UNDERSTANDING AND IMPROVING UNIQUE HIT SCALING IN PROTEIN BINDER DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative models are accelerating the discovery of proteins that bind specific molecular targets, creating new opportunities for therapeutic development. The goal of an in silico campaign is to propose a diverse library of promising candidates for experimental testing, motivating the objective of maximizing unique hits under a fixed compute budget. However, how unique hit scales with compute remains poorly characterized, and methods focused on improving hit rate have produced mixed results. We propose a cluster-level view of binder design that decomposes unique hit scaling into two processes: discovering structural clusters through generation and discovering hits within those clusters through evaluation. This decomposition provides a predictive model of campaign yield and exposes a new axis for scaling: allocation of compute between generation and evaluation. We introduce a cluster-guided allocation (CGA) policy that estimates two processes online and allocates compute according to the expected unique hit yield per unit cost. Across 19 targets and three generators, the two-process decomposition characterizes the campaign well, and CGA outperforms naive sampling, showing consistent and predictable gains not seen in previous reward-tilting methods. Beyond these gains, the framework provides a mechanistic understanding of binder design scaling and reveals complementary opportunities.

1 INTRODUCTION

Proteins that bind specific molecular targets are central to biological research and many medicines. Such binders can detect molecules, modulate pathways, and direct therapeutic activity to particular cells or tissues. Recent diffusion and flow models generate candidate structures conditioned on molecular targets (Yim et al., 2023; Ingraham et al., 2023; Geffner et al., 2025). Experimental studies have validated functional binders and recovered structures that closely match their computational designs (Zambaldi et al., 2024; Pacesa et al., 2025).

A typical *in silico* binder design campaign generates candidate binders, evaluates the resulting target-binder complexes with structure prediction models and other computational filters, and removes near-duplicate structures (Cao et al., 2022; Zambaldi et al., 2024; Bennett et al., 2023; Watson et al., 2023). The goal of such a campaign is to translate a fixed compute into a *diverse* library of promising candidates for experimental testing. Generating additional computational hits does not necessarily expand this library: structurally similar hits may share binding modes and, consequently, failure modes. This is reflected in the clustering step at the end of a campaign, making the number of **unique hits**, or hit clusters, under a fixed **compute budget** a natural measure of a campaign's success (Zhang et al., 2025; Didi et al., 2026b; Lin et al., 2026).

What determines how the number of unique hits scales with compute? Hit rates–the fraction of samples exceeding a *reward* threshold–capture only part of the answer. Recent reward-tilting methods steer generation toward higher predicted reward (Didi et al., 2026a;b; Hartman et al., 2025), often increasing hit rate, yet still underperforming naive sampling in *unique* hit yield across many targets (Didi et al., 2026b). The reason is that improving hit rate does not account for whether that hit belongs to a new cluster, nor for the compute required to discover and evaluate it.

We instead view unique hit scaling as arising from two underlying processes. The first is *cluster discovery*: how the generator distributes its samples across clusters and how often it discovers novel

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

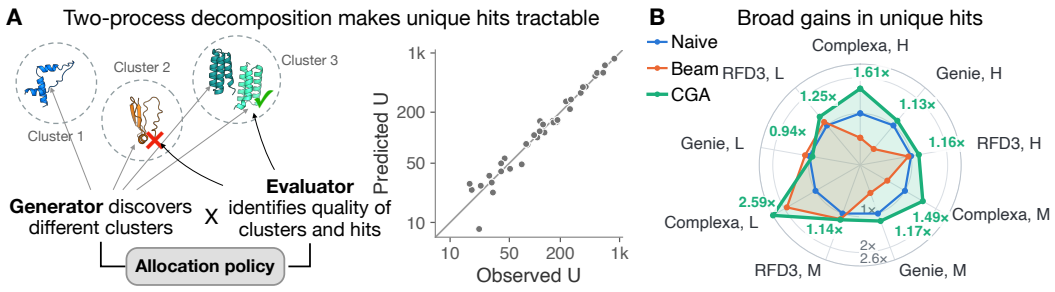

Figure 1: **(A)** Proposed two-process decomposition, with allocation policy, determine how unique hit U scales with compute. Each dot in the plot is a distinct generator-target pair. **(B)** CGA delivers broad gains in unique hits over naive across generators and target sets. The 19 targets are grouped by hit count into high(H)/medium(M)/low(L) sets, since totals can be dominated by high yield targets.

ones. The second is hit discovery within clusters, which is governed by *cluster quality*, the distribution of hit probabilities across clusters. Two processes determine unique hit scaling, making the yield of naive sampling even directly computable.

This cluster-level view (Fig 1A) also exposes a novel *allocation* axis that has been absent from prior work. Naive sampling and reward-tilting generate candidates and evaluate essentially all of them. Tracking cluster identities and qualities allows a campaign to skip evaluations with little expected value and redirect compute toward further generation. To demonstrate the gains achievable through allocation alone, we propose a **cluster-guided allocation (CGA)** policy that decides whether to generate or evaluate and, when evaluating, which cluster to prioritize, based on expected unique hit yield per unit compute. When both processes are known, CGA's expected yield can be estimated.

In practice, neither process is known. What statistical models explain these processes, and how much can CGA improve unique hit yield over naive sampling in real campaigns? We evaluate CGA and other sampling methods on the 19-target benchmark of Didi et al. (2026b) across three binder generators, with a budget of 24 H200 hours per campaign. We find that a Pitman-Yor model of cluster discovery (Pitman & Yor, 1997) and a zero-inflated Beta model of cluster quality together characterize campaign behavior well (Fig 1A, Secs. 4.2, 4.4). CGA estimates both processes online, and achieves **broad and large gains** over naive sampling across generators and targets, a pattern *not* observed with any reward-tilting methods (Didi et al., 2026b). On the medium set, CGA yields $1.49\times$, $1.17\times$, and $1.14\times$ as many unique hits as naive sampling with Complexa, Genie 3, and RFDiffusion3 (RFD3), respectively, compared with $0.61\times$, $0.58\times$, and $1.12\times$ for beam search (Fig. 1B).

Contributions. We develop a cluster-level framework that connects structural cluster discovery, hit discovery, and computational cost to unique hit yield, and formulates binder design as a compute allocation problem. Our statistical instantiation enables campaign forecasting and interpretable comparisons of generator-target behavior. Our policy instantiation, cluster-guided allocation (CGA), improves unique hit yield across 19 targets and three generators.

2 Preliminary

Protein structure diffusion and flow models for binder design. Our framework applies broadly to protein structure generators (Ahern et al., 2026; Stark et al., 2025; Chu et al., 2024), while focusing on relatively fast diffusion and flow models that learn the conditional distribution of binder structures given a target (Didi et al., 2026b; Butcher et al., 2025; Lin et al., 2026). These models are often trained on large datasets of experimental and predicted structures (Varadi et al., 2024), spanning diverse protein folds.

In silico evaluation of binders. In silico binder design studies commonly use structure prediction models as computational filters (Zambaldi et al., 2024; Watson et al., 2023). Sequences generated jointly with binder structures, or designed separately to match generated backbones, are refolded in complex with the target. Candidates are assessed using prediction confidence and structural con-

sistency between the designed and refolded complexes. Those satisfying the filtering criteria are designated *in silico hits*. Hits are usually clustered (Ahern et al., 2026; Sappington et al., 2026) using structural similarity measures such as TM-score, to preserve diverse designs that may satisfy downstream requirements not known in advance (Zambaldi et al., 2024). Clusters also better represent the distinct structural solutions proposed by a generator, reflecting its ability to explore different regions of protein fold space.

Reward-tilted generation. Reward-tilting aims to reweight a generator's sampled distribution to favor higher-reward candidates. Beam search maintains multiple candidate trajectories and uses lookahead estimates of final structures and their rewards to prioritize promising trajectories. Feynman–Kac steering targets the tilted distribution through particle weighting and resampling (Singhal et al., 2025). In protein design, rewards typically come from structure prediction feedback (Didi et al., 2026b). Thus, computing reward often costs more than generation, limiting feedback to be sparse and keeping lookahead samples in the candidate pool. However, even exact sampling from the tilted distribution does not account for structural redundancy. Consequently, higher hit rates need not translate into more unique hits under a fixed compute budget.

3 Scaling Binder Design

3.1 Two processes govern unique hit scaling

We propose to model a binder design campaign at the level of structural clusters, assigning candidates to clusters as they get generated; clustering need not be deferred until the end of a campaign. The objective counting at most one unique hit per cluster leads to our decomposition of unique hits into two processes: how many distinct structural clusters the generator discovers for a target, and how likely those clusters are to yield a hit.

(i) Cluster discovery. At any point, n candidates can be partitioned into k structural clusters $W_1, \ldots, W_k$. The cluster discovery process governs both the number and sizes $|W_i|$ of these clusters, describing how generation revisits observed clusters and discovers new ones.

(ii) Cluster quality and hit discovery. Structurally similar candidates may share binding modes and failure modes, motivating a cluster-level model of binder quality. Individual outcomes can vary due to differences among candidates, such as their interface residues, and stochasticity in structure prediction. We summarize these sources of variation by a latent hit probability P_i for each cluster and model candidate outcomes within that cluster as conditionally i.i.d. $\mathrm{Bernoulli}(P_i)$. This approximation treats candidates as independent draws from a fixed within-cluster distribution, with P_i representing their average probability of success.

We assume cluster hit probabilities P_i are independently drawn from a distribution G specific to the generator–target pair. For example, a target with only one viable binding mode accessible to the generator may induce a distribution G with most of its mass near zero and a small fraction at high hit probabilities. If the generator effectively accesses many viable binder folds, a larger fraction of clusters may have high hit probabilities.

We use binary hit instead of modeling reward distribution, since the confidence and consistency metrics are often uncalibrated, and our objective depends only on whether a candidate is a hit. Incorporating the full continuous metrics is a natural extension.

3.2 Campaign as an Allocation Problem

Defining quality at the cluster level changes how we view the binder design scaling. The campaign need not evaluate every sample it generates. A candidate in a cluster that has already produced a hit cannot add another unique hit, and for an unhit cluster, previous failures provide evidence that further evaluations in that cluster may be less promising. Meanwhile, generating another candidate can discover a new cluster or replenish an existing one worth evaluating.

This naturally leads us to view a binder design campaign as a sequential compute allocation problem. Each cluster has a queue of unevaluated candidates and a record of evaluation outcomes, and the pol-

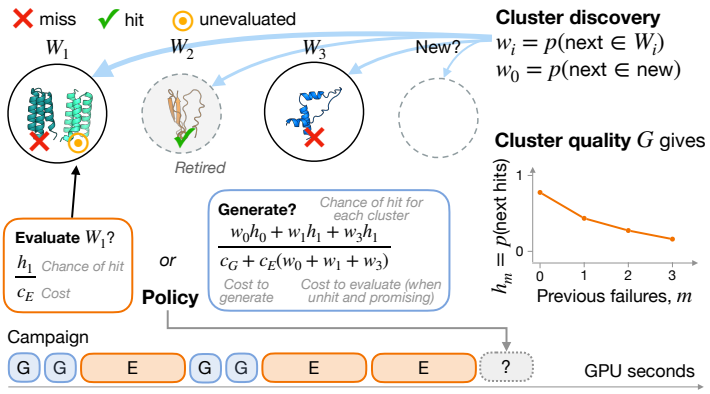

Figure 2: **Cluster-guided allocation (CGA).** Cluster discovery probabilities w_0, w_i and hit probabilities h_m conditioned on previous failures guide the choice between generation (G) and evaluation (E), based on expected unique hit yield per unit compute. Clusters are retired after their first hit, and evaluation prioritizes unhit clusters with available candidates and the fewest failures.

icy chooses between two actions: **Generate**, which samples a candidate at cost c_G, and **Evaluate**(i), which evaluates an available candidate from W_i at cost c_E. Evaluation is feasible only when the selected cluster contains an unevaluated candidate. An evaluation receives a reward $R_t = 1$ exactly when it produces its cluster's first hit; all other actions receive zero reward. The objective is

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_t R_t\right] \quad \text{subject to} \quad \sum_t c(A_t) \le B,$$

where the sums run over executed actions and $c(A_t)$ is c_G or c_E. A better allocation policy can improve unique hit yield while leaving the generator unchanged, by using the two processes to direct compute toward opportunities that can still expand the library.

3.3 Cluster-Guided Allocation (CGA)

To design an allocation policy, we take inspiration from multi-armed bandits, where a decision maker repeatedly chooses among arms, which gives a reward sampled from a different distribution that is not known to them. Here, evaluating a cluster corresponds to selecting an arm, while generation discovers new arms or supplies additional candidates to existing ones. An *index policy* simplifies these decisions by assigning each option an *index*, a score that accounts for future rewards and learning. Gittins theorem states that choosing an arm with maximum index is optimal when arms evolve independently and remain unchanged while unselected (Gittins, 1979; Weber, 1992).

These index policies motivate (See App. B.1 for the scope of optimality in our policy) our approach: we derive tractable indices for generation and evaluation to guide compute allocation. We first describe the policy's three components: *retire-on-hit*, an evaluation index that yields *least-failures-first* scheduling, and a generation index that balances further generation against evaluation.

(i) Retire-on-hit. Each cluster contributes at most one unique hit, so evaluating further candidates after its first hit provides no additional reward. We therefore retire that cluster from evaluation. For known G, independent cluster qualities, and fixed cluster identities, retirement is globally optimal (Proposition 1).

(ii) Least-failures-first. For clusters that have not yet produced a hit, the relevant quantity is the probability that the next evaluation succeeds. Each cluster's hit probability P_i is unknown, but its previous evaluation outcomes inform our belief about it. Under the shared quality distribution G, let

$$h_m = \Pr(\text{next evaluation is a hit} \mid m \text{ previous failures}, G) = \mathbb{E}[P_i \mid m \text{ previous failures}, G]\,.$$

Repeated failures provide evidence of lower cluster quality, so h_m is nonincreasing in m for fixed G. For each unhit cluster W_i with at least one unevaluated candidate, assign the evaluation index

$$I_E(i) = \frac{h_{m_i}}{c_E}, \tag{1}$$

where m_i is the number of its failed candidates. This score is the expected immediate unique hit reward per unit compute. Since evaluation costs are equal, prioritizing clusters with the fewest

Algorithm 1 Batched CGA

Require: Compute budget B; costs c_G, c_E; model priors; batch executors
Initialize structure records $\mathcal{D}$
while compute remains **do**
  Select a batch $\mathcal{E}$ of eligible clusters with the fewest failures
  Compute I_E from mean index (Eq. 1) of $\mathcal{E}$
  Compute I_G from Eq. 2
  if $I_G > I_E$ **then**
    Generate a batch, assign clusters, and append its structures to $\mathcal{D}$.
  else
    Evaluate one candidate each in $\mathcal{E}$ in parallel and record all outcomes in $\mathcal{D}$.
  Subtract the charged compute from B
  Fit cluster and quality posteriors and compute predictive w_0, w_i, h_m
Cluster all hits and return the number of unique hits

failures maximizes this score. The *least-failures-first* ordering also has an exact justification: for a fixed finite cluster members, known G, one worker, and no further generations, it maximizes infinite-horizon discounted unique hit reward (Theorem 1). This result respects our setting, where a cluster can be evaluated only while it has an unevaluated candidate.

(iii) Balance generation and evaluation. Generation produces no immediate reward, but creates opportunities for subsequent evaluation. Its value depends both on where the generated candidate lands and on whether evaluating that candidate would be worthwhile. Inspired by the Gittins index, which values an arm by considering future rewards and optimizing when to stop, we use a short lookahead for generation arm: generate one candidate, observe its cluster, and then decide whether to evaluate it.

If we evaluate novel candidates and candidates belonging to a selected set S of unhit existing clusters, the expected unique hit reward is $w_0 h_0 + \sum_{i \in S} w_i h_{m_i}$. The expected compute includes generation and the evaluation cost weighted by the probability of accepting the candidate. We assign

$$I_G = \max_{S \subseteq i: W_i \text{ is unhit}} \frac{w_0 h_0 + \sum_{i \in S} w_i h_{m_i}}{c_G + c_E\big(w_0 + \sum_{i \in S} w_i\big)}. \tag{2}$$

The maximization accounts for the option to leave candidates from low quality clusters unevaluated. Although we cannot direct the generator to sample from a specific cluster W_i, we can choose which generated candidates to evaluate, saving the substantial compute cost of evaluating less promising candidates.

Policy We choose generation when I_G exceeds the largest feasible evaluation index, and otherwise evaluate a cluster with the fewest failures, choosing evaluation on a tie. If no unhit cluster has an unevaluated candidate, we generate, provided the budget permits it. Figure 2 illustrates the policy components and how the two processes inform the generation and evaluation indices, using an example campaign state.

3.4 Modeling two processes

The indices require quantities computed from two processes that are not known and differ by generator-target combinations. We choose two common priors that suit each process and learn online, updating posteriors after each round of decision.

We model the unknown distribution over clusters using a Pitman-Yor process (Pitman & Yor, 1997). It accommodates unequal cluster frequencies and continued discovery without fixing the number of clusters in advance. Its power-law behavior also motivates its use for the approximately power-law cluster size distributions observed in our campaigns (App. Fig. 9). Its discount or tail weight $0 \leq \sigma < 1$ controls the persistence of cluster discovery as generation continues, and concentration $\theta > -\sigma$ controls the tendency to place samples in existing clusters. Given the current partition, the

next candidate forms a new cluster or joins an existing cluster W_i with probabilities

$$w_0 = \frac{\theta + \sigma k}{n + \theta}, \qquad w_i = \frac{|W_i| - \sigma}{n + \theta}, \quad i = 1, \ldots, k, \tag{3}$$

respectively. We model the quality distribution G as a zero-inflated Beta distribution, $P_i \sim G = \pi_0 \delta_0 + (1 - \pi_0)\,\mathrm{Beta}(a, b)$, where π_0 represents the fraction of clusters with zero hit probability, and the Beta component captures varying hit probabilities among the remaining clusters. We choose the Beta distribution for its flexible two-parameter form on $[0, 1]$ and tractable expressions for downstream quantities.

These models provide a **compact statistical description of the campaign-level behavior of complex generative and structure prediction pipelines**. Once fitted, the processes and action costs make expected unique hit yield tractable without running the underlying deep models: naive yield can be computed analytically, while CGA or any allocation policy yield can be estimated by simulating the two processes (App. B.5). Sec. 4.2 and App. D.5 evaluate the adequacy of these choices.

3.5 Practical implementation

To use parallel hardware, we execute the selected action in a batch. Algorithm 1 summarizes the procedure, using decision rules and indices defined in the previous section. Eligible clusters denote unhit clusters with an unevaluated sample. Both discovery and quality fits use standard likelihood-tempered sequential Monte Carlo (SMC) (Chopin, 2002; Del Moral et al., 2006) with 2,000 particles. The policy uses posterior-averaged predictive probabilities. See App. A.5 for other details.

4 Predicting and Improving Binder Design Campaigns

We first test whether the fitted processes predict future outcomes under naive sampling and CGA, then evaluate the gains from using them to allocate compute.

4.1 Experimental Settings

We evaluate on the 19-target binder design benchmark reported by Didi et al. (2026b).[1] We use three state-of-the-art generative binder design models: Proteina-Complexa (Didi et al., 2026b), Genie 3 (Lin et al., 2026), and RFDiffusion3 (Butcher et al., 2025). We compare three sampling policies: naive sampling, beam search–the best-performing reward-tilting method overall in Didi et al. (2026b)–and CGA. We additionally evaluate Feynman-Kac steering with Genie 3. We run a campaign for 24 H200 hours for each generator-target-policy combination. This corresponds to 1,000-5,000 generation and evaluations per target, depending on the costs c_G and c_E, which vary with protein length and the policy. All methods follow Genie 3's *in silico* evaluation pipeline, using AlphaFold-Multimer for structure prediction and the AlphaProteo hit criteria. Additional details on the baselines and evaluation pipeline are provided in the Appendix.

Compute accounting and clustering. All experiments use H200 nodes. We precompute c_G and c_E (App. A.5). Observed runtimes closely track these cost predictions apart from startup overhead; the complete cost table is provided in the Appendix. To keep clustering overhead negligible, we compare newly generated structures with cluster representatives rather than repeatedly computing all pairwise TM scores. App. D.1 shows that this approximation has little effect on clustering outcome.

4.2 Early Fits from CGA Predict Campaign Outcomes

The processes characterize scaling for each generator-target pair, and guide the policy's allocation decisions. We therefore test whether their predictions generalize beyond the fitting horizon and across sampling policies. Using posteriors fitted after 8 GPU-hours of CGA, we predict (App. B.5) naive sampling and CGA outcomes at 22 GPU-hours. Fig. 3 shows these predictions closely track the observed numbers of unique hits (U) and discovered clusters (K). The comparison is restricted

[1] The original benchmark contains 22 targets. Didi et al. (2026b) excludes three targets on which all methods fail and reports results for the remaining 19.

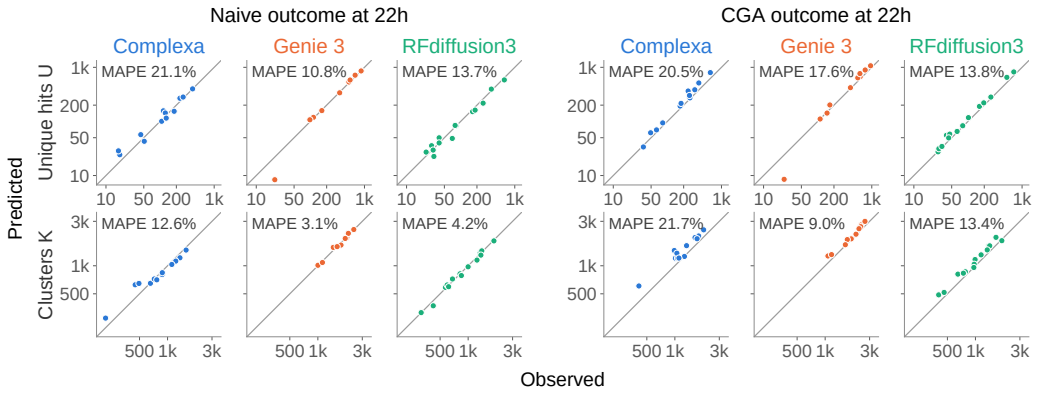

Figure 3: Posteriors fitted after 8 H200 hours of CGA predict unique hits (top) and discovered clusters (bottom) at 22 hours under naive (left) and CGA (right). Each point represents a target; diagonal lines indicate perfect prediction. MAPE denotes mean absolute percentage error.

to generator-target pairs with at least 20 observed hits, giving us 12 targets for Complexa, 10 for Genie 3, and 14 for RFDiffusion3.[2]

Table 1: MAPE (%; lower is better) of 22-hour naive U/K forecast from 8-hour prefixes. PY denotes Pitman-Yor. Emp. p denotes a single hit probability fitted and shared across all clusters.

Estimator	Complexa	Genie 3	RFD3
Linear fit	35.6 / 27.3	20.6 / 27.6	25.8 / 35.4
PY + emp. p	21.4 / 12.6	17.1 / 3.1	15.8 / 4.2
Ours	**21.1 / 12.6**	**10.8 / 3.1**	**13.7 / 4.2**

Although our goal is to summarize complex pipelines with simpler processes, these must retain enough flexibility to capture relevant dynamics. For cluster counts K, PY achieves 3.1-12.6% MAPE across generators, compared with 27.3-35.4% for linear fits, supporting the modeling of changing discovery rates as generation continues. Likewise, modeling heterogeneous cluster qualities improves MAPE of U over using a single hit probability shared by all clusters (Table 1, details in App. D.5).

These predictions also have two implications. First, they suggest that CGA can improve unique hit yield by more than 10% over naive sampling for 12/12 Complexa, 8/10 Genie 3, and 13/14 RFDiffusion3 targets, with the largest expected gain in Complexa (Sec. 4.4). Second, under Pitman-Yor cluster discovery, w_0 decays as a power law while new clusters continue to appear, i.e., the expected cluster count $\asymp n^{\sigma}$.

4.3 Scaling allocation axis with CGA

Figure 4A shows how unique hit counts, summed across targets, scale with compute under naive sampling, reward-tilting, and CGA. The curves show no clear saturation through 22 H200 hours, with approximately linear growth for Complexa and Genie 3 and a widening gain for CGA with RFDiffusion3. Figure 1B further breaks down aggregate gains into low-, medium-, and high-yield target sets.[3]

For Complexa, CGA substantially improves over naive sampling across all three sets, yielding $2.59\times$, $1.49\times$, and $1.61\times$ as many unique hits on the low, medium, and high sets, respectively, at 22 H200 hours. In contrast, beam search degrades and yields only around half as many unique hits on the medium and high sets. App. Fig. 8 shows that beam search produces more total hits but fewer hit clusters, indicating that reward-tilting concentrates on redundant folds rather than discovering diverse high reward candidates.

[2]This threshold is to support reliable posterior estimates; it does not imply that CGA requires a minimum number of hits to be effective. App. Fig. 7 shows that CGA generally outperforms naive sampling even in regimes with fewer than 20 hits.

[3]Targets are ranked by their maximum unique hit count across the three generators and divided into low-, medium-, and high-yield sets containing 7, 7, and 5 targets, respectively.

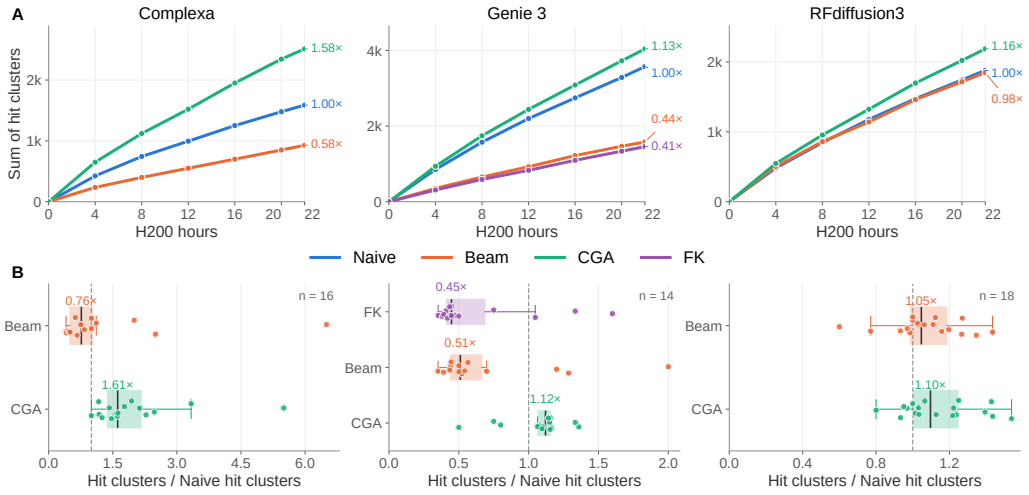

Figure 4: **(A)** Total *in silico* hit clusters versus charged H200 hours. Each run has a 24-hour budget, with curves shown through 22 hours. Annotations are ratios relative to the naive. See App. Fig. 7 for per target curves. **(B)** Unique hit clusters of sampling methods relative to naive. Each point represents a target, and annotations are the median. Targets with naive unique hits ≤ 1 are excluded.

For Genie 3, CGA yields $0.94\times$, $1.17\times$, and $1.13\times$ as many unique hits on the respective sets. Both tilting methods substantially underperform naive sampling in aggregate, with Feynman-Kac steering performing slightly worse than beam search. Only three Genie 3 targets–CbAgo, VEGFA, and Betv1–benefit from at least one tilting method (App. Fig. 7). RFDiffusion3 responds least to reward-tilting and beam search. Nevertheless, CGA yields $1.25\times$, $1.14\times$, and $1.16\times$ as many unique hits on the low, medium, and high sets, respectively.

Aggregate yields can be dominated by a few targets. The boxplots in Fig. 4B show per-target ratios of unique hit counts to naive sampling, revealing generally larger and more consistent gains for CGA than for beam search. Although CGA's aggregate gains are more modest for Genie 3 and RFDiffusion3 than for Complexa, individual targets show substantial improvements. For example, with RFDiffusion3, CGA yields $1.44\times$, $1.54\times$, $1.43\times$, and $1.39\times$ as many unique hits as naive sampling for BHRF1, Insulin, IL7RA, and Claudin1, respectively. Appendix Fig. 7 provides per-target scaling curves for all generators.

Cluster-aware allocation improves yield, with further gains from CGA. App. Fig. 11 shows on four Complexa targets, retiring hit clusters under the fixed decision schedule increases aggregate unique hits from 411 to 558, with CGA further increasing yield to 671. These results demonstrate the value of cluster-aware allocation, while supporting an additional benefit of the full policy.

4.4 Generators Differ in Both Discovery and Quality

Complexa and RFDiffusion3 appear similar in their numbers of discovered clusters and unique hits (Fig. 5A), but the fitted processes reveal distinct behavior. The generators occupy different regions of the Pitman-Yor parameter space, characterized by concentration θ and tail weight σ (Fig. 5B). The PD-L1 cluster-size profiles illustrate these differences: Complexa concentrates samples in a few dominant clusters, RFDiffusion3 is more even, and Genie 3 is more diffuse. Across the generators, the size of the largest cluster shrinks by an order of magnitude while the number of clusters increases from 149 to 399 to 616.

The quality process further distinguishes the generators. Fresh cluster hit probabilities h_0 (as well as total hits, App. Fig. 8) vary substantially across targets, and generator rankings can reverse (Fig. 5C). The half-life of h_m, defined as the number of failures needed to reduce h_m to $h_0/2$, also differs substantially (Fig. 5D). RFDiffusion3 generally exhibits longer half-lives, meaning that failures provide weaker evidence for deprioritizing a cluster. These differences help explain variation in CGA's gains, which depend jointly on both processes: repetitive generation creates opportunities

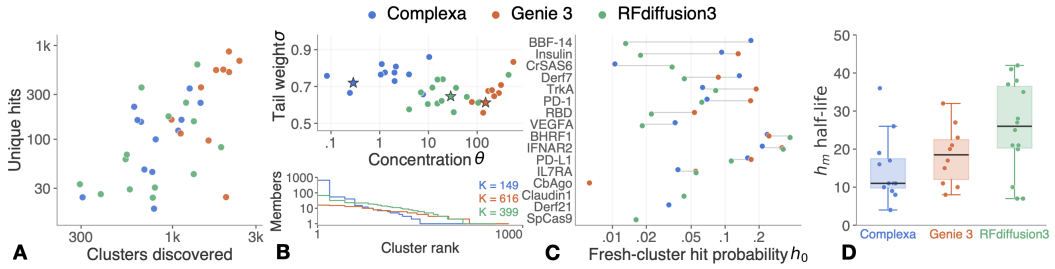

Figure 5: **(A)** Discovered clusters and unique hits. **(B)** Fitted Pitman-Yor parameters (top; stars mark PD-L1) and PD-L1 cluster size profiles (bottom). **(C)** Fresh-cluster hit probabilities h_0 across targets. **(D)** Half-life of h_m: the number of failures required to halve the initial hit probability. C-D is restricted to generator-target pairs with at least 20 observed hits.

for retire-on-hit, access to many clusters makes prioritization through least-failures-first valuable, and shorter half-lives make failures more informative for identifying unpromising clusters.

5 Conclusion

In this paper, we presented a cluster-level framework for understanding and improving unique hit scaling in protein binder design. Decomposing campaigns into cluster discovery through generation and hit discovery through evaluation enables prediction of campaign yield and guides compute allocation. Our proposed statistical models explain the campaign well, and our cluster-guided allocation (CGA) policy shows general improvements over naive sampling and the tested reward-tilting methods at matched compute.

Our framework has several limitations that also motivate extensions. We use binary hit outcomes without exploiting continuous evaluation metrics, CGA has no general optimality guarantee in the practical setting (App. B.1), and combining better allocation with reward tilting remains untested.

Our decomposition helps organize these extensions through a common map of how algorithms change campaign yield (Fig. 6). Beam search often increases U/K at the expense of K, with a net loss in yield, while CGA increases K by redirecting evaluation compute toward generation (Fig. 6A). More importantly, this map reveals further opportunities (Fig. 6B): within-cluster reward tilting and evaluation guided by continuous metrics could raise U/K, while diversity-guided generation could increase K. Their benefits depend on the resulting balance and compute costs, making explicit three axes for scaling binder design: cluster exploration, effective hit discovery, and adaptive compute allocation.

As more generators become available and both generators and structure predictors grow in size, understanding how their capabilities and computational costs interact becomes increasingly important. Our work provides a basis for comparing these evolving pipelines through a lens of simpler statistical models and compute allocation problem.

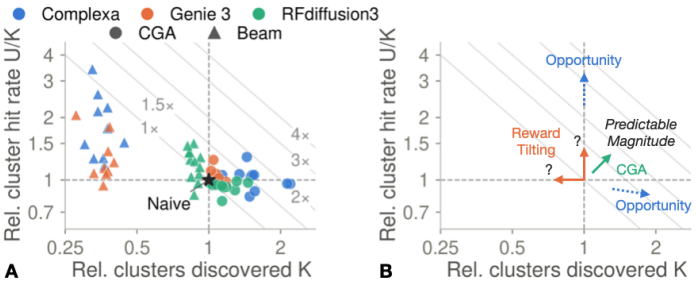

Figure 6: **(A)** CGA and beam search's cluster count K and cluster hit rate U/K relative to naive sampling at matched compute. **(B) Conceptual binder design scaling map.** Both panels use logarithmic axes; diagonals mark constant unique hit gains.

AI USE STATEMENT

In this work, we used generative AI tools to assist in writing proofs and in creating and editing software code. We did not use generative AI tools to generate synthetic datasets, formulate the conceptual framework, or summarize or analyze existing literature. We have reviewed all AI-assisted work. Multiple authors verified the theoretical claims and proofs. Multiple authors verified the AI-generated and AI-assisted software. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Woody Ahern, Jason Yim, Doug Tischer, Saman Salike, Seth M Woodbury, Donghyo Kim, Indrek Kalvet, Yakov Kipnis, Brian Coventry, Han Raut Altae-Tran, et al. Atom-level enzyme active site scaffolding using rfdiffusion2. *Nature Methods*, 23(1):96–105, 2026.

Nathaniel R Bennett, Brian Coventry, Inna Goreshnik, Buwei Huang, Aza Allen, Dionne Vafeados, Ying Po Peng, Justas Dauparas, Minkyung Baek, Lance Stewart, et al. Improving de novo protein binder design with deep learning. *Nature Communications*, 14(1):2625, 2023.

Jasper Butcher, Rohith Krishna, Raktim Mitra, Rafael I Brent, Yanjing Li, Nathaniel Corley, Paul T Kim, Jonathan Funk, Simon Mathis, Saman Salike, et al. De novo design of all-atom biomolecular interactions with rfdiffusion3. *bioRxiv*, 2025.

Longxing Cao, Brian Coventry, Inna Goreshnik, Buwei Huang, William Sheffler, Joon Sung Park, Kevin M Jude, Iva Marković, Rameshwar U Kadam, Koen HG Verschueren, et al. Design of protein-binding proteins from the target structure alone. *Nature*, 605(7910):551–560, 2022.

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

Alexander E Chu, Jinho Kim, Lucy Cheng, Gina El Nesr, Minkai Xu, Richard W Shuai, and Po-Ssu Huang. An all-atom protein generative model. *Proceedings of the National Academy of Sciences*, 121(27):e2311500121, 2024.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

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

Kieran Didi, Danny Reidenbach, Matthew Penner, Supriya Ravi, Marshall Case, Mike Nichols, Erik Swanson, Alex Reis, Maggie Prescott, Yue Qian, Dongming Qian, Jingjing Yang, Weiji Li, Le Li, Daichi Shonai, Sean Gay, Bhoomika Basu Mallik, Ho Yeung Chim, Liurong Chen, Miguel Atienza Juantay, Hubert Klein, Anna Macintyre, Maxim Secor, Daniele Granata, Zhonglin Cao, Guoqing Zhou, Tomas Geffner, Xi Chen, Micha Livne, Zuobai Zhang, Tianjing Zhang, Michael M. Bronstein, Martin Steinegger, Kristine Deibler, Scott Soderling, Alena Khmelinskaia, Florian Hollfelder, Christian Dallago, Emine Kucukbenli, Arash Vahdat, Pierce Ogden, and Karsten Kreis. Latent generative search unlocks de novo design of untapped biomolecular interactions at scale. `https://research.nvidia.com/labs/genair/proteina-complexa/assets/proteina_complexa_validation.pdf`, 2026a.

Kieran Didi, Zuobai Zhang, Guoqing Zhou, Danny Reidenbach, Zhonglin Cao, Sooyoung Cha, Tomas Geffner, Christian Dallago, Jian Tang, Michael Bronstein, et al. Scaling atomistic protein binder design with generative pretraining and test-time compute. In *International Conference on Learning Representations*, volume 2026, pp. 61056–61109, 2026b.

Daniel Fershtman and Alessandro Pavan. *Searching for arms: Experimentation with endogenous consideration sets*. Foerder Institute for Economic Research, Tel Aviv University, The Eitan ..., 2021.

Tomas Geffner, Kieran Didi, Zuobai Zhang, Danny Reidenbach, Zhonglin Cao, Jason Yim, Mario Geiger, Christian Dallago, Emine Kucukbenli, Arash Vahdat, et al. Proteina: Scaling flow-based protein structure generative models. In *International Conference on Learning Representations*, volume 2025, pp. 98803–98851, 2025.

John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 41(2):148–164, 1979.

Erik Hartman, Jonas Wallin, Johan Malmström, and Jimmy Olsson. Controllable protein design with particle-based feynman-kac steering. *arXiv preprint arXiv:2511.09216*, 2025.

John B Ingraham, Max Baranov, Zak Costello, Karl W Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M Lord, Christopher Ng-Thow-Hing, Erik R Van Vlack, et al. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, 2023.

Yeqing Lin, Minji Lee, Aakarsh Vermani, Ellena Jiang, Sebastiaan De Cooman, Matej Špetko, and Mohammed AlQuraishi. Fast and ultra-capable protein design: Advancing the frontier through atomistic se (3)-equivariance with genie 3. *bioRxiv*, 2026.

Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: making protein folding accessible to all. *Nature methods*, 19(6):679–682, 2022.

Martin Pacesa, Lennart Nickel, Christian Schellhaas, Joseph Schmidt, Ekaterina Pyatova, Lucas Kissling, Patrick Barendse, Jagrity Choudhury, Srajan Kapoor, Ana Alcaraz-Serna, et al. One-shot design of functional protein binders with bindcraft. *Nature*, 646(8084):483–492, 2025.

Jim Pitman and Marc Yor. The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pp. 855–900, 1997.

Isaac Sappington, Martin Toul, David S Lee, Stephanie A Robinson, Inna Goreshnik, Clara McCurdy, Tung Ching Chan, Nic Buchholz, Buwei Huang, Dionne Vafeados, et al. Improved protein binder design using β-pairing targeted rfdiffusion. *Nature Communications*, 17(1):1101, 2026.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. *arXiv preprint arXiv:2501.06848*, 2025.

Hannes Stark, Felix Faltings, MinGyu Choi, Yuxin Xie, Eunsu Hur, Timothy O'Donnell, Anton Bushuiev, Talip Uçar, Saro Passaro, Weian Mao, et al. Boltzgen: Toward universal binder design. *BioRxiv*, 2025.

Michel Van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature biotechnology*, 42(2):243–246, 2024.

Mihaly Varadi, Damian Bertoni, Paulyna Magana, Urmila Paramval, Ivanna Pidruchna, Malarvizhi Radhakrishnan, Maxim Tsenkov, Sreenath Nair, Milot Mirdita, Jingi Yeo, et al. Alphafold protein structure database in 2024: providing structure coverage for over 214 million protein sequences. *Nucleic acids research*, 52(D1):D368–D375, 2024.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Richard Weber. On the gittins index for multiarmed bandits. *The Annals of Applied Probability*, pp. 1024–1033, 1992.

Gideon Weiss. Branching bandit processes. *Probability in the Engineering and Informational Sciences*, 2(3):269–278, 1988.

Peter Whittle. Arm-acquiring bandits. *The Annals of Probability*, 9(2):284–292, 1981.

Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023.

Vinicius Zambaldi, David La, Alexander E Chu, Harshnira Patani, Amy E Danson, Tristan OC Kwan, Thomas Frerix, Rosalia G Schneider, David Saxton, Ashok Thillaisundaram, et al. De novo design of high-affinity protein binders with alphaproteo. *arXiv preprint arXiv:2409.08022*, 2024.

Odin Zhang, Xujun Zhang, Haitao Lin, Cheng Tan, Qinghan Wang, Yuanle Mo, Qiantai Feng, Gang Du, Yuntao Yu, Zichang Jin, et al. Odesign: A world model for biomolecular interaction design. *arXiv preprint arXiv:2510.22304*, 2025.

Yang Zhang and Jeffrey Skolnick. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic acids research*, 33(7):2302–2309, 2005.

A Implementation details

A.1 Targets and Generative Models

We use the 19 target benchmark of Complexa (Didi et al., 2026b), with the target structures, crops, and hotspot residues specified in its benchmark. The excluded targets are H1, IL17A, and TNF-α, for which the original benchmark reported no successes within its 20-hour evaluation window. For all three generators, binder lengths are sampled uniformly from $\{70, 75, \ldots, 120\}$ residues. Restricting lengths to this grid limits AlphaFold-Multimer to 11 binder-length-dependent input shapes, reducing repeated JAX compilation.

Genie 3. We use the default binder design configuration with 100 sampling steps, DDIM stochasticity $\eta = 1$, and direction scale zero. Conditioning uses Genie 3's interface extension. ProteinMPNN (Dauparas et al., 2022) designs one sequence per generated backbone at temperature 0.1.

Proteina-Complexa. We use the released binder-generation configuration with 400 sampling steps and its default schedules and noise settings. Conditioning uses the benchmark's defined hotspot residues. The model directly decodes the binder sequence and atomistic structure.

RFDiffusion3. We use the default binder design configuration with 100 sampling steps and its default schedules and noise settings. Conditioning uses the benchmark's defined hotspot residues. ProteinMPNN designs one sequence per generated backbone at temperature 0.1.

A.2 Clustering

We cluster structures with Foldseek (Van Kempen et al., 2024), skipping the prefilter stage and computing structural similarity with exact TM-score via TM-align (Zhang & Skolnick, 2005).

A.3 Baseline Implementations

Naive sampling. We independently sample complete designs from each generator and evaluate every design, retaining all in silico hits. The generator settings are those described above.

Beam search. We follow Complexa's implementation, using beam width $W = 4$, branching factor $B = 4$, and four evenly spaced scoring checkpoints. At each checkpoint, all 16 candidates are rolled out to completion and evaluated; the four highest-scoring partial states continue. Following Complexa, every evaluated rollout is included in the sampling pool, giving 64 designs per complete search trajectory. Selection maximizes $r = -\overline{\mathrm{pAE}}_{\mathrm{interface}}/31$, where the bar denotes the mean over both binder–target off-diagonal blocks of the predicted aligned error matrix.

Feynman–Kac steering. We additionally evaluate FK steering with Genie 3, using the same branching, checkpoints, rollouts, and scoring function as beam search. Four partial states are resampled with replacement at each checkpoint, with probabilities

$$p_i = \frac{\exp(z_i/T)}{\sum_{j=1}^{16} \exp(z_j/T)}, \qquad z_i = \frac{r_i - \bar{r}}{s_r}, \qquad T = 0.1,$$

where $\bar{r}$ and s_r are the checkpoint's reward mean and population standard deviation. Equal rewards give uniform resampling. This implementation standardizes rewards before applying the temperature.

Choice of reward-tilting baselines We chose 2 most performant methods from Complexa paper. A further baseline across all three generators would require a significant compute ($19 \times 3 \times 24 = 1368$ H200 hours).

A.4 Structure Prediction and Hit Definition

All methods use the same ColabFold (Mirdita et al., 2022) implementation of AlphaFold-Multimer, following Genie 3's adaptation of the AlphaProteo evaluation protocol (Lin et al., 2026). We

Table 2: Action costs in H200 GPU-seconds per design.

Target	c_G, Complexa	c_G, Genie 3	c_G, RFDiffusion3	c_E
BBF-14	2.50	2.85	5.10	15.15
Betv1	3.07	4.19	5.73	23.12
BHRF1	2.99	6.27	5.73	22.54
CbAgo (PAZ)	2.43	3.05	4.74	15.54
CD45	2.33	2.67	4.77	14.46
Claudin1	3.50	6.33	6.25	56.47
CrSAS6	2.87	3.35	5.56	26.23
Derf21	2.56	3.40	5.21	18.60
Derf7	3.56	5.00	6.43	25.71
HER2-AAV	4.21	6.18	6.90	41.50
IFNAR2	2.39	4.00	5.04	16.50
IL-7Rα	3.51	5.70	6.21	24.49
Insulin	2.95	4.72	5.68	13.53
PD-1	2.49	3.88	5.13	15.11
PD-L1	2.56	3.62	5.21	11.44
SARS-CoV-2 RBD	3.54	6.26	6.15	36.20
SpCas9	3.91	5.81	6.85	34.93
TrkA	2.35	3.59	4.82	11.31
VEGFA	3.46	5.36	6.31	30.57

use all five `alphafold2_multimer_v3` models, one seed, single-sequence input, and the target structure as a template. Recycling is capped at 20, with the model's default convergence tolerance of 0.5 Å. Dropout and relaxation are disabled. We select the prediction with the highest $0.8\,\mathrm{ipTM} + 0.2\,\mathrm{pTM}$ score and compute the following criteria on that prediction.

A design is an *in silico* hit if all of the following hold: `pTM_binder`> 0.8, `RMSD_complex`$<$ 2.5Å, `min pAE_interface`< 1.5Å, and hotspot coverage is at least 1 for ($1 \leq H \leq 3$) and at least 0.8 for ($H \geq 4$). where H is the number of specified hotspot residues. A hotspot is covered when at least one of its heavy atoms is within 5 Å of a binder heavy atom in the predicted complex. BBF-14 and CrSAS6 have no specified hotspots and automatically satisfy the coverage condition. Coverage is checked against the specified hotspots, rather than the expanded set used to condition Genie 3.

A.5 Hardware, Action Costs, and Budget Accounting

All experiments use NVIDIA H200 GPUs. Generation uses batch size four per GPU, and each evaluation worker processes one design at a time. Naive sampling and our policy use eight GPUs per run; beam search and FK use four, so their 16 candidates also give a generation batch size of four per GPU. Each target–generator–method run receives a budget of $86{,}400$ charged GPU-seconds i.e., 24 GPU-hours.

Table 2 gives the fixed rates used by and charged at all methods. The generation rates c_G were measured at binder length 95, batch size of 4, with three warmup rounds followed by five measured rounds. Genie 3 and RFDiffusion3 c_G includes ProteinMPNN inference time. The evaluation rates c_E were measured at binder lengths $70, 80, 90, 100, 110, 120$, with two warmup rounds and three measured rounds of four sequential evaluations at each length. The reported c_E is the mean over these 18 measured rounds (72 designs per target). The evaluator process remains warm across the length sweep. Rates are execution time divided by the number of designs, not batches.

A partial trajectory from sampling step a to b is charged $(b-a)c_G/S$, where S is the generator's total number of sampling steps. A complete beam or FK trajectory costs $40c_G + 64c_E$, because children share trajectory of parent node, giving slightly reduced cost per candidate. Runs stop when they cannot afford the next action or complete search trajectory, leaving a small unspent balance. We therefore report all performance metrics at 22 GPU-hours.

We report charged GPU-hours to separate compute allocation from implementation- and hardware-dependent runtime variation. Even for the same generator-target pair, runtime varies across runs and between four- and eight-GPU configurations. Since our primary focus is the allocation frame-

work rather than optimizing its implementation, we separately report the overhead of components introduced by CGA, including clustering (Table 4). Posterior fitting takes less than 0.1s with 2,000 particles thus adds negligible overhead. Also, both operations run on CPUs and can overlap with GPU operations.

B Analysis of cluster discovery bandits

B.1 Scope of Optimality

Extensions of classical bandits accommodate the discovery of new arms (Whittle, 1981; Weiss, 1988; Fershtman & Pavan, 2021), but our setting introduces additional dependency: Evaluation is feasible only when a cluster contains an unevaluated candidate, so generation changes the inventories and available actions of existing clusters. This falls outside the independent arm structure required for a direct application of the classical Gittins theorem, and its optimality guarantees therefore do not directly apply.

The practical setting introduces further departures through a finite compute budget, parallel batches, and estimating w_i and h_m. We therefore make no general optimality claim for the full policy, and focus on empirical results. The exact guarantees apply to its components under the stated assumptions: retire-on-hit is optimal for both the budgeted and discounted objectives under the known, independent-quality model with fixed cluster identities, while least-failures-first is optimal for discounted evaluation with fixed finite inventory, one worker, and no further generations (Proposition 1 and Theorem 1).

B.2 Assumptions and predictive hit probabilities

Cluster identities are fixed, and qualities P_i are independent draws from a known distribution G on $[0, 1]$. Conditional on these qualities, candidate outcomes are independent Bernoulli draws. Generation depends only on the population history, independently of qualities and evaluation outcomes. The policy observes cluster assignments and binary outcomes, with no additional quality information. Generation and evaluation cost $c_G, c_E > 0$; an evaluation consumes one available candidate. Write u_i for cluster i's remaining inventory. Each cluster rewards only its first hit. For the discounted objective, a reward obtained at cumulative compute T is multiplied by β^T, where $0 < \beta < 1$.

After m failures in an unhit cluster, its next-hit probability is

$$s_m = \mathbb{E}_G[(1-P)^m], \qquad h_m = \frac{\mathbb{E}_G[P(1-P)^m]}{s_m} = 1 - \frac{s_{m+1}}{s_m}. \tag{4}$$

We condition only on histories with $s_m > 0$.

Lemma 1 (Failure monotonicity)**.** *If $s_m > 0$ and $h_m < 1$, then*

$$h_m - h_{m+1} = \frac{\mathrm{Var}(P \mid m \textit{ failures})}{1 - h_m} \geq 0. \tag{5}$$

Proof. Writing $\mathbb{E}_m$ for expectation after m failures, Bayes' rule gives

$$h_{m+1} = \frac{\mathbb{E}_m[P(1-P)]}{\mathbb{E}_m[1-P]} = \frac{h_m - \mathbb{E}_m[P^2]}{1 - h_m}.$$

Subtracting from h_m gives the result. □

B.3 Retire-on-hit

Proposition 1 (Retire-on-hit)**.** *Under Appendix B.2, every feasible policy is weakly dominated by one that never evaluates a cluster after its first hit. This holds for both a fixed compute budget and infinite-horizon discounted reward, even when generation continues.*

Proof. A post-hit evaluation gives no reward and, because G is known and qualities are independent, supplies no information about other clusters or future arrivals. Replace each such evaluation by

private simulation: at retirement draw a quality from that cluster's posterior, then use it to generate all its omitted outcomes. Run the original policy on this simulated history, including its original virtual budget and clock. By charging compute only for actual actions, this preserves its decision distribution.

Couple all actual generations and unhit-cluster evaluations to the original policy. Their candidates remain available and their rewards occur at no greater cumulative compute. Thus every reward within the original budget remains feasible, and discounted rewards are no smaller. Retired clusters and their generated arrivals remain in the population counts. □

B.4 Least-failures-first with fixed inventory

Theorem 1 (Finite-inventory evaluation)**.** *Under Appendix B.2, fix a finite generated pool and allow no further generations. With equal evaluation costs and one worker, least-failures-first among populated, unhit clusters maximizes infinite-horizon discounted unique hit reward. Inventories may be unequal.*

Proof. By Proposition 1, successful clusters can be retired. Each cluster is now an independent rested project, with a successful or exhausted cluster represented by an absorbing zero-reward state. The Gittins theorem therefore applies (Gittins, 1979; Weber, 1992).

Set $\gamma = \beta^{c_E}$. Using the standard discounted-time normalization, the index of an unhit cluster in state (m, u), $u \geq 1$, is

$$J(m,u) = \sup_{\tau} \frac{\mathbb{E}[\sum_{r=1}^{\tau} \gamma^{r-1} R_r]}{\mathbb{E}[\sum_{r=1}^{\tau} \gamma^{r-1}]}. \tag{6}$$

Completion-time discounting multiplies every policy's reward by the same factor γ, leaving its ordering unchanged. Here τ is a stopping time with $1 \leq \tau \leq u$, stopping no later than the first hit; these restrictions lose nothing because later rewards are zero. Let $a_r = \Pr(\tau \geq r)$. Reaching evaluation r requires $r-1$ failures, so the ratio for any such stopping rule is

$$\frac{\sum_{r=1}^{u} a_r \gamma^{r-1} h_{m+r-1}}{\sum_{r=1}^{u} a_r \gamma^{r-1}} \leq h_m. \tag{7}$$

Terms with $a_r = 0$ are omitted. The inequality follows from Lemma 1; taking one evaluation attains it. Hence

$$J(m,u) = h_m \qquad (u \geq 1). \tag{8}$$

The largest index is therefore attained by a cluster with the fewest failures, independently of its positive inventory size. Dividing by the common cost c_E gives the same ordering as I_E. □

B.5 Computation of naive and CGA unique hit yield

We predict fresh 22-hour campaigns using fixed posterior-mean parameters fitted to the last complete CGA round within eight charged hours

Naive sampling evaluates all generated candidates, so it can generate $N_0 = \lfloor B/(c_G + c_E) \rfloor$ candidates. Combining the Pitman–Yor cluster size prediction (Pitman & Yor, 1997) with the fitted hit-probability model, expected naive hit

$$\widehat{U}_0(B) = \sum_{j=1}^{N_0} \underbrace{\mathbb{E}_{\mathrm{PY}}[M_{N_0,j}]}_{\text{expected number of size-}j\text{ clusters}} \underbrace{(1-s_j)}_{\text{probability of at least one hit}} \tag{9}$$

$$= \sum_{j=1}^{N_0} \binom{N_0}{j} \frac{(1-\sigma)_{j-1}(\theta+\sigma)_{N_0-j}}{(\theta+1)_{N_0-1}} (1-s_j), \tag{10}$$

where $(x)_q$ is the rising factorial.

CGA yield is averaged over 256 independent simulations, with cluster assignments sampled from the fitted Pitman-Yor process and cluster qualities P_i drawn i.i.d. from the fitted G. Evaluations are simulated as $\mathrm{Bernoulli}(P_i)$ outcomes, and generation and evaluation incur costs c_G and c_E, respectively, without running the binder generator or structure predictor.

C EXPANDED RESULTS

C.1 SCALING CURVES

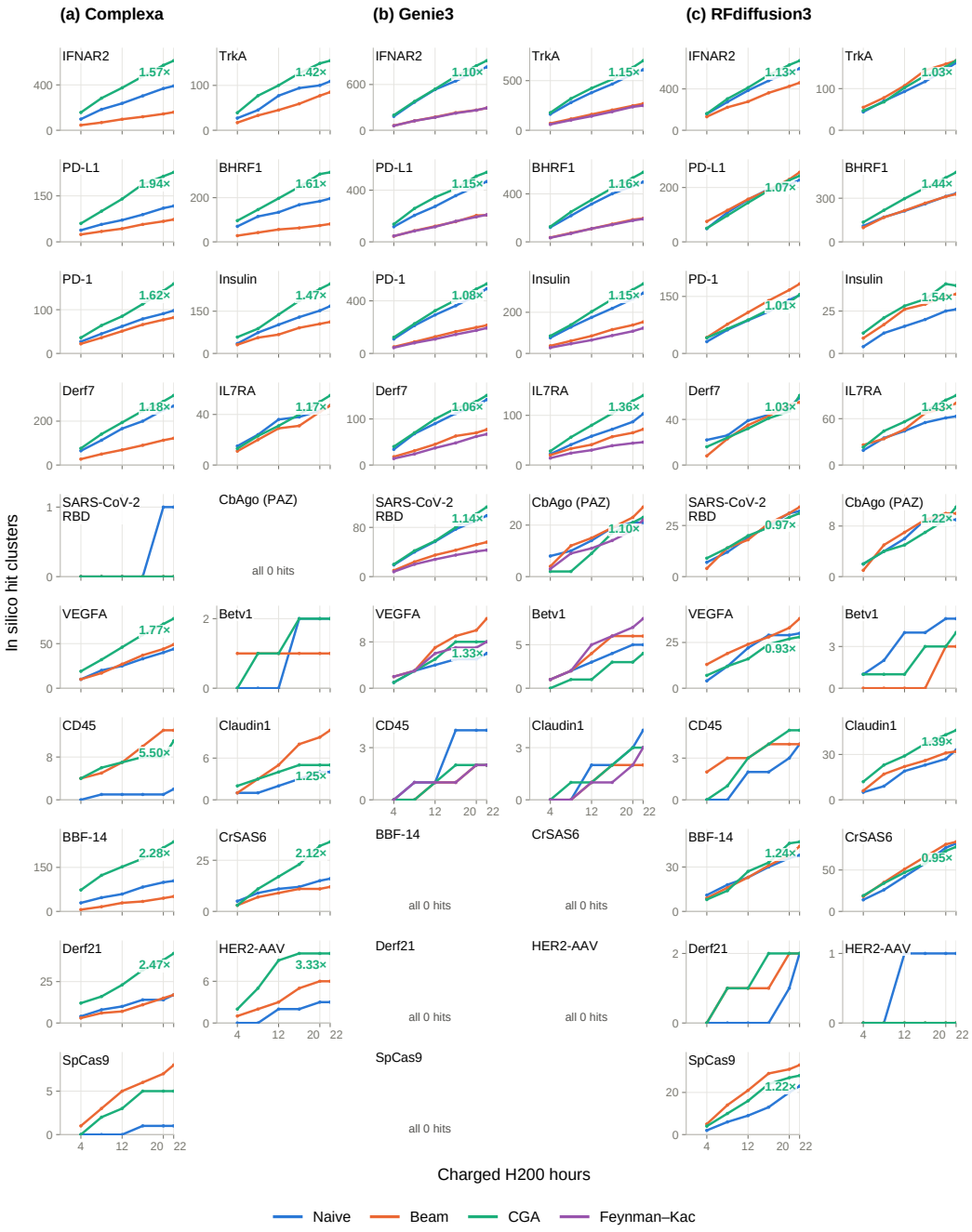

Figure 7: ***In silico* hit clusters versus charged H200 hours for (a) Complexa, (b) Genie 3, and (c) RFDiffusion3.** Each run uses a 24-hour budget; curves are shown up to 22 hours. The annotation shows CGA's hit cluster ratios relative to naive sampling. We omit annotation for generator-target pairs where naive sampling yields fewer than 5 hits, as ratios in this regime can be noisy and exaggerate apparent gains.

C.2 HIT AND HIT CLUSTERS AT TM=0.6

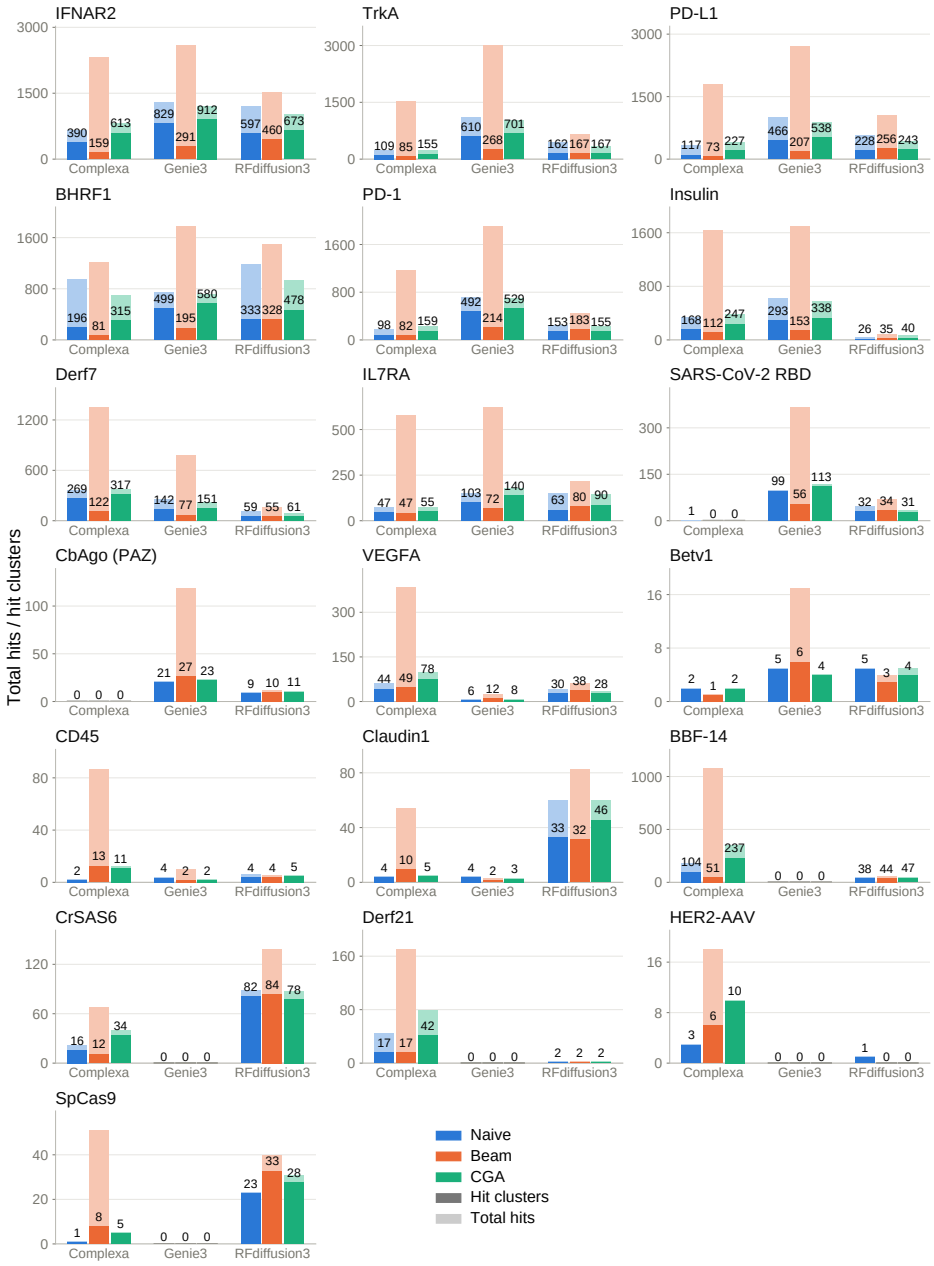

Figure 8: ***In silico* hit and hit clusters at 22 H200 hours**

CGA can produce more total hits than final hit clusters because retire-on-hit applies to the partition available at each decision. Our Foldseek workflow uses greedy set-cover clustering, whose representatives and memberships can change as the input set changes. Consequently, hits discovered in distinct clusters during the campaign may fall into the same cluster under final reclustering. This can occur even with full reclustering at every update and is therefore not specific to incremental clustering.

C.3 CONTINUED CLUSTER DISCOVERY

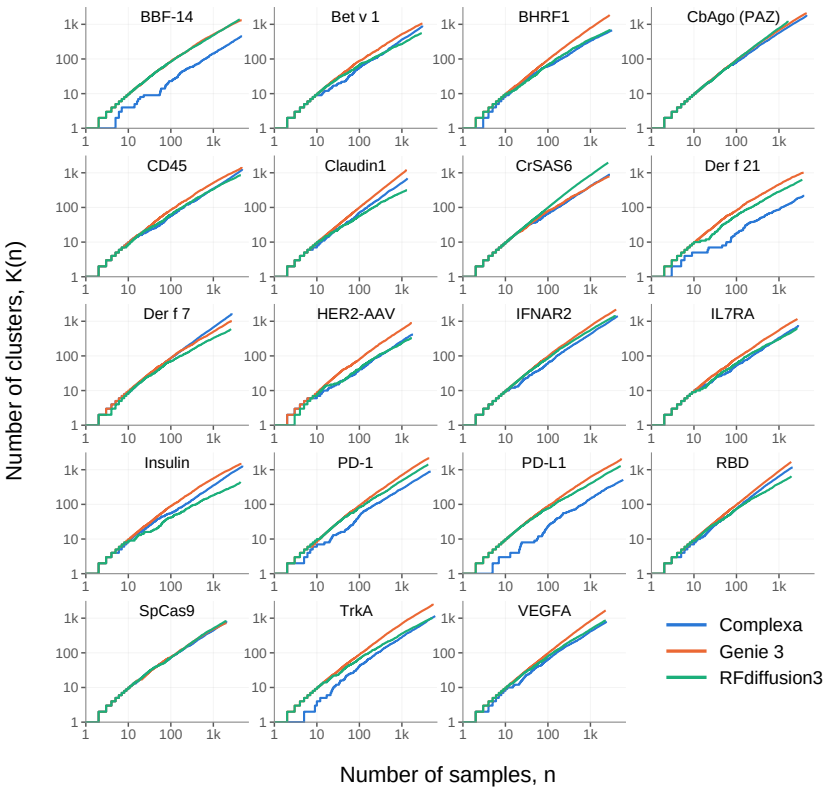

Figure 9: Structural cluster counts versus samples generated across 19 targets and three generators under naive sampling. Curves reuse final full clustering labels; both axes are logarithmic.

D Additional Results

D.1 Effect of incremental clustering

For 38 CGA runs (Genie 3 and Complexa), we compare the last online incremental partition with the full clustering partition recomputed. Table 3 shows median changes of 0.70% for Complexa and 0.63% for Genie 3; every run changes by less than 4.88%. These are counts of *all* clusters, including those without hits.

Table 3: Cluster count drift in the 38 policy runs. Cluster totals sum independent target partitions; relative changes are computed within each run.

	Total clusters		Absolute change (%)	
Generator	Online	Full	Median	Maximum
Complexa	27,020	26,726	0.70	4.72
Genie 3	33,018	32,669	0.63	4.87

Measured runtime. CPU clustering is outside the charged GPU budget but contributes to elapsed runtime. Table 4 shows speedup of incremental clustering keeps clustering overhead negligible and remains tractable as campaigns scale.

Table 4: Clustering wall time at fixed pool sizes. Measured once per size on Genie 3-TrkA with eight CPU threads. Incremental updates add 32 structures.

Structures	Full (s)	Incremental (s)	Speedup
500	25.57	3.89	6.6×
1,000	79.90	5.42	14.8×
2,000	282.91	5.70	49.7×
5,000	984.82	5.04	195.2×

D.2 Effect of clustering method and threshold

Even though CGA uses the observed cluster partition to guide allocation, its gains are not specific to the default TM-score cutoff of 0.6. Fig. 10 reclusters the hits at different TM-score cutoffs of 0.5 and 0.7, and shows that the broad gains of CGA persist.

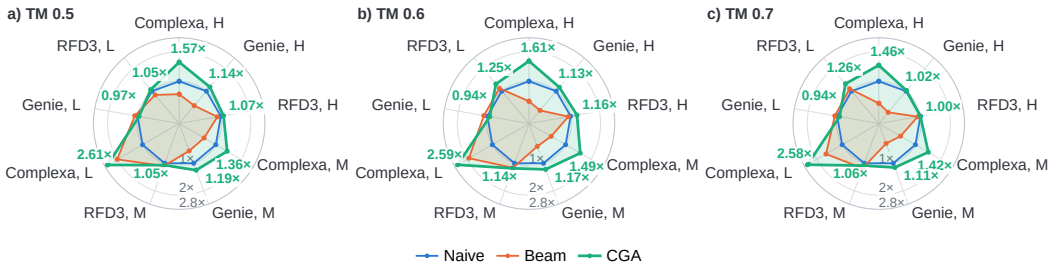

Figure 10: **Sensitivity to clustering method and threshold.** Unique hit counts relative to naive sampling under different TM-score cutoffs for clustering, across generators and low (L), medium (M), and high (H)-yield target sets. Green annotations indicate CGA fold gains.

D.3 Effect of adaptive allocation

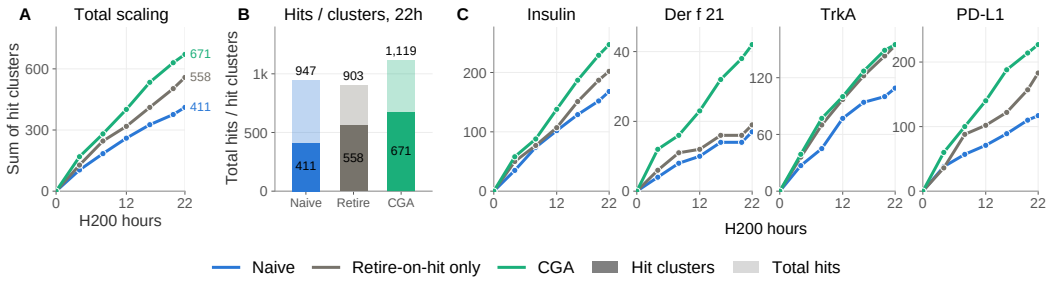

Figure 11: Naive sampling, retire-on-hit with a fixed generation-evaluation schedule, and CGA on four Complexa targets. Fixed schedule uses one generation batch (32 samples) followed by four evaluation batches (32 evaluations). **(A)** Aggregate hit clusters versus compute per target. **(B)** Total hits and hit clusters at 22 H200 hours. **(C)** Per-target scaling curves.

D.4 Replicates

Our strongest evidence for CGA is the breadth and consistency of its improvements across 19 targets, three generators, and multiple GPU-hour checkpoints. To quantify variability across random seeds, we additionally run CGA with three seeds on selected RFDiffusion3 targets.

The trajectories in Fig. 12 show low variability in final unique-hit yield: at 22 H200 hours, counts range from 76 to 78 for CrSAS6 and from 148 to 155 for PD-1. The primary campaign allocates 4,104 H200 GPU-hours across targets, generators, and methods, making replication of every setting across multiple seeds impractical.

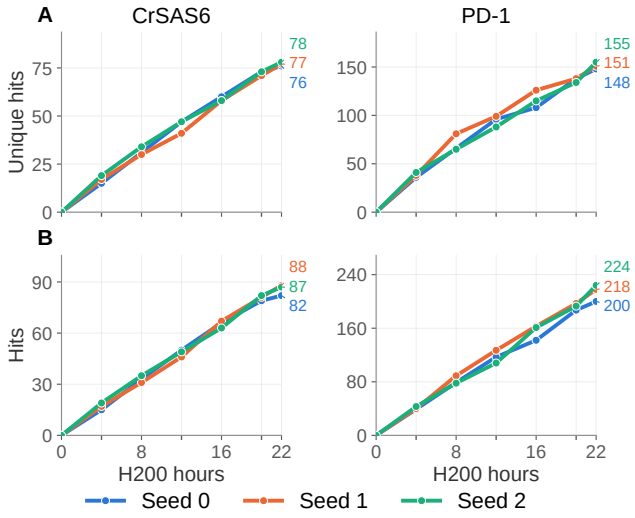

Figure 12: CGA across three random seeds for RFDiffusion3 on CrSAS6 and PD-1. (A) Unique hits; (B) total hits, versus charged H200 hours. Labels show 22-hour counts.

D.5 VALIDATION OF THE MODELING CHOICES

We compare our two-process model with simple extrapolation baselines for predicting 22-hour naive sampling outcomes from observations available after 8 hours. Let N_8, K_8, H_8, and U_8 denote the numbers of generated candidates, discovered clusters, total hits, and unique hits at 8 hours of naive sampling. The naive candidate budget is $N_{22} = \left\lfloor \frac{22\times3600}{c_G+c_E} \right\rfloor$, since naive sampling generates and evaluates every candidate. We compare three estimators:

- **Empirical rate**: This estimator assumes that the numbers of discovered clusters and unique hits per generated candidate remain constant: $\widehat{K}_{22} = N_{22}\frac{K_8}{N_8}, \widehat{U}_{22} = N_{22}\frac{U_8}{N_8}$.
- **Linear fit**: This estimator fits to each full prefix curve: cluster count versus generated candidate count and unique hit count versus generated candidate count. It then evaluates these lines at N_{22}.
- **Pitman-Yor (PY) with empirical hit rate**: This estimator models cluster discovery using the PY process fitted from 8-hour of CGA run but assigns every cluster the same candidate hit probability, using $p = H_8/N_8$ as an empirical proxy.

Empirical rate and linear estimators contain no model of cluster discovery or cluster quality, and therefore extrapolate directly from an 8-hour prefix of naive sampling. In contrast, our model explicitly estimates these latent processes, allowing posteriors fitted from an 8-hour CGA trajectory to predict the outcome of a 22-hour naive campaign.

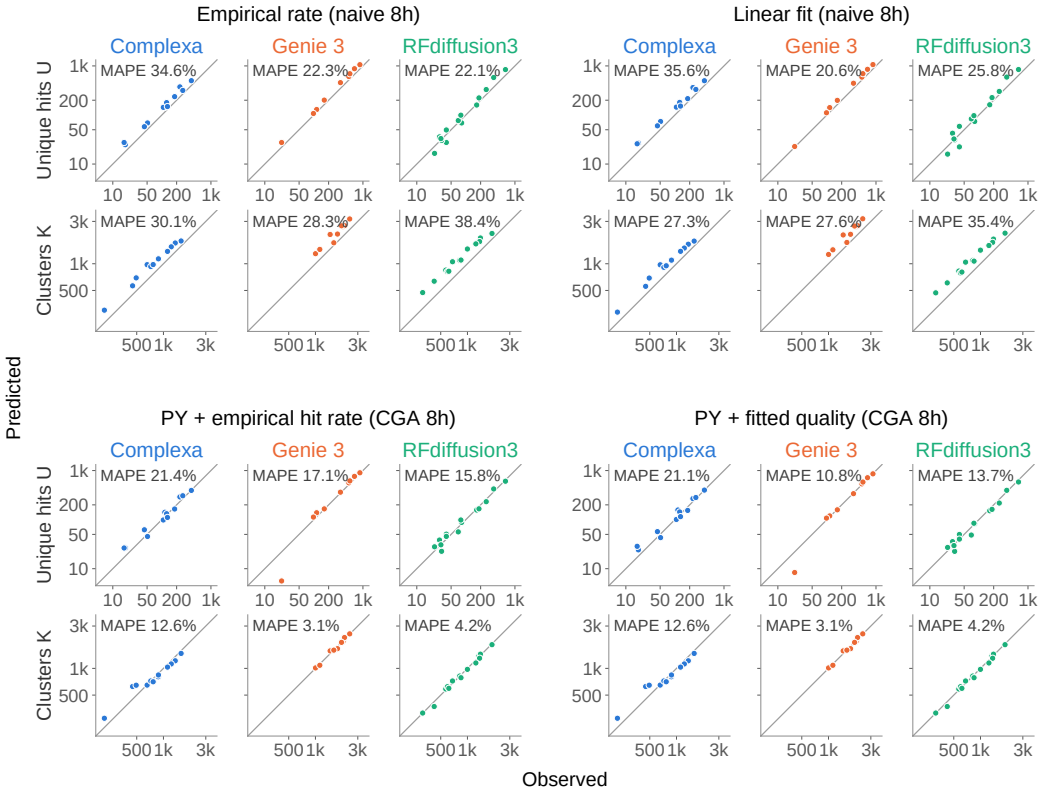

Figure 13: **Predicting cluster and unique hit counts after 22 hours of naive sampling.** Each group shows predictions from a different estimator; PY + fitted quality is our model.

Fig. 13 shows that PY process significantly reduces cluster count and unique hit MAPE across generators. The empirical rate and linear estimators generally overestimate the cluster and hit discoveries. These results support modeling the changing rate of cluster discovery when extrapolating campaign outcomes. Adding fitted heterogeneous quality further reduces unique hit MAPE.