	7	8
w_i	0.025	0.073	0.009	0.207	0.016	**0.510**	0.046	0.114
w_i^2	0.0006	0.0053	0.0001	0.0428	0.0003	**0.2601**	0.0021	0.0130

E.2. Adapting $\Delta\beta$ by bisection on ESS

We now have:

$$\mathrm{ESS} = \frac{1}{\sum_i w_i^2} = \frac{1}{0.3241} \approx 3.08, \qquad \frac{\mathrm{ESS}}{N} = 0.39. \tag{5}$$

The ESS-adaptive schedule chooses the largest $\Delta\beta$ such that ESS $\geq \alpha N$, up to numerical tolerance and the remaining distance to β_*. With $\alpha = 0.5$, the target threshold is $\mathrm{ESS}_{\mathrm{target}} = 4$. Algorithm 2 shows the bisection used in this toy example.

The bisection gives $\Delta\beta \approx 11.7$, with ESS $\approx 4.0 = \alpha N$. Note that our original choice of $\Delta\beta = 15$ is more aggressive than what the adaptive schedule would choose.

E.3. Selective upsampling: multinomial vs. systematic

Multinomial. Multinomial upsampling draws copy counts $(c_1, \ldots, c_N) \sim \mathrm{Multinomial}(N, w)$. The expected copy count is $\mathbb{E}[c_i] = Nw_i$, but the realized counts can have substantial sampling variance.

Systematic. Systematic upsampling lays the weights end-to-end on the unit interval $[0, 1]$, where segment i has width w_i. It then places N evenly spaced pointers at positions $u + k/N$ for $k = 0, \ldots, N-1$, with $u \sim \mathrm{Uniform}[0, 1/N)$. Particle i receives one copy for each pointer that falls in its segment. This has the same expected copy counts as multinomial resampling but lower variance, and is the default in GPA.

For the offset used in Fig. 5, the eight pointers fall into the segments for particles $2, 4, 4, 6, 6, 6, 6, 8$, giving the copy counts below:

i	1	2	3	4	5	6	7	8
copies	0	1	0	2	0	4	0	1

Particles x^1, x^3, x^5, x^7 are removed, while x^6 is duplicated four times. After resampling, all retained particles have weight $1/N$.

E.4. Mutation: partial mask + denoise on a length-12 DNA toy

After selective usampling, each retained particle is independently mutated by the generator-defined proposal. For a diffusion backbone, this can be implemented by randomly masking a fraction nf of positions and denoising with p_θ. With $\mathrm{nf} = 0.25$ on a length-12 toy sequence, three positions are masked:

Step	Sequence
Before mutation	`ACGTATGCACGT`
Random mask 3 positions	`AC?TA?GC?CGT`
Denoise via p_θ	AC**T**TA**A**GC**A**CGT
Net edits	pos 3: G→T; pos 6: T→A; pos 9: C→A

This mutation step has two practical roles:

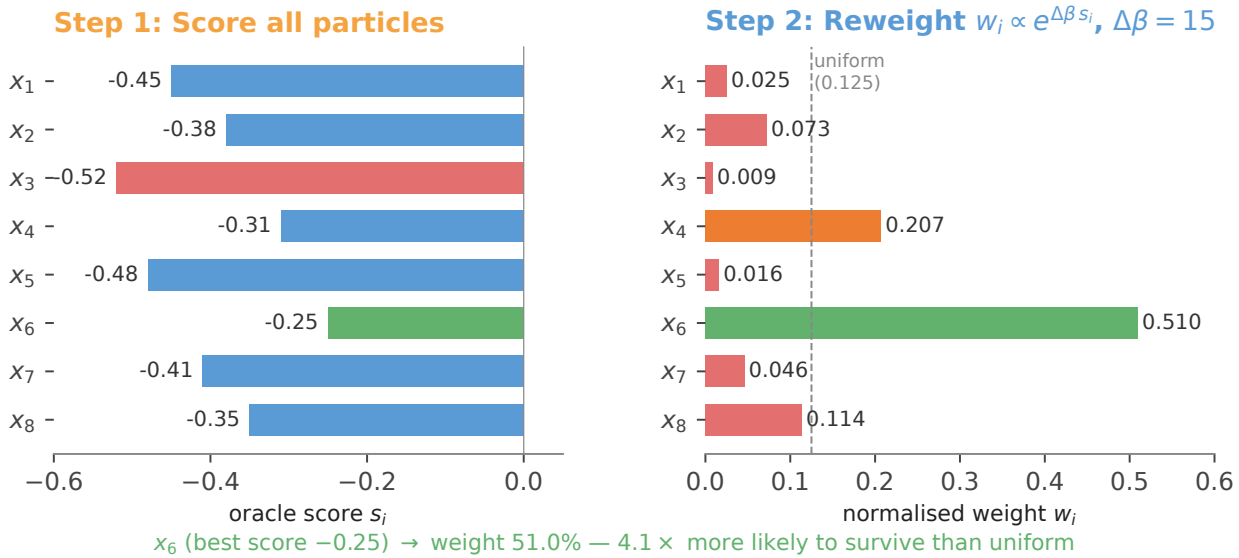


Figure 3. Score and reweight on the $N = 8$ toy example. **Left:** per-particle oracle scores r_i; x_6 has the highest score and x_3 the lowest. **Right:** normalized weights $w_i \propto \exp(\Delta\beta r_i)$ at $\Delta\beta = 15$; uniform weight $1/N = 0.125$ is shown as a dashed line.

Algorithm 2 Bisection for $\Delta\beta$ given an ESS threshold

0: **Input:** scores $\{r_i\}_{i=1}^N$, threshold αN, range $[\Delta\beta_{\min}, \Delta\beta_{\max}]$, max iterations T_{bs}
0: **for** $t = 1, \ldots, T_{\text{bs}}$ **do**
0: $\quad \Delta\beta_{\text{mid}} \leftarrow (\Delta\beta_{\min} + \Delta\beta_{\max})/2$
0: $\quad w_i \leftarrow \exp(\Delta\beta_{\text{mid}} r_i)/Z$, with $Z = \sum_j \exp(\Delta\beta_{\text{mid}} r_j)$
0: $\quad \text{ESS} \leftarrow 1/\sum_i w_i^2$
0: $\quad$ **if** $\text{ESS} > \alpha N$ **then**
0: $\quad\quad \Delta\beta_{\min} \leftarrow \Delta\beta_{\text{mid}}$ {larger tilt is still admissible}
0: $\quad$ **else**
0: $\quad\quad \Delta\beta_{\max} \leftarrow \Delta\beta_{\text{mid}}$ {tilt is too large}
0: $\quad$ **end if**
0: **end for**
0: **Return:** $\Delta\beta_{\min}$ =0

- *Prior-anchored mutation.* The denoiser conditions on the unmasked context and samples replacements from the pretrained model, instead of drawing independent per-position mutations.
- *Diversity after resampling.* In this toy example, duplicated particles receive independent mask locations and denoising samples, so resampled copies can separate again after mutation.

The mask fraction nf controls the size of the local move: smaller values give more local proposals, while larger values produce broader proposals. For example, on a 200 bp enhancer, $\text{nf} = 0.05$ masks about 10 positions per mutation step.

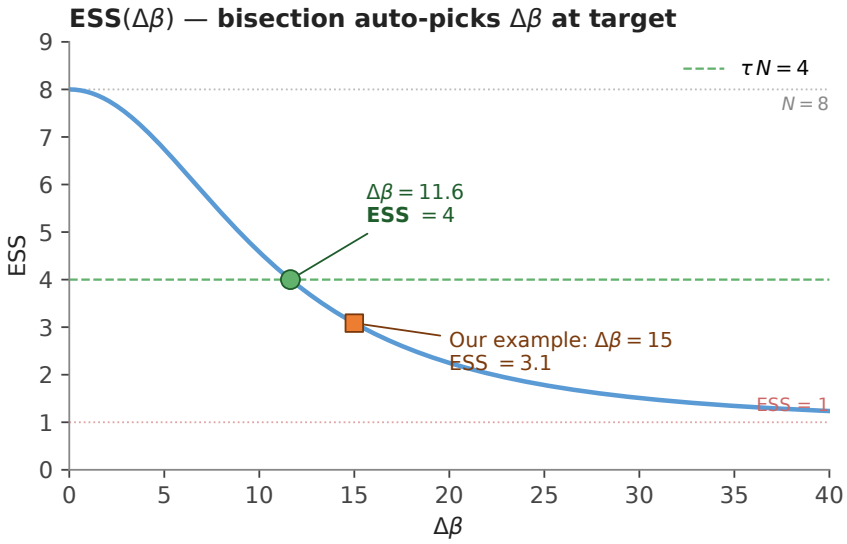


Figure 4. $\mathrm{ESS}(\Delta\beta)$ for the $N = 8$ toy oracle scores. The ESS decreases as the temperature increment increases because weight mass concentrates on higher-scoring particles. Bisection finds $\Delta\beta \approx 11.7$, where the ESS reaches the threshold $\alpha N = 4$. The illustrative value $\Delta\beta = 15$ drops ESS below the target threshold.

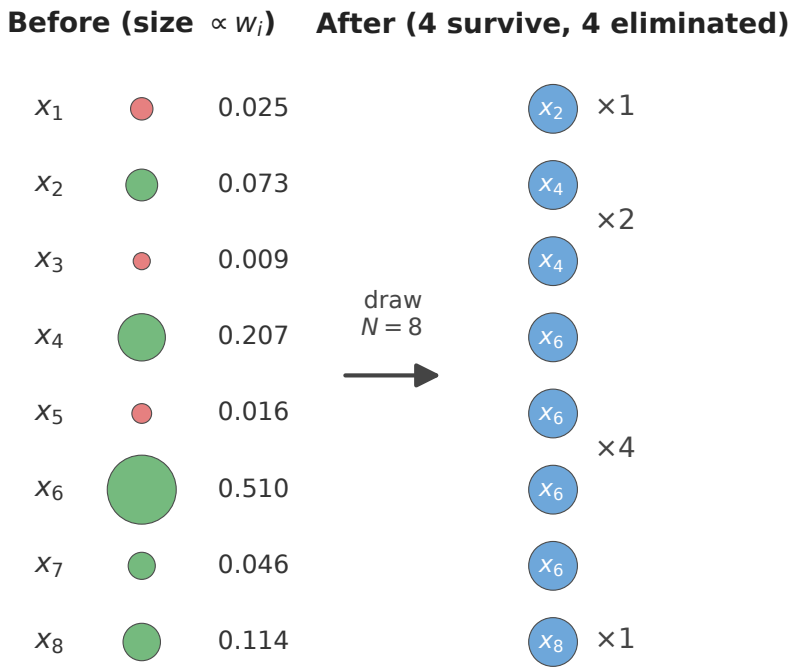


Figure 5. Systematic upsampling of the $N = 8$ toy example. **Left:** bubble area is proportional to w_i; green particles are retained and red particles are removed. **Right:** after selective upsampling, all retained particles have weight $1/N$; x_6 appears four times, x_4 twice, and x_2 and x_8 once each.

F. Related Work

SMC and Feynman–Kac steering for diffusion models. Sequential Monte Carlo and Feynman–Kac formulations have been used as inference-time mechanisms for steering pretrained diffusion models. Twisted-Diffusion-Sampler (Wu et al., 2023) introduced SMC for continuous diffusion at small particle counts ($N \leq 32$) using a first-order twist approximation of the reward. Lee et al. (2025) extended SMC ideas to discrete diffusion with a debiasing identity for partial-mask∘denoise kernels. Pani et al. (2025) proposed SMC_{amot}, a twisted SMC method with a first-order Taylor reward expansion, evaluated on synthetic 2-D Gaussians, binarized MNIST, and HepG2 enhancers at $N = 16$ with a fixed annealing schedule. Luo et al. (2026) use SMC with model confidence as the reward, which is orthogonal to our oracle-driven setting. Li et al. (2024) propose Soft Value-based Decoding (SVDD), a derivative-free per-step value-tilted resampler that we benchmark against in §4.3. Feynman–Kac correctors provide a broader theoretical lens: Skreta et al. (2025) formalize annealing, guidance, and product-of-experts inference as Feynman–Kac SMC for continuous diffusion; Singhal et al. (2025) cast inference-time scaling and steering of diffusion models in the same framework; and Hasan et al. (2026) extend the framework to discrete masked diffusion via Doob h-transform-based correctors. We view Feynman–Kac SMC as the relevant theoretical language for this class of inference-time steering methods. Concurrent FK-style steering for protein *structure* design (Hartman et al., 2025) is complementary to the sequence-design setting considered here.

The methodological contribution is the biological sequence-design instantiation: ESS-adaptive scheduling, population-scale outputs, frozen generators as mutation proposals, and evaluation on regulatory-sequence design benchmarks. We do not claim a new SMC convergence theorem. In the benchmarks here, GPA uses populations of 5,000 sequences, substantially larger than the small-particle SMC settings used in most prior diffusion-steering experiments. We treat the order-statistic branch factor K as proposal shaping; it does not exactly preserve the base target π_β.

MCTS and tree-search diffusion. DNA-CRAFT (Awasthi et al., 2026) uses Monte-Carlo Tree Search on a DiMamba diffusion backbone: each node selects via UCB, expands $M = 128$ children, rolls each leaf out to $t = 0$ by conditional ancestral sampling, and updates a MinGap-Set memory of the best 64 designs. Its benchmark protocol, including MinGap, motif correlation, 3-mer correlation, and diversity, provides the basis for Table 1. The comparison to GPA is not backbone-matched, since DNA-CRAFT reports its strongest results with DiMamba and those checkpoints were not available for our experiments. The MDLM ablation reported by DNA-CRAFT is therefore the closest available evidence for how much of their performance comes from the backbone itself. The algorithms also allocate computation differently: DNA-CRAFT uses tree search as the main design procedure, while GPA uses a population-level SMC sampler with generator-defined mutations. This leads to a different cost profile: DNA-CRAFT spends compute on tree expansion and leaf rollouts, whereas GPA spends compute on maintaining and mutating a large population before matched-budget selection. Appendix I evaluates an optional `ucb_stacked` variant that inserts UCB-style planning into the GPA mutation step. This variant tests whether tree-search planning improves the motif and k-mer metrics where base GPA is closest to DNA-CRAFT.

Reward fine-tuning of biological sequence generators. DRAKES (Wang et al., 2024) backpropagates Gumbel-softmax rewards through the denoising trajectory of a pre-trained MDLM. CTRL-DNA (Chen et al., 2025b) applies constrained PPO to a HyenaDNA backbone for cell-type-specific promoter design. These methods use the reward signal to update or align the generator for a given task. This can produce strong activity scores, but changing the oracle, cell type, or reward definition generally requires a new optimization run. This prior-anchored reward optimization view also has close analogues in RLHF-style KL-regularized learning (Ouyang et al., 2022). GPA instead keeps the generator fixed and uses the reward only at inference time through SMC reweighting and proposal modifications. The comparison to CTRL-DNA in §4.4 tests whether this frozen-generator strategy can approach or exceed several reward-fine-tuning metrics without policy optimization.

Test-time gradient, edit guidance, and directed evolution. ISM (Vaishnav et al., 2022; Taskiran et al., 2024) and LEDIDI (Schreiber et al., 2025) are single-sequence editors: ISM performs greedy in-silico mutagenesis, and LEDIDI reparameterizes the input one-hot sequence through Gumbel-softmax and follows an oracle gradient. AdaLead (Sinai et al., 2020) extends greedy editing with rejection sampling against a fitness threshold, connecting this line to model-guided directed evolution. More broadly, in silico directed-evolution and neural-network-guided sequence design methods iteratively propose variants, score them with a learned model, and retain high-scoring candidates (Vaishnav et al., 2022; Taskiran et al., 2024; Jiang et al., 2024). Bayesian optimization methods for biological sequence design, including LamBO-style approaches, also use learned surrogates to allocate expensive evaluations in sequence space (Stanton et al., 2022). These methods are adjacent to GPA in their use of learned rewards, but they do not directly target the same frozen-prior, reward-tilted sampling distribution.

Other test-time guidance methods provide per-step steering mechanisms. NOS (Gruver et al., 2023) and PROTEINGUIDE (Goel et al., 2024) guide generation through hidden-state gradients or token-level rate modulation; TFG-Flow (Ye et al., 2024) extends test-time gradient guidance to continuous flow models. For masked discrete diffusion, classifier-free or classifier guidance (Schiff et al., 2024) and the discrete-state guidance derivation of Nisonoff et al. (2024) formalize per-step rate modulation. These methods provide a range of test-time guidance mechanisms, from single-sequence editing to per-step rate or hidden-state modulation. Many operate per trajectory instead of maintaining a population. In §4.2, we use a held-out cross-oracle evaluation to test whether gains under the optimization oracle transfer to an independent evaluator. In GPA, optional gradient information enters as a proposal bias inside a population-scale SMC loop, applied to the full population at each step.

Annealed importance sampling and population annealing. The base GPA sampler builds on annealed importance sampling and annealed SMC (Neal, 2001; Del Moral et al., 2006; 2012; Naesseth et al., 2019). The same algorithmic skeleton was developed independently in statistical physics under the name *population annealing* for spin-glass Monte Carlo (Hukushima & Iba, 2003; Machta, 2010; Wang et al., 2015). We use "annealed SMC" for the formal sampling view and "population annealing" for the population-as-output intuition. Adaptive SMC scheduling (Syed et al., 2024) and waste-free SMC (Dau & Chopin, 2022) are related refinements that could be combined with the present implementation.

The branch factor K is related to proposal-selection ideas in auxiliary particle filters (Pitt & Shephard, 1999) and to tournament selection in evolutionary computation (Blickle & Thiele, 1996). The pool-as-output view is closest in spirit to evolutionary algorithms, but the selection and reweighting steps are organized through an annealed-SMC target instead of a hand-designed evolutionary loop. The recent "Global Annealing" work is the closest neighbor by name (Del Bono et al., 2025), combining population annealing with generative-model-proposed global moves for spin-glass combinatorial optimization. GPA differs in its application domain and target: the proposal comes from a frozen biological sequence generator, and the target is an oracle-tilted biological sequence distribution, distinct from the Ising or QUBO energies considered there. Population-based SMC steering has also been explored for language models (Lew et al., 2023; Loula et al., 2025); GPA applies the same broad inference-time sampling idea to biological sequence design with generator-defined mutation proposals.

Generative Flow Networks for biological sequence design. GFlowNets (Jain et al., 2022) train an amortized sampler whose marginal distribution is proportional to a reward, with diversity encouraged by the trajectory-balance objective. They share GPA's motivation of sampling high-reward sequences and not only optimizing a single best candidate. The main difference is where adaptation occurs: GFlowNets train a task-specific sampler against the reward, while GPA uses a frozen pretrained generator and applies reward guidance at inference time. These approaches are not mutually exclusive; a trained GFlowNet could in principle serve as a proposal mechanism inside a population-based sampler.

Multi-objective biological sequence design. Multi-objective design has been studied with Pareto-guided continuous diffusion (Yao et al., 2024), MCTS-guided discrete diffusion for therapeutic peptides (Tang et al., 2025), and multi-objective-guided discrete flow matching with hypercone filters (Chen et al., 2025a). In GPA, the off-target penalty pw is a linear scalarization with a Lagrangian interpretation for a relaxed off-target-constrained objective (Appendix C.4). Sweeping pw evaluates different points along the observed activity–specificity trade-off without retraining the generator. Other inference-time scaling methods for diffusion, including particle Gibbs sampling (Dang et al., 2025), replica exchange (He et al., 2025), drift-controlled SMC (Ren et al., 2025), initial-particle alignment (Yoon et al., 2025), and training-free reward alignment (Kim et al., 2025), are related directions in inference-time steering.

Generative backbones used in this paper. We evaluate GPA with two generator classes. For enhancer benchmarks, we use MDLM (Sahoo et al., 2024), a masked discrete-diffusion model with a partial-mask$\circ$denoise mutation proposal. For promoter design, we use HyenaDNA (Nguyen et al., 2023), an autoregressive sequence model with partial-context resampling. DNA-CRAFT (Awasthi et al., 2026) reports its strongest results on a Mamba-style discrete-diffusion backbone, DiMamba, and reports DiMamba-vs-MDLM ablations indicating that the backbone contributes to biological-fidelity metrics. We did not run GPA on DiMamba because the corresponding code and pretrained checkpoints were not publicly available. Other sequence generators, including SEDD (Lou et al., 2023; Sarkar et al., 2024), D3PM (Austin et al., 2021), and large causal models such as EVO 2 (Brixi et al., 2026), could be used if they expose an appropriate mutation or resampling interface, but those combinations are not evaluated here.

G. Hyperparameters

Table 7 lists the GPA hyperparameters for the SMC-protocol comparison on Gosai HepG2 enhancers. Table 8 lists the GPA recipe from Table 3 used for the CTRL-DNA promoter comparison. Table 9 lists the two GPA recipe

variants used for the DNA-CRAFT enhancer comparison: the headline recipe from Table 1 with a soft GC-centering term in the DPS update, and the same recipe without that. The effect of this GC-centering term is evaluated in Appendix H. The code to fully reproduce our results are available at https://anonymous.4open.science/status/gpa-paper-release-7A2A.

Table 7. Hyperparameters for the SMC-protocol comparison on Gosai HepG2 enhancers (MDLM backbone, DRAKES split-oracle activity head).

Parameter	Table 2 GPA recipe
N (population)	5,000
α (ESS threshold)	0.5
β_* (target inverse temperature)	5, 10 or 15
$T_{\max}$ (max steps)	30
nf (mask fraction)	0.10
K (branch factor)	8
λ (population-conformity penalty)	1.0
DPS gradient strength η	3,000

Table 8. Hyperparameters for the CTRL-DNA promoter comparison (Reddy promoters, HyenaDNA backbone, gReLU MSE oracles).

Parameter	Table 3 GPA recipe
N (population)	5,000
α (ESS threshold)	0.5
β_* (target inverse temperature)	100
$T_{\max}$ (max steps)	60
nf (mask fraction)	0.05
K (branch factor)	8
pw (off-target penalty weight)	0.35
λ (population-conformity penalty)	1.0

Table 9. Hyperparameters for the DNA-CRAFT enhancer comparison (Gosai 50/50 split, top-64 selection by greedy pool-composite rule). Table 1 GPA recipe has a soft GC-centering term with target $\mathrm{gct} = 0.60$ inside the DPS update; the other recipe is same except the GC-centering term.

Parameter	Table 1 GPA without GC-centering	Table 1 GPA
N (population)	5,000	5,000
β_* (target inverse temperature)	25	25
pw (off-target penalty weight)	0.30	0.30
DPS	on	on
GC-centering target gct	—	0.60
K (branch factor)	8	8

H. DNA-CRAFT ablation for GC penalty

The headline GPA column in Table 1 uses a soft GC-centering term inside the DPS update. This section compares that recipe to an otherwise matched variant without the GC-centering term, under the same DNA-CRAFT matched-output-budget protocol: `by_pool_composite` top-64 selection from a 5,000-sequence final pool, with 3 independent runs for each variant. Both variants use the MDLM backbone and run in approximately 3.6 min/seed on a single H100 without MCTS-style planning. The comparison isolates the effect of the GC-centering term while holding K, β_*, pw, N, and DPS fixed.

Recipe decomposition. Each component below maps to a specific knob in the GPA algorithm of §3:

- `dps`: enables the Gumbel-softmax oracle-gradient proposal with strength $\eta = 3000$ (§3.2). This gradient-biased proposal is shared by both recipes, so the ablation does not test DPS itself.
- `pw030`: sets the off-target penalty weight $\mathrm{pw} = 0.30$ in the linear fitness of Eq. 3. The same value is used in the DPS objective.
- `b25`: sets the SMC tilt ceiling $\beta_* = 25$. Both variants use this value.
- `gct060`: adds the soft GC-centering term used only in the headline recipe. The term is

$$-w\,(g(x) - 0.60)^2,$$

where $g(x) = \overline{P(C) + P(G)}$ is the per-position soft-onehot GC fraction and $w = 10$. This term is included in the differentiable DPS objective; it is not a hard accept/reject filter.

Both variants additionally use branch factor $K = 8$ and population size $N = 5{,}000$. Because $K > 1$ changes the proposal distribution (Prop. C.3), both recipes should be interpreted as using the same K-branch proposal modification; the contrast below isolates the GC-centering term.

Because both recipes use the same DPS setting, this comparison should be read as an ablation of the GC-centering term within the gradient-biased proposal, not as a comparison of gradient-guided versus non-gradient-guided sampling. The two GPA variants give similar MinGap values across the three cell types: the no-GC variant is slightly higher on K562 and SK-N-SH ($+0.05$ and $+0.09$), while the GC-centered recipe is slightly higher on HepG2 ($+0.04$). The larger differences are in diversity and fidelity metrics. The GC-centered recipe improves diversity in all three cell types ($+0.05$, $+0.27$, and $+0.27$), improves motif correlation in all three cell types ($+0.02$, $+0.01$, and $+0.03$), and improves 3-mer correlation in K562 and SK-N-SH ($+0.03$ and $+0.07$). We therefore use the GC-centered recipe as the headline result; the no-GC variant is included to isolate the effect of the GC-centering term.

Several baselines report Diversity $= 1.98$ across cell types, near the apparent saturation point of the DNA-CRAFT diversity metric. Both GPA variants remain below this ceiling,

Table 10. DNA-CRAFT benchmark with the GPA alternate recipe (`dps_pw030_b25`; no GC-centering term), reported alongside the headline GPA recipe. Both GPA columns use `by_pool_composite` top-64 selection from the final pool. Values are means over 3 runs; standard deviations are omitted here for readability. Small differences between the two GPA variants should therefore be interpreted as descriptive rather than statistically resolved. Baselines are taken from Awasthi et al. (2026).

Cell	Metric	SMC	CG	TDS	DRAKES	D3	Ledidi	Ctrl-DNA	DNA-CRAFT	GPA, no GC	GPA
HepG2	MinGap	1.61	−0.23	0.40	−1.40	0.05	5.77	**7.79**	4.35	6.87	6.91
	Motif	0.55	0.86	0.40	0.06	0.87	0.58	0.63	**0.92**	0.90	**0.92**
	3-mer	0.81	0.97	0.74	−0.36	**0.98**	0.76	0.49	**0.98**	0.95	0.95
	Diversity	0.83	**1.98**	0.96	1.86	**1.98**	**1.98**	1.90	**1.98**	1.81	1.86
K562	MinGap	4.12	0.00	1.62	−0.20	0.18	7.66	**9.07**	5.69	8.18	8.13
	Motif	0.45	0.85	0.51	0.14	0.86	0.65	0.63	0.93	0.93	**0.94**
	3-mer	0.66	0.94	0.65	−0.35	0.96	0.69	0.41	**0.98**	0.92	0.95
	Diversity	0.31	**1.98**	0.64	1.96	**1.98**	**1.98**	1.90	**1.98**	1.56	1.83
SK-N-SH	MinGap	0.56	−0.28	0.19	0.09	−0.01	3.03	3.72	3.23	**5.07**	4.98
	Motif	0.52	0.86	0.48	0.23	0.84	0.38	0.48	0.88	0.89	**0.92**
	3-mer	0.78	0.95	0.72	−0.38	0.93	0.37	0.20	**0.97**	0.87	0.94
	Diversity	1.27	**1.98**	0.92	1.83	1.97	**1.98**	1.86	**1.98**	1.59	1.86

with the GC-centering term partially closing the gap. DNA-CRAFT also retains a consistent 3-mer Pearson advantage over GPA across all three cell types (0.98 vs. 0.95 in HepG2, 0.98 vs. 0.95 in K562, and 0.97 vs. 0.94 in SK-N-SH). Conversely, GPA matches or exceeds DNA-CRAFT on motif correlation and improves MinGap relative to DNA-CRAFT in all three cell types. Ctrl-DNA achieves the highest MinGap on HepG2 and K562, exceeding GPA by 0.88 and 0.94, respectively, while GPA is higher on SK-N-SH. This is consistent with the broader activity–fidelity trade-off in Table 1: reward-fine-tuned methods can reach higher activity in some cells, whereas GPA preserves stronger motif and 3-mer fidelity than Ctrl-DNA under the matched-output protocol.

Both GPA variants run at approximately 3.6 min/seed on a single H100 without MCTS-style tree expansion or per-task policy optimization. Matched wall-times for DNA-CRAFT are not available because the DiMamba checkpoints and code were not publicly released.

One possible explanation for the GC-centering effect is that the term discourages the DPS update from moving the soft sequence representation toward narrow GC-biased regions. This interpretation is consistent with the observed diversity increase, but the ablation does not isolate all possible sequence-level effects of the term.

Backbone caveat. As noted in §4, the DNA-CRAFT values in Table 1 were reported using a Mamba-style discrete-diffusion backbone, DiMamba, whose code and pretrained checkpoints were not publicly available for our experiments. DNA-CRAFT also reports DiMamba-vs-MDLM ablations indicating that the backbone contributes to biological-fidelity metrics. Our GPA results use the publicly available MDLM backbone of Sahoo et al. (2024). The comparison should therefore be read with this backbone mismatch in mind, and not as a fully controlled comparison of the search algorithms alone.

I. DNA-CRAFT K and i scaling sweeps (UCB-MCTS variant)

Motivation. DNA-CRAFT uses Monte-Carlo Tree Search (MCTS) on a DiMamba diffusion backbone and reports strong motif and 3-mer fidelity on the Gosai enhancer benchmark. Standard GPA already includes a depth-1 proposal-selection step through the K-branch operator: draw K candidate mutations and keep the highest-reward proposal. The `ucb_stacked` variant tests whether adding deeper UCB-style planning inside each GPA mutation step improves motif or k-mer metrics where the standard recipe lags DNA-CRAFT in Appendix H.

The `ucb_stacked` mutation kernel. At each SMC mutation step, each particle $x_t^{(j)}$ becomes the root of an independent Monte Carlo tree. We run i MCTS iterations on that tree. Each iteration follows the standard UCT loop (Kocsis & Szepesvári, 2006): select a leaf using UCB1, expand it with K candidate children, score the children, and back-propagate the observed rewards.

The selection rule is

$$\mathrm{UCB1}(a) = \bar{Q}(a) + c\sqrt{\frac{\ln n(v)}{n(a)}}, \tag{6}$$

where $\bar{Q}(a) \in [0, 1]$ is the range-normalized mean reward backed up through child a, $n(a)$ is the visit count of child a, $n(v)$ is the visit count of the parent node, and $c = \sqrt{2}$ is the exploration constant (Auer et al., 2002; Kocsis & Szepesvári, 2006). Unvisited children are selected first by convention. Expansion uses the same proposal operator as the standard GPA recipe: partial-mask∘denoise with DPS and branch factor K. We bound the tree depth at $d = 2$, so the planner looks at most two mutation steps

ahead. After i iterations, the mutated state for particle $x_t^{(j)}$ is the highest-reward leaf in that tree. The outer SMC loop then continues with reweighting, resampling, and mutation as in Algorithm 1.

Relation to DNA-CRAFT. The `ucb_stacked` variant uses the same broad MCTS ingredients as DNA-CRAFT: UCB selection, child expansion, reward evaluation, and reward back-propagation. The algorithms differ in how the tree search is used. In DNA-CRAFT, the tree search is the full design procedure: it uses $N_{\text{iter}} = 64$ MCTS iterations, expands $M = 128$ children per expansion, performs full leaf-to-clean rollouts, and returns a MinGap-Set archive of $N_{\max} = 64$ designs. In `ucb_stacked`, many small trees are run, one per particle and per SMC mutation step, and each tree defines only a local mutation proposal. The final output remains the SMC population. The name `ucb_stacked` refers to this layering: UCB-style planning is stacked on top of the K-branch+DPS proposal inside the SMC mutation step.

Cost and sweep design. Each `ucb_stacked` mutation step evaluates approximately iK scored nodes per particle, compared with K proposals in the standard K-branch mutation step. At $K = 8, i = 18$, this is up to 145 total nodes per particle per mutation step including the root. The variant remains substantially more expensive than the standard GPA recipe: $K = 8, i = 12$ takes 31.35 min/seed, approximately $8.7\times$ the 3.6 min/seed standard non-MCTS recipe used in Table 1; $K = 8, i = 18$ takes 32.26 min/seed, approximately $9.0\times$ the standard recipe.

We evaluate two K562 sweeps under the matched-output-budget protocol of Table 1: top-64 selection by `by_pool_composite` from the final pool, with tree depth fixed at $d = 2$. The i-axis sweep fixes $K = 8$ and varies i. The K-axis sweep fixes $i = 12$ and varies K. Wall-time is reported per seed on a single H100. The pilot anchor $K = 8, i = 12$ uses 5 seeds; larger settings use 2 seeds.

For reference, the standard non-MCTS GPA recipe on K562 reaches MinGap $= 8.13$, Motif $= 0.94$, 3-mer $= 0.95$, and Diversity $= 1.83$ at approximately 3.6 min/seed (Appendix H). The `ucb_stacked` variant should therefore be read as a planning-augmented alternative with substantially higher wall-time.

The i-axis sweep shows that tree-search planning mainly affects the metrics where the standard recipe has the largest remaining gaps to DNA-CRAFT. From $i = 12$ to $i = 24$, 3-mer Pearson increases from 0.945 to 0.968, and Diversity increases from 1.903 to 1.941. MinGap remains in a narrow range (8.02–8.15), and Motif remains near 0.94. Increasing to $i = 36$ does not produce further improvement. Thus, deeper planning partially closes the 3-mer and diversity gaps, but does not materially change MinGap or Motif under this protocol.

The K-axis sweep shows no clear benefit from expanding beyond $K = 16$ in this setting. $K = 16$ gives the highest 3-mer Pearson in the sweep (0.959) and a MinGap comparable to the other settings, but larger values of K increase wall-time and do not give monotone improvements. We therefore keep $K = 8$ as the default headline setting and treat larger K as a higher-cost option that may be useful only when the additional 3-mer or diversity gain is worth the compute.

These results do not rule out benefits from tree-search planning in other settings, but they do not justify using `ucb_stacked` as the main recipe here. The standard GPA recipe is therefore used for the headline DNA-CRAFT comparison, and `ucb_stacked` is reported as an optional higher-cost variant.

J. Population-size (N) ablation

The headline GPA numbers in Table 1 use a population of $N = 5{,}000$ sequences. In the idealized SMC setting, empirical averages converge to the reward-tilted target at rate $\mathcal{O}(1/\sqrt{N})$ (Theorem C.1). In the design setting, N also controls the size of the candidate pool available for post-hoc matched-budget selection. We therefore test how performance changes as the population size increases, holding the standard GPA recipe and the `by_pool_composite` top-64 selection rule fixed. We sweep $N \in \{128, 500, 1000, 2500, 5000, 10000, 20000\}$ across all three cell types of the DNA-CRAFT enhancer benchmark, with 5 SMC seeds per (N, cell) pair. This sweep spans approximately 2.2 orders of magnitude in population size.

Across the full range from $N = 128$ to $N = 20{,}000$, all three cell types show large improvements in the matched-budget metrics. MinGap increases by $+1.03$ in HepG2, $+1.08$ in K562, and $+2.98$ in SK-N-SH. Motif correlation increases by $+0.10$ to $+0.12$, 3-mer correlation by $+0.20$ to $+0.27$, and diversity by $+0.42$ to $+0.53$. These gains support the population-as-budget view: larger SMC populations provide more candidate sequences for the fixed top-64 selection rule. The SK-N-SH effect is largest: it starts from the lowest MinGap at $N = 128$ and continues to improve through $N = 20{,}000$, indicating that harder cell-type tasks can benefit more from larger populations.

The relevant trend is overall improvement with increasing population size and diminishing returns at larger N. Some adjacent-N values fluctuate within run-to-run variation, especially on smaller-effect metrics; for example, SK-N-SH Motif and 3-mer decrease slightly from $N = 5{,}000$ to $N = 10{,}000$. This ablation should also not be read as a direct empirical verification of the $\mathcal{O}(1/\sqrt{N})$ SMC rate,

Table 11. DNA-CRAFT i-axis sweep for the `ucb_stacked` variant (K562, $K = 8$, $d = 2$). Results use `by_pool_composite` top-64 selection from the final pool. Values are mean(std) across n seeds.

i	n	wall (min)	MinGap	Motif	3-mer	Diversity
12	5	31.35 (3.62)	+8.066 (0.063)	0.940 (0.004)	0.945 (0.023)	1.903 (0.022)
18	5	32.26 (0.62)	+8.115 (0.013)	0.942 (0.003)	0.961 (0.003)	1.926 (0.027)
24	2	37.07 (4.08)	+8.153 (0.075)	0.940 (0.002)	0.968 (0.003)	1.941 (0.012)
36	2	46.60 (3.23)	+8.015 (0.001)	0.941 (0.003)	0.965 (0.002)	1.937 (0.014)

Table 12. DNA-CRAFT K-axis sweep for the `ucb_stacked` variant (K562, $i = 12$, $d = 2$). Results use `by_pool_composite` top-64 selection from the final pool. Values are mean(std) across n seeds. The displayed 0.000 standard deviation for one motif row reflects rounding at the shown precision.

K	n	wall (min)	MinGap	Motif	3-mer	Diversity
8	5	31.35 (3.62)	+8.066 (0.063)	0.940 (0.004)	0.945 (0.023)	1.903 (0.022)
16	2	49.39 (6.64)	+8.124 (0.108)	0.939 (0.002)	0.959 (0.008)	1.898 (0.043)
24	2	69.00 (13.13)	+8.070 (0.016)	0.942 (0.004)	0.953 (0.007)	1.935 (0.008)
32	2	77.40 (4.31)	+8.022 (0.103)	0.936 (0.000)	0.953 (0.002)	1.907 (0.045)

since the reported quantities include top-64 post-hoc selection and nonlinear pool-level metrics.

The headline $N = 5{,}000$ setting is a runtime/benefit compromise. Increasing from $N = 5{,}000$ to $N = 20{,}000$ improves every metric in every cell type, but the gains are modest relative to the $4\times$ increase in population size: averaged across cells, approximately $+0.22$ MinGap, $+0.010$ Motif, $+0.015$ 3-mer, and $+0.045$ diversity. At the standard runtime of approximately 3.6 min/seed for a 5,000-sequence pool, the headline setting corresponds to roughly 0.043 sec per generated candidate before matched-budget selection. Wall-time scales approximately linearly with N at fixed K, β_*, and number of SMC steps, so $N = 20{,}000$ would require roughly 14–15 min/seed under the same recipe. We therefore use $N = 5{,}000$ as the headline operating point for Table 1; practitioners with a larger wall-time budget can use larger populations for additional gains.

Population scaling improves GPA but does not remove all differences to the strongest task-aligned baselines. On K562, where CTRL-DNA reaches MinGap = 9.07 in Table 1, the $N = 20{,}000$ GPA setting reaches 8.31. On SK-N-SH, where CTRL-DNA reports MinGap = 3.72, GPA already exceeds that value at $N = 5{,}000$. Even the smallest HepG2 setting, $N = 128$, reaches MinGap = 6.05, above the Ledidi (5.77) and DNA-CRAFT (4.35) reference values in Table 1. These comparisons are descriptive because the methods differ in optimization procedure, backbone, and compute budget, but they show where population scaling changes the benchmark picture.

All runs in this ablation use the same standard GPA recipe as Table 1 ($K = 8$, DPS, GC-centering term, no MCTS, MDLM backbone); only the SMC population size N is varied. The $N = 5{,}000$ rows here use a 5-seed sweep, while Table 1 and Appendix H report separate 3-seed runs; small differences between the corresponding rows reflect different seed sets and are within the run-to-run variation reported here.

K. Using an encoder-only AlphaGenome fine-tuned model as a held-out oracle

For the held-out cross-oracle experiments in §4.2, we use an encoder-only AlphaGenome model fine-tuned for MPRA activity prediction. The model attaches a prediction head to the AlphaGenome encoder and is fine-tuned for the K562 LentiMPRA task. We use it only as an independent evaluation oracle; it is never used for GPA sampling, temperature selection, proposal scoring, or post hoc selection.

This oracle is used only for evaluation in the cross-oracle experiments. The optimization oracle is LegNet K562, whereas the AlphaGenome-derived model is held out from the optimization loop. Both LegNet and the AlphaGenome-derived oracle are learned sequence-to-function models trained on MPRA-style data. Cross-oracle agreement therefore measures consistency between independently trained predictors and does not constitute wet-lab validation. The advantage over single-oracle reporting is that sequence features exploited by only one model are less likely to transfer to the held-out evaluator.

L. GPA-vs-ISM/LEDIDI runtime comparison

Table 14 reports wall-time for producing a 5,000-sequence K562 enhancer pool on a single H100, averaged across the random and natural seed pools used in §4.2. The comparison is matched by output volume and hardware; output quality on this task is reported in §4.2, and this appendix addresses runtime only. GPA propagates the full population

Table 13. Population-size ablation on the DNA-CRAFT enhancer benchmark. All runs use the standard GPA recipe from Table 1 without MCTS and the `by_pool_composite` top-64 selection rule. Values are mean(std) over 5 SMC seeds. Increasing N generally improves the matched-budget design metrics, with diminishing gains beyond the headline $N = 5{,}000$ setting.

N	MinGap ↑	Motif ↑	3-mer ↑	Diversity ↑
HepG2				
128	6.052 (0.087)	0.829 (0.036)	0.757 (0.084)	1.415 (0.101)
500	6.559 (0.153)	0.884 (0.022)	0.843 (0.070)	1.600 (0.167)
1,000	6.677 (0.130)	0.895 (0.008)	0.923 (0.021)	1.725 (0.104)
2,500	6.788 (0.129)	0.912 (0.005)	0.938 (0.024)	1.845 (0.029)
5,000	6.962 (0.076)	0.915 (0.006)	0.952 (0.013)	1.870 (0.056)
10,000	7.013 (0.189)	0.920 (0.003)	0.968 (0.005)	1.908 (0.014)
20,000	**7.081** (0.149)	**0.928** (0.006)	**0.972** (0.004)	**1.912** (0.020)
K562				
128	7.236 (0.160)	0.871 (0.012)	0.774 (0.104)	1.515 (0.120)
500	7.709 (0.175)	0.915 (0.011)	0.863 (0.044)	1.677 (0.129)
1,000	7.968 (0.083)	0.933 (0.008)	0.912 (0.106)	1.813 (0.070)
2,500	7.995 (0.082)	0.939 (0.005)	0.951 (0.017)	1.821 (0.065)
5,000	8.193 (0.086)	0.942 (0.005)	0.960 (0.010)	1.893 (0.042)
10,000	8.230 (0.118)	0.946 (0.004)	0.965 (0.007)	1.919 (0.014)
20,000	**8.313** (0.060)	**0.952** (0.004)	**0.977** (0.005)	**1.936** (0.021)
SK-N-SH				
128	2.230 (0.347)	0.807 (0.028)	0.672 (0.081)	1.384 (0.067)
500	4.149 (0.470)	0.887 (0.011)	0.844 (0.068)	1.588 (0.128)
1,000	4.388 (0.282)	0.899 (0.009)	0.901 (0.021)	1.783 (0.049)
2,500	4.841 (0.062)	0.908 (0.010)	0.923 (0.015)	1.852 (0.017)
5,000	4.980 (0.079)	0.919 (0.004)	0.937 (0.007)	1.867 (0.042)
10,000	5.054 (0.077)	0.918 (0.006)	0.934 (0.022)	1.892 (0.028)
20,000	**5.209** (0.037)	**0.926** (0.003)	**0.946** (0.011)	**1.917** (0.028)

through the SMC loop, with particles processed in GPU-memory chunks using `mutation_batch_size=128`. ISM consists of 5,000 independent greedy trajectories of 115 sequential edit steps each, with each step scoring candidate single-nucleotide edits before selecting the next edit. LEDIDI processes the 5,000 seeds in chunks of 128 through its Gumbel-softmax optimization under the settings used in §4.2.Under these settings, GPA is approximately $3.4\times$ faster than LEDIDI and $9.3\times$ faster than ISM for producing the same number of candidate sequences.

M. DPS gradient ablation for specificity

We evaluate the contribution of the optional DPS gradient proposal in the base GPA sampler on the Gosai HepG2 enhancer task using the MDLM backbone and the DRAKES split-oracle activity head. The configuration here is the simple annealed-SMC reference: $N = 5{,}000$ particles, ESS threshold $\alpha = 0.5$, a GC filter restricting samples to $[0.45, 0.55]$, and top-k initialization from natural HepG2 seeds. We toggle only the DPS gradient ($\eta = 3{,}000$) on and off, at a high-pressure operating point, target temperature $\beta_* = 100$ and mask fraction $\mathrm{nf} = 0.05$, so that each step moves a few positions under a strong reward tilt and the quality of the mutation proposal is stressed. We evaluate two reward regimes: a "Mean" track using the target-activity reward alone ($\mathrm{pw} = 0$), and a "Spec" track that adds an off-target penalty $\mathrm{pw} = 0.35$. Specificity is computed here as

$$\boxed{\mathrm{Specificity}(x) = r_{\mathrm{eval}}^{\mathrm{HepG2}}(x) - \max\left(r_{\mathrm{eval}}^{\mathrm{K562}}(x),\ r_{\mathrm{eval}}^{\mathrm{SKNSH}}(x)\right)}$$

DPS improves both evaluated quantities in this ablation. Across the mean-activity and specificity tracks, DPS adds roughly one point of held-out activity and approximately 1.5 specificity points. The specificity gain is similar with and without the explicit off-target penalty ($+1.48$ and $+1.50$), suggesting that DPS contributes more than a simple rescaling of the linear specificity objective. Because these gains appear already in the bare single-branch sampler ($K = 1$, $\lambda = 0$), they isolate the value of the gradient proposal itself, independent of the branch-factor and diversity extensions used elsewhere in the paper. A plausible interpretation is that the gradient-biased proposal helps move particles along locally high-specificity directions of the oracle landscape, although this ablation does not isolate that mechanism directly. Because DPS is not part of the base SMC target and requires differentiable oracle access, the non-DPS setting remains the appropriate reference for black-box or API-only oracles.

Table 14. Runtime comparison for producing a 5,000-sequence K562 enhancer pool on a single H100. Wall-times are averaged across the random and natural seed pools from §4.2. Speedup is computed as $\text{wall}_{\text{baseline}}/\text{wall}_{\text{GPA}}$.

Method	Wall (5K pool)	Parallelism	Relative wall-time
GPA	$\sim$11.6 **min**	population SMC, batched particles	1.0×
LEDIDI	$\sim$40 min	39 chunks × 128 seeds; **1000** gradient steps	$\sim$3.4× slower
ISM	$\sim$107.5 min	5,000 greedy trajectories; 115 edit steps	$\sim$9.3× slower

Table 15. DPS gradient ablation on the Gosai HepG2 enhancer task using the MDLM backbone and DRAKES split-oracle activity head. The "Mean" track uses the target-activity reward alone. The "Spec" track includes the off-target penalty $\text{pw} = 0.35$. Values are mean ± std. over 3 random seeds;

Track	Metric	GPA only	GPA + DPS	Δ DPS
Mean	H_{eval}	8.73 ± 0.50	$\mathbf{10.06 \pm 0.19}$	+1.33
Mean	Specificity	1.58 ± 0.31	3.06 ± 0.55	+1.48
Spec	H_{eval}	8.28 ± 0.12	9.35 ± 0.20	+1.07
Spec	Specificity	6.11 ± 1.45	$\mathbf{7.61 \pm 0.25}$	+1.50

N. Naturalness audit

We audit HepG2 top-128 design pools against simple sequence-level statistics computed from the Gosai MPRA distribution. This analysis is intended as a diagnostic for obvious sequence pathologies, not as experimental validation of designed activity. We compare GPA and CTRL-DNA using the same five-axis audit. The GPA recipe used here omits the diversity penalty λ, uses DPS, and includes the specificity weight $\text{pw} = 0.35$.

Real Gosai values are computed over the top-128 HepG2 specificity-ranked sequences from the full Gosai MPRA dataset ($\sim$735K sequences). This should be interpreted when comparing against top-ranked design pools, since top-activity sequences may differ from the full empirical sequence distribution. The GC tolerance used for the audit is wider than the GC filter used during GPA sampling ($[0.40, 0.60]$ in the audit vs. $[0.45, 0.55]$ during sampling). Of the six criteria in `bio_pass_rate`, only the GC range overlaps with the generation-time filter, and the audit range is wider. Thus, GPA outputs pass the GC criterion by construction, while the remaining five criteria, entropy, CpG, AT asymmetry, GC asymmetry, and homopolymer length, are independent checks under this audit.

Audit components.

- **GC mean.** Per-sequence GC fraction averaged over the pool. The real Gosai reference is near 0.44.
- **bio_pass_rate.** The fraction of sequences satisfying all six sequence-level criteria:
  1. GC content in $[0.40, 0.60]$;
  2. per-sequence Shannon entropy over base frequencies ≥ 1.9 bits, with $\log_2 4 = 2.0$ as the maximum;
  3. CpG dinucleotide frequency ≤ 0.072;
  4. AT strand asymmetry $|f(A) - f(T)| < 0.15$;
  5. GC strand asymmetry $|f(G) - f(C)| < 0.15$;
  6. maximum homopolymer run ≤ 12 bp.

  A sequence must pass all six criteria to count toward `bio_pass_rate`. Since only 51% of real Gosai sequences pass this conjunction, this metric should be read as a stringent pathology screen; it is not a definitive measure of biological realism.
- **Maximum homopolymer.** The mean over the pool of the per-sequence longest same-base run; real Gosai has typical maxima near 5 bp.
- $k = 3$ **Pearson.** Pearson correlation between the pool-level 3-mer count vector ($4^3 = 64$ bins) and the same vector computed on the real Gosai reference pool.
- **HepG2 motif rates.** Hit rates for HepG2-associated TF motifs scanned by the audit script: `HNF4A`, `CEBPA`, `FOXA1`, `FOXA2`, and `HNF1A`. Motifs are scanned with IUPAC-aware regular expressions in both forward and reverse-complement orientation. We report hit rates rather than raw fold-enrichment when the real-background hit rate is zero or near zero, since fold-enrichment is then unstable or undefined.

Table 16. Naturalness audit on HepG2 top-128 design pools. The audit is a sequence-level pathology screen, not experimental validation. Bold values indicate the most favorable design-method value for each row, except where noted.

Metric	Real (Gosai)	GPA	CTRL-DNA
GC mean	0.44	0.41	0.45
bio_pass_rate	0.51	**1.00**	0.05
max homopolymer	5.4	**5.1**	20.9
k=3 Pearson	1.0	**0.61**	0.39

Under this audit, the GPA pool passes all six sequence-level criteria for all top-128 sequences, while only 5% of the CTRL-DNA pool passes the same conjunction. Because the GC criterion overlaps with the GPA generation-time filter, the more informative point is that GPA also passes the five

Table 17. HepG2-associated motif hit rates in the naturalness audit. Motifs are scanned in both forward and reverse-complement orientation using the same IUPAC-aware matching rule, and reported as fold-enrichment over the Real (Gosai) hit rate, matching the convention of Table 16. †: motif absent from the Real reference set, so the enrichment is computed against the audit script's $\varepsilon = 10^{-6}$ floor and should be read as "present in design pool, absent in natural reference," not as a calibrated fold-change.

Motif	Real (Gosai)	GPA	CTRL-DNA
HNF4A	1×	**10.2**×	5.6×
CEBPA†	0×	240×	0×
FOXA1	1×	0.04×	**1.7**×
FOXA2	1×	0.04×	**1.7**×
HNF1A†	0×	**10,040**×	7,812×

non-filtered criteria at 100%: entropy, CpG, AT asymmetry, GC asymmetry, and homopolymer length.

The strongest difference is the homopolymer statistic. GPA has a mean maximum homopolymer length of 5.1 bp, close to the real Gosai value of 5.4 bp, whereas CTRL-DNA has a mean maximum homopolymer length of 20.9 bp. This indicates a clear low-complexity sequence feature pattern in the CTRL-DNA top designs under this audit, consistent with an oracle-overfit failure mode.

The real Gosai `bio_pass_rate` of 0.51 also calibrates the audit. The conjunction is conservative: approximately half of real sequences contain at least one flagged feature. Thus, GPA's 100% pass rate should not be interpreted as evidence that GPA sequences are more biologically realistic than real sequences. It indicates that this particular design pool lies in a tighter, less pathological region under the audit criteria.

The 3-mer Pearson values show that neither design pool fully matches the real Gosai trinucleotide spectrum. GPA is closer to the reference than CTRL-DNA (0.61 vs. 0.39), but this remains a moderate correlation; the design pool does not fully match the real distribution. The motif analysis should also be interpreted cautiously. Motif presence is a useful cell-type-relevant diagnostic, but motif enrichment alone does not establish biological plausibility, especially when accompanied by low-complexity sequence features.

Overall, this audit supports a limited conclusion: GPA avoids the most obvious sequence pathologies detected by this battery, whereas the CTRL-DNA top designs contain long homopolymer tracts and fail the combined sequence-level screen. Removing the GC-centering term (Appendix H) changes diversity and fidelity metrics, but the GPA no-GC outputs still avoid the long homopolymer tracts seen in CTRL-DNA. The homopolymer-avoidance pattern is therefore more consistent with the pretrained generator's sequence prior than with the explicit GC filter alone. The filter is a backstop; the generator does most of the work. This audit is not a substitute for experimental validation and does not prove that all GPA designs are biologically functional.

O. Additional Tables

Table 18. DNA-CRAFT enhancer benchmark with standard deviations. We followed top-64 sequence selection by `by_pool_composite`, matched to DNA-CRAFT's G^* at $N_{\max} = 64$. We ran 3 replicates of GPA with a 5,000-particle SMC pool and report mean(std). Baseline numbers are taken from Awasthi et al. (2026).

Cell	Metric	SMC	CG	TDS	DRAKES	D3	Ledidi	Ctrl-DNA	DNA-CRAFT	**GPA**
HepG2	MinGap ↑	1.61 (1.67)	−0.23 (0.10)	0.40 (0.57)	−1.40 (0.05)	0.05 (0.03)	5.77 (0.05)	**7.79** (0.07)	4.35 (0.05)	6.91 (0.10)
	Motif ↑	0.55 (0.05)	0.86 (0.01)	0.40 (0.10)	0.06 (0.01)	0.87 (0.01)	0.58 (0.03)	0.63 (0.05)	**0.92** (0.01)	**0.92** (0.01)
	3-mer ↑	0.81 (0.10)	0.97 (0.00)	0.74 (0.10)	−0.36 (0.01)	**0.98** (0.00)	0.76 (0.01)	0.49 (0.03)	**0.98** (0.01)	0.95 (0.00)
	Diversity ↑	0.83 (0.43)	**1.98** (0.00)	0.96 (0.10)	1.86 (0.00)	**1.98** (0.00)	**1.98** (0.00)	1.90 (0.03)	**1.98** (0.00)	1.86 (0.06)
K562	MinGap ↑	4.12 (0.89)	0.00 (0.05)	1.62 (1.61)	−0.20 (0.07)	0.18 (0.07)	7.66 (0.15)	**9.07** (0.17)	5.69 (0.04)	8.13 (0.05)
	Motif ↑	0.45 (0.03)	0.85 (0.03)	0.51 (0.13)	0.14 (0.02)	0.86 (0.04)	0.65 (0.04)	0.63 (0.08)	0.93 (0.01)	**0.94** (0.01)
	3-mer ↑	0.66 (0.13)	0.94 (0.01)	0.65 (0.20)	−0.35 (0.01)	0.96 (0.02)	0.69 (0.02)	0.41 (0.06)	**0.98** (0.00)	0.95 (0.02)
	Diversity ↑	0.31 (0.11)	**1.98** (0.00)	0.64 (0.52)	1.96 (0.00)	**1.98** (0.00)	**1.98** (0.00)	1.90 (0.02)	**1.98** (0.00)	1.83 (0.08)
SK-N-SH	MinGap ↑	0.56 (0.15)	−0.28 (0.01)	0.19 (0.33)	0.09 (0.05)	−0.01 (0.01)	3.03 (0.22)	3.72 (0.18)	3.23 (0.02)	**4.98** (0.03)
	Motif ↑	0.52 (0.15)	0.86 (0.03)	0.48 (0.09)	0.23 (0.02)	0.84 (0.01)	0.38 (0.04)	0.48 (0.04)	0.88 (0.03)	**0.92** (0.01)
	3-mer ↑	0.78 (0.04)	0.95 (0.01)	0.72 (0.03)	−0.38 (0.00)	0.93 (0.01)	0.37 (0.02)	0.20 (0.17)	**0.97** (0.01)	0.94 (0.01)
	Diversity ↑	1.27 (0.11)	**1.98** (0.00)	0.92 (0.21)	1.83 (0.00)	1.97 (0.00)	**1.98** (0.00)	1.86 (0.09)	**1.98** (0.00)	1.86 (0.04)

Table 19. **Gosai HepG2 sequence-design benchmark.** Pred-Activity and App-Log-Lik are reported as medians; ATAC-Acc, 3-mer Corr, and JASPAR Corr are reported as means; all values are averaged across 3 random seeds. Baseline rows from Pretrained through SMC_{amot} are collected from Pani et al. (2025), originally from DRAKES (Wang et al., 2024). GPA rows are appended at the bottom.

Method	Pred-Activity ↑	ATAC-Acc ↑ (%)	3-mer Corr ↑	JASPAR Corr ↑	App-Log-Lik ↑
Pretrained	0.17 (0.04)	1.5 (0.2)	−0.061 (0.034)	0.249 (0.015)	−261 (0.6)
CG	3.30 (0.00)	0.0 (0.0)	−0.065 (0.001)	0.212 (0.035)	−266 (0.6)
CFG	5.04 (0.06)	92.1 (0.9)	0.746 (0.001)	0.864 (0.011)	−265 (0.6)
DRAKES w/o KL	6.44 (0.04)	82.5 (2.8)	0.307 (0.001)	0.557 (0.015)	−281 (0.6)
DRAKES	5.61 (0.07)	92.5 (0.6)	0.887 (0.002)	**0.911 (0.002)**	−264 (0.6)
SGDD ($\beta = 30$)	8.85 (0.07)	90.9 (0.00)	0.470 (0.014)	0.466 (0.015)	−263 (1.6)
SGDD ($\beta = 50$)	**9.32 (0.04)**	96.4 (0.01)	0.370 (0.010)	0.398 (0.001)	−269 (0.1)
SVDD ($N = 8$)	6.57 (0.01)	67.4 (0.01)	0.813 (0.009)	0.753 (0.011)	−258 (0.2)
SVDD ($N = 16$)	6.89 (0.04)	84.3 (0.01)	0.891 (0.009)	0.834 (0.011)	−260 (0.2)
SMC_{amot} ($N = 1$)	5.40 (0.02)	82.1 (0.01)	0.653 (0.001)	0.778 (0.005)	−259 (0.1)
SMC_{amot} ($N = 8$)	6.35 (0.01)	95.8 (0.01)	0.736 (0.003)	0.845 (0.005)	−261 (0.2)
SMC_{amot} ($N = 16$)	6.68 (0.02)	**97.6 (0.01)**	0.796 (0.005)	0.886 (0.002)	−261 (0.4)
GPA ($\beta = 5$)	6.82 (0.32)	92.3 (4.7)	**0.912 (0.037)**	0.813 (0.069)	−274 (5.3)
GPA ($\beta = 10$)	8.04 (0.15)	93.5 (2.1)	0.794 (0.071)	0.711 (0.064)	−262 (4.3)
GPA ($\beta = 15$)	8.45 (0.06)	95.3 (1.8)	0.563 (0.082)	0.551 (0.042)	**−257 (1.5)**