CTRL-P: MULTI-SCENARIO GENERATIVE PROTEIN DESIGN VIA PREFERENCE ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative models can now design proteins that perform one function in one context: bind a given target, scaffold a given motif, catalyze a given reaction. However, many applications need a sequence that behaves differently depending on its context, like a binder that engages two unrelated targets, or a switch that adopts one fold on its own and another once a ligand is bound. Training a model for such multi-scenario tasks directly is impractical, since structural data of the same protein across contexts is scarce. To address this, we introduce Control-Proteina (`Ctrl-P`), which instead coordinates separately trained models at inference time, eliminating the need for training on multi-scenario data: each model expresses its preferences as a population of sequences and conditions its own generation on the preferences of the others to search for designs that succeed in every context. In a multi-target protein–protein interaction benchmark that spans similar and dissimilar binding interfaces, `Ctrl-P` solves 18 of 20 target pairs, including 8 of the 10 challenging dissimilar pairs where the strongest baseline solves 2. We further demonstrate the generality of our approach by applying `Ctrl-P` to design allosteric motif switches that adopt specified conformations in response to a binding partner: it solves 8 of 12 tasks, compared to the strongest baseline's 7, while finding more than twice as many successes per task on average. A case study on pan-arenavirus binder design shows that `Ctrl-P` generalizes beyond two scenarios, yielding candidates predicted to engage all three targets where no evaluated baseline yields any. Together, these results extend protein design toward programmable behavior across molecular contexts.

1 INTRODUCTION

Machine learning models are increasingly capable of reliably generating proteins that scaffold functional motifs (Watson et al., 2023; Geffner et al., 2025a), bind other proteins (Pacesa et al., 2025; Didi et al., 2026b), small molecules (Krishna et al., 2024; Didi et al., 2026a), or nucleic acids (Glasscock et al., 2025), or catalyze chemical reactions (Lauko et al., 2025). However, these proteins are designed for a single objective, with a single behavior in mind. Yet, designing proteins that behave differently in different contexts would unlock new capabilities, such as cross-reactive binders that recognize multiple targets, or proteins that are active in one context but inert in another. The latter would enable more precise therapeutics, such as designed switches that activate only on a specific combination of cell-surface antigens for cancer therapy (Lajoie et al., 2020).

We refer to this as multi-*scenario* design, where a scenario is a context, such as the other biomolecules present with the protein, and a desired outcome, such as binding to the other molecule, or adopting a particular fold. Unfortunately, there is very limited structural data of proteins available in different conformations or contexts, making it difficult to directly train a model for multiple scenarios. Moreover, even if such data for two constraints per protein existed, we would want the ability to design for an arbitrary number of conditions.

To address this, we propose to coordinate multiple single-scenario models to find designs that satisfy all scenarios simultaneously. Existing coordination methods either average the per-residue logits of an inverse folding model decoded jointly over several structures, as in tied ProteinMPNN decoding (Dauparas et al., 2022), or modify the SDE or ODE of the generative process to compose the score fields of several models (Liu et al., 2022; Bar-Tal et al., 2023; Garipov et al., 2023; Skreta et al.,

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.

2025). Both are tied to the model they coordinate, and the latter is especially limiting: state-of-the-art protein models rely on samplers tuned well away from the process they were trained to reverse, such as RFdiffusion's reduced inference noise (Watson et al., 2023) or the separate time schedules for structure and latents in Proteina and Proteina-Complexa (Geffner et al., 2025b; Didi et al., 2026b). Composition rules derived for the unmodified SDE carry no guarantees under these heuristics, and empirically these methods fare poorly: SuperDiff applied to Proteina-Complexa solves none of the tasks in our benchmark.

Instead, we propose an algorithm that alternates between sampling from the different models, similar to a negotiation, as illustrated in Fig. 1. In order to achieve coordination, we train the models to condition on each other's preferences, allowing them to reach a consensus that satisfies all of their objectives jointly. The sampling step is then just a black-box subroutine and does not require modifying the sampling process, does not require logits, and can be used with any choice of generative models, including faster distilled flow models, as long as they are trained with our preference-conditioning.

As a demonstration, we finetune a state-of-the-art protein design model, Proteina-Complexa (Didi et al., 2026b), to condition on preferences, yielding Control-Proteina (`Ctrl-P`). We demonstrate that combined with our negotiation protocols, `Ctrl-P` can design dual-target protein binders, solving 18 of the 20 pairs in our benchmark against 12 for the strongest baseline, with the margin concentrated on the dissimilar-interface pairs (8/10 against 2/10). We further demonstrate the generality of our framework by using `Ctrl-P` to design allosteric protein switches, solving 8 of 12 tasks, compared to 7 solved by SwitchCraft (Jing et al., 2026), a recent method for designing fold-switching proteins, with on average 2.4 unique successes against its 1.0. Finally, we apply our method to a three-target arenavirus case study, where `Ctrl-P` recovers candidates predicted to engage all three viral targets and no evaluated baseline recovers any. Together this shows that by training models to condition on other models' preferences, they can be efficiently coordinated to design proteins with complex behaviors that depend on context.

2 Related Work

Single-objective protein design. Most generative design methods target one context at a time, whether by conditional generation (Watson et al., 2023; Geffner et al., 2025a; Didi et al., 2026b; Stark et al., 2025; Protenix Team et al., 2025) or by optimizing a sequence through a structure predictor (Pacesa et al., 2025; Cho et al., 2025). These methods differ in architecture and in how they allocate inference-time compute, but they share the assumption of a single specification; we instead ask a design to satisfy several at once.

Multi-objective protein design. Two recent methods address context-dependent behavior directly: Chamaileon (Cao et al., 2026) trains a contextualized model for multi-target and multi-state binder design, and SwitchCraft (Jing et al., 2026) optimizes a compositional loss by backpropagating through a co-folding model. Both couple the coordination mechanism to a particular model. We unify these tasks under multi-scenario design and give a procedure that composes any generative model trained with preference conditioning.

Combining generative models. A broad line of work composes pretrained diffusion models by summing or averaging their scores (Liu et al., 2022; Bar-Tal et al., 2023), by classifier guidance on a composed distribution (Garipov et al., 2023), or by reweighting vector fields to sample from products and mixtures (Skreta et al., 2025). These methods act on the generative process itself, which conflicts with the tuned samplers protein models depend on, as discussed above. We instead leave the sampler untouched and train the model to condition on other models' preferences, so coordination is a property of the conditioning rather than of the SDE. This keeps us free to choose any sampling procedure, schedule, or distilled model, and lets us spend train-time compute on the coordination mechanism itself. A more detailed discussion is given in Appendix A.

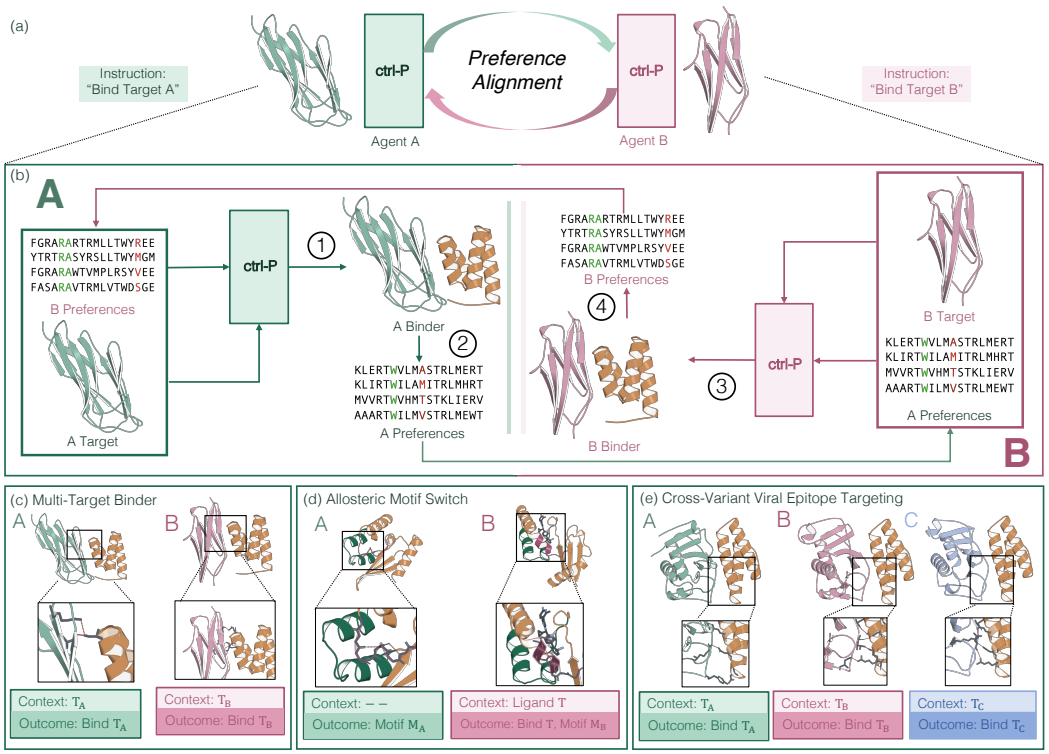

Figure 1: **Overview of `Ctrl-P`. (a)** Preference alignment coordinates models for scenarios A and B. **(b)** One cycle of the two-scenario negotiation: (1) the A-side model designs against target A while conditioning on B's preferences; (2) inverse folding the resulting design produces a population of sequences expressing A's preferences; (3) the B-side model uses these preferences to design against target B; and (4) inverse folding that design updates B's preferences for the next cycle. **(c,d)** Two benchmark tasks, expressed as (context, outcome) pairs: binding T_A or T_B in their respective contexts (c), and scaffolding M_A without a ligand or binding ligand T while scaffolding M_B (d). **(e)** A three-target viral case study, with separate predicted complexes for MACV, JUNV, and SABV GP1.

3 METHODS

Standard protein design tasks, such as motif scaffolding, binder design, or enzyme design, can be thought of as specifications of what a protein should do in a given context. For binder design, the context is the target molecule, and the desired outcome is that the design and target form a complex. We define this pair of a context and desired outcome as a *scenario*.

Our objective is to find a single sequence that satisfies J scenarios, even though its structure may differ between them. As a concrete example, an allosteric fold-switching protein can be defined by two scenarios, A and B: in the absence of a small-molecule ligand (A), the protein should scaffold motif M_A, whereas in the presence of the ligand (B), it should instead scaffold motif M_B. This is illustrated in Fig. 1(d).

We distinguish a sequence $s \in \mathcal{A}^L$, where $\mathcal{A}$ is the amino-acid alphabet, from a structural design x, which includes a sequence and its conformation in one context. A conditional generator $p_\theta(x \mid C_j)$ produces designs for scenario C_j. The object shared across scenarios is the sequence, and each scenario has an accompanying objective $f_j(s)$ which may depend on a folding model.

We coordinate these generators through sequence populations. Given a design from a generator, we sample a population of sequences compatible with its design and scenario using an inverse folding model. These sequences act like a representation of the generator's preferences. We then train the generators to accept such populations as an additional condition, allowing them to condition on populations derived from other models and scenarios at inference time.

3.1 Representing Preferences as Sequence Populations

Let $q_\tau(s \mid z, C)$ be an inverse folding model with sampling temperature τ. For a structure z in context C, we construct a population

$$S = (s^{(1)}, \ldots, s^{(P)}), \qquad s^{(p)} \sim q_\tau(\cdot \mid z, C). \tag{1}$$

Each row is a length-L sequence for the designed protein, and each column corresponds to the same residue position. Conserved columns express a preference for particular residues, while variation within and across rows retains alternatives and correlations that would be lost by passing only a consensus sequence.

Temperature controls the concentration of this signal. Low τ yields closely related sequences; higher τ allows more variation. We use a fixed ProteinMPNN decoding order so that greedy decoding at $\tau = 0$ is deterministic (App. B.1). We also provide τ to the generator, yielding $p_\theta(x \mid C, S, \tau)$. A separate masked-population condition, denoted $\tau = \infty$, removes preference conditioning altogether.

3.2 Training the Preference-Conditioned Generator

`Ctrl-P` is a a conditional generator $p_\theta(x|C, S, \tau)$ trained to condition on both a target scenario and τ-dependent sequence population preferences S.

Constructing training examples. Training requires only single-scenario examples (x, C). For each example, we inverse fold its structure to obtain S at a sampled temperature, then train p_θ to recover x given (C, S, τ). Here the population is only an additional input and the supervised output remains the original structure. No paired structures of the same sequence in different scenarios are needed. The populations are sampled at random temperatures $\tau \in [0, 5]$. At training time we also train with a special $\tau = \infty$ token which masks the population conditioning. See B.1 for more details.

Conditioning and objective. We extend the Proteina-Complexa (Didi et al., 2026b) architecture with a population encoder that takes an $L \times P \times 20$ one-hot representation, processes the residue and population axes with axial attention, and produces $L \times d$ residue-level features. These features are added to the generator's residue-level representation, allowing the denoising network to use the population alongside its ordinary scenario conditioning.

The training objective is the same as the base Proteina-Complexa loss with the additional sampled populations and temperature as inputs:

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,C),\,\tau,\,S,\,\eta}\left[\ell_{\text{base}}\big(\theta; x, C, S, \tau, \eta\big)\right]. \tag{2}$$

Here, S is generated from the training structure as in Eq. 1, and η denotes the noise and time samples used to construct the base model's noisy training input. We finetune from a pretrained Proteina-Complexa checkpoint.

Why this supports cross-scenario conditioning. Under this training construction, the conditional data distribution satisfies

$$p_\theta(x \mid C, S, \tau) \propto p_{\text{data}}(x \mid C)\, q_\tau(S \mid x, C), \tag{3}$$

where $q_\tau(S \mid x, C)$ denotes population generation from the structure in x. Thus, the model learns to generate structures compatible with both the scenario and the supplied population. At inference time, we supply preferences from a different scenario. This is therefore an indirect mechanism by which the model conditions on separate scenarios. Indeed we show in App. D.2 that the trained model reliably trades off retention of the source scenario against performance in the active scenario as a function of the temperature τ.

For motif switching, we train a separate preference-conditioned model for motif scaffolding and small-molecule binding using AFDB and PLINDER (Varadi et al., 2024; Durairaj et al., 2024). See App. B.1 for more details.

3.3 Negotiating a Shared Sequence

Given a trained model $p_\theta(x|C, S, \tau)$ which can condition on preferences, we use a negotiation procedure to coordinate models across scenarios.

Search state and objective. A negotiation evolves a sequence s_k to satisfy J scenarios with objectives f_j. We orient all objectives so that larger is better and aim to maximize the worst-scenario objective,

$$G(s) = \min_{1 \le j \le J} f_j(s). \tag{4}$$

For multi-target binding, the per-scenario selection score is ipSAE. For motif switching, the component metrics have different units and directions, so we calibrate them with sigmoid gates centered on their success thresholds (App. D.5).

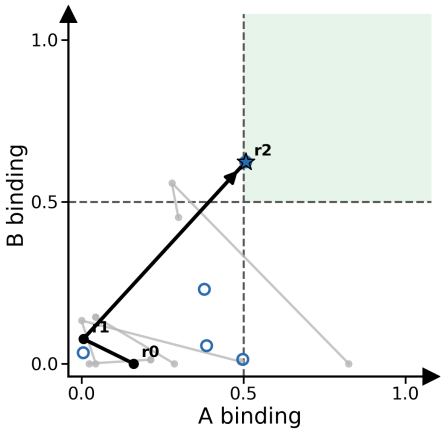

Figure 2: **Search dynamics.** A trajectory in the two-scenario score space retains an incumbent while sampling new proposals. At round 2, the star marks the selected design among N proposals (blue circles). The shaded quadrant is the dual-success region.

Initialization and preference exchange. We initialize by generating a design for a randomly chosen scenario without population conditioning, extracting its sequence, and predicting and scoring that sequence in every context. At round k, we choose the scenario with the lowest score, $j_k \in \arg\min_j f_j(s_k)$. For each other scenario i, we inverse fold the incumbent's prediction $z_i(s_k)$ to obtain a population $S_{i,k}$. The active generator receives their row-wise concatenation,

$$S_{-j_k,k} = \mathrm{concat}_{\mathrm{rows}}\big(S_{i,k} : i \neq j_k\big). \tag{5}$$

Columns retain the same residue correspondence since concatenation only increases the population size, and not the sequence length. For two scenarios, this simply passes the other scenario's population. For $J > 2$, it passes preferences from all other scenarios through the same conditioning interface.

Generating and evaluating proposals. We generate N proposals conditioned on C_{j_k} and $S_{-j_k,k}$. In our implementation, generation is seeded with the incumbent sequence's folding prediction in the active context: we add noise to this prediction at an intermediate sampling step and denoise with the preference-conditioned model. This gives the generator a structural starting point for the current sequence while allowing it to change both sequence and structure. We denote this seeded sampling procedure by $\mathcal{G}_\theta$ in Algorithm 1; App. D.3 evaluates the seeding step.

For each proposal, we extract its sequence and predict its structure in every context. We then evaluate all scenario scores on those predictions. In particular, a good generated structure in the active context is not sufficient for acceptance. We choose the sequence with the largest worse-scenario score G from the proposals and incumbent, retaining the incumbent unless a proposal strictly improves it. The next round constructs fresh preferences from the selected sequence's context-specific predictions.

Preference strength and termination. The schedule $\tau_k = T(G(s_k))$ sets how strongly the active model follows the other scenarios. Poor incumbents receive weak preferences (high temperature), allowing larger changes, while better incumbents receive more concentrated populations. At $\tau_k = \infty$, the population input is masked rather than sampled. See App. D.4 for the score-dependent schedule. We run each trajectory for K rounds and return its last design, while multiple parallel trajectories explore different initial designs.

Under fixed evaluation, retaining the incumbent ensures $G(s_{k+1}) \geq G(s_k)$, as illustrated in Fig. 2. For bounded scores, the aggregate score therefore converges as the number of rounds increases, though this does not imply that every individual score improves, that the sequence converges, or that a jointly successful design exists or will be found.

Finally, we note two extensions that we explored but that did not empirically improve results. First, the negotiation procedure also permits other proposal mechanisms. We evaluate tied inverse folding in App. D.8. Second, in App. D.9 we also tried a population search variant of our algorithm that expands a shared candidate pool instead of independent trajectories. Unless stated otherwise, our experiments use independent trajectories with the negotiation algorithm above. While our benchmarks focus on two scenarios for simplicity, section 4.3 also tests the extension to three scenarios.

Algorithm 1: Stochastic Negotiation of a Shared Sequence

Input: Scenarios $C_1, \ldots, C_J$; predictors F_j and scores f_j; generator $\mathcal{G}_\theta$; inverse folder q; K rounds; N proposals per round; temperature schedule T
Output: Incumbent sequence s_K
Choose j_0 uniformly from $\{1, \ldots, J\}$
Generate x_0 for C_{j_0} without preferences; extract s_0
Predict $z_i(s_0) = F_i(s_0, C_i)$ and evaluate $f_i(s_0)$ for every i
for $k = 0, \ldots, K-1$ **do**
  $j \leftarrow \arg\min_i f_i(s_k)$; $\tau_k \leftarrow T(G(s_k))$
  foreach $i \neq j$ **do**
    Sample P sequences from $q_{\tau_k}(\cdot \mid z_i(s_k), C_i)$ to form $S_{i,k}$
  $S \leftarrow \mathrm{concat}_{\mathrm{rows}}(S_{i,k} : i \neq j)$
  for $n = 1, \ldots, N$ **do**
    $x^{(n)} \leftarrow \mathcal{G}_\theta(C_j, S, \tau_k;\ z_j(s_k))$; extract sequence $s^{(n)}$
    Predict $z_i(s^{(n)})$ and evaluate $f_i(s^{(n)})$ for every i
  $s_{k+1} \leftarrow \arg\max_{s \in \{s_k, s^{(1)}, \ldots, s^{(N)}\}} G(s)$, retaining s_k on ties
return s_K

4 Experiments

4.1 Multi-Target Binder Design

Benchmark and Evaluations We evaluate multi-target binder design on a benchmark of target pairs (C_1, C_2). We stratify the pairs into easy pairs with similar target interfaces and more challenging pairs with dissimilar interfaces. Starting from a pool of targets that are individually easy to design against, we compute all-pairs interface similarity (iAlign IS-score (Gao & Skolnick, 2010)) between targets. After deduplicating near-identical binding sites, we obtain 10 *similar* (SIM) and 10 *dissimilar* (DIS) target pairs. Full construction details are given in App. C. We also considered the multi-conformation benchmark from Cao et al. (2026), but found it to be saturated by a simple single-target baseline. See App. C.2 for details.

For each task instance, we generate binders at a fixed compute budget of around 4 GPU-hours, and score them against *both* targets in the task. Because of the multiple contexts, we cannot always compare refolded structures to model-generated structures, therefore a candidate only corresponds to a sequence, rather than a sequence and structure, and we use confidence metrics from a folding model's prediction of the sequence's structure to score. This type of evaluation is also routinely used for sequence-only design models (Lin et al., 2023). A design is a dual success if it satisfies ipSAE > 0.5 (Dunbrack, 2025) and pLDDT > 0.9 (Jumper et al., 2021) against both C_1 and C_2 simultaneously. We use Boltz-2 (Passaro et al., 2025) (with the crystal structure of the target as a template).

Baselines We compare against the previously published method Chamaileon (Cao et al., 2026), as well as naive baselines that condition on a single target, and methods that were not evaluated against multi-target design, but that could be straightforwardly adapted to the task. Run configurations and compute budgets are given in App. B.2.

1. **Single-target Proteina-Complexa** The naive baseline generates binders with Proteina-Complexa conditioned on one target at a time, without preferences from the other target. We sample against each target separately and evaluate dual success against both targets post hoc.
2. **Single-target search** Same generation as above, but we additionally use the beam search of Didi et al. (2026b), scoring against *both* C_1 and C_2.
3. **BindCraft** We run BindCraft (Pacesa et al., 2025) by summing the objectives over the two targets.

Table 1: Multi-target binder design results on the 10 SIM / 10 DIS pair benchmark. We report the number of tasks solved and the average number of TM-unique dual successes per task for each method.

Method	DIS solved / 10	DIS avg. unique	SIM solved / 10	SIM avg. unique
Single-Target				
Proteina-Complexa	0	0.0	5	4.3
Proteina-Complexa + beam search	2	1.0	**10**	**9.5**
Multi-Target				
Proteina-Complexa + SuperDiff	0	0.0	0	0.0
BindCraft	0	0.0	3	0.3
Chamaileon	0	0.0	0	0.0
`Ctrl-P`	**8**	**1.5**	**10**	5.3

Table 2: TM-unique dual successes per task. Gray cells are unsolved pairs (0 successes); green cells have at least one unique dual success.

Method	DIS 1	2	3	4	5	6	7	8	9	10	SIM 1	2	3	4	5	6	7	8	9	10
Proteina-Complexa	0	0	0	0	0	0	0	0	0	0	2	0	32	5	0	0	0	3	1	0
Proteina-Complexa + beam search	0	0	0	0	0	0	3	0	7	0	21	13	25	12	5	3	4	4	7	1
Proteina-Complexa + SuperDiff	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BindCraft	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0
Chamaileon	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
`Ctrl-P`	0	1	3	5	0	1	1	1	2	1	7	9	12	6	3	2	1	8	3	2

4. **SuperDiff** We use the SuperDiff AND operator (Skreta et al., 2025) to sample from the product of Proteina-Complexa conditioned on each target.

5. **Chamaileon** We sample designs using the model and MOPS with beam search algorithm from Cao et al. (2026).

Results Table 1 reports the number of pairs solved (out of 10 per group, solved if at least one success is found) and the average number of TM-unique dual successes per task. `Ctrl-P` solves 18 of the 20 pairs—every SIM pair and 8/10 DIS pairs—against 12 for the strongest baseline, averaging 5.3 SIM and 1.5 DIS TM-unique dual successes per task (Table 1). The single-target baselines struggle on the hard tasks. Proteina-Complexa alone cannot solve any, while Proteina-Complexa combined with multi-target beam search only does marginally better. BindCraft, Proteina-Complexa with SuperDiff AND, and Chamaileon all perform poorly as well. Table 2 breaks these unique successes down by individual target pair. The dissimilar group is harder and more uneven across pairs than the similar group. Several DIS pairs yield only a handful of unique successes for any method. App. D.7 examines how progress and time to first success differ between SIM and DIS pairs.

We note there is a trade-off between breadth—finding more unique successes on tasks where we can already find one—versus depth—finding the first success on more tasks. Compared to `Ctrl-P`, Proteina-Complexa with beam search does well on breadth, finding many successes on the easy tasks, but it suffers on depth. We can make a similar trade-off with `Ctrl-P` by using the population search algorithm that we explore in App. D.9. However, while that can find more successes on the easy tasks, it does not improve on the harder ones, and also does worse overall on the motif switching task we consider in the next section. As a trade-off, it is preferable to be able to solve more tasks than to get a few more successes on already solved tasks, since that expands the range of design problems that can be addressed.

Table 4: Motif-switching successes per task. Gray cells are unsolved tasks (0 successes); green cells have at least one successful design, labeled with the count.

	OQO						FAD					
Method	3IXT →1YCR	1YCR →3IXT	3IXT →1PRW	1PRW →3IXT	1YCR →1PRW	1PRW →1YCR	3IXT →1YCR	1YCR →3IXT	3IXT →1PRW	1PRW →3IXT	1YCR →1PRW	1PRW →1YCR
SwitchCraft	2	2	2	1	2	2	1	0	0	0	0	0
`Ctrl-P`	8	3	1	5	7	1	1	0	0	0	0	3

4.2 Allosteric Motif Switches

To demonstrate the generality of our approach, we also apply it to designing allosteric protein switches. Since this is the composition of two scenarios involving both motif scaffolding and ligand binding, we finetune a version of `Ctrl-P` for small molecule binding and motif scaffolding.

Benchmark and Evaluations We construct a benchmark of allosteric motif switching tasks based on the motif and small molecule combinations in SwitchCraft (Jing et al., 2026). Concretely, we take 3 motifs (1PRW, 1YCR, 3IXT) from other tasks they consider, and 2 ligands (OQO, FAD), including the single combination they evaluate on for motif-switching (3IXT and 1YCR + OQO). We then create 12 tasks corresponding to all 6 directional pairs of motifs and 2 ligands. We use the same success criteria as in their evaluations, with the addition of a ligand binding gate requiring that the ligand minimum iPAE be under 2Å.

Results Table 3 summarizes our results. `Ctrl-P` solves 8/12 tasks, compared to SwitchCraft's 7 at matched compute, and also finds more unique successes per task on average (2.4 vs. 1.0). Table 4 breaks these successes down by individual task. Both methods find essentially all of their successes on the OQO tasks. The FAD tasks, with their larger and more complex cofactor, are uniformly hard, with only 5 successes found across all 6 FAD tasks and both methods combined. Compared to the multi-target binder task, these allosteric switches are harder because of the switching requirement: each motif must be scaffolded in one scenario, but not the other. The objectives are therefore fundamentally conflicting with each other, compared to the multi-target case where targets may still share some similarities.

Table 3: Allosteric motif-switching results, aggregated over the 12-task sweep. We report the number of tasks solved (out of 12, solved if at least one success is found) and the average number of unique successes per task.

Method	solved / 12	avg. unique
SwitchCraft	7	1.0
`Ctrl-P`	**8**	**2.4**

4.3 Beyond Two Conditions: A Viral Case Study

Can preference alignment generalize to an arbitrary number of conditions, beyond the two partners considered in our benchmarks? The conditioning interface is not restricted to pairs: with J scenarios, each instance can receive the concatenated sequence populations of the other $J - 1$ instances. We test this extension in an important biological case study with three conditions.

We choose cross-reactive recognition of the GP1 domains of Machupo (MACV), Junín (JUNV), and Sabiá (SABV) viruses. These New World arenaviruses cause severe hemorrhagic disease and share a host receptor (Radoshitzky et al., 2007). Recognition across related viral targets is thus biologically relevant, but shared receptor usage does not imply identical antigenic surfaces. Evidence that antibodies can cross-neutralize members of this group (Clark et al., 2018) motivates studying recognition across multiple targets. Here, the computational objective is one sequence predicted to bind each of the three GP1 domains in its respective context.

Three scenarios, pooled preferences. Each target defines a separate scenario, with one `Ctrl-P` instance conditioned on that target and on the sequence preferences of the other two instances (Fig. 3a). We combine those preferences by concatenating the two sequence populations along the population dimension: sequence rows are stacked while residue columns and sequence length

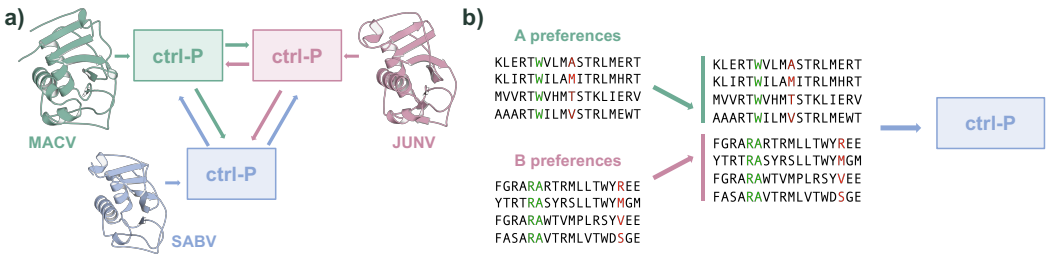

Figure 3: **Preference alignment across three viral target contexts. (a)** One `Ctrl-P` instance per target. Each instance receives the sequence preferences of the other two. **(b)** Illustration of pooling preferences from A and B to condition C by concatenating sequence rows while preserving residue columns and sequence length. The displayed sequence blocks are schematic assets reused from Fig. 1, not the sequences of the viral candidates.

remain fixed (Fig. 3b). This extends the same preference-conditioning interface to three scenarios without changing the model architecture.

Evaluation and results. We ran each method for 12 hours and evaluated candidates against all three targets using AF2 predictions. A hit must satisfy both hard constraints, $\mathrm{ipTM} > 0.8$ and directional binder-to-target minimum interface PAE < 1.5 Å, in every target context. These constraints are applied jointly: a strong prediction for one target cannot compensate for failing either threshold on another. `Ctrl-P` produced **17 hits**, whereas **every evaluated baseline produced zero** within the same wall-clock budget. The budget is matched by elapsed time rather than by the number of evaluated candidates.

Hard prediction-confidence filters are also used in recent PPI design studies, including RFdiffusion3 (Butcher et al., 2025). Its PPI benchmark uses AF3 minimum inter-chain PAE ≤ 1.5 Å, binder pTM ≥ 0.8, and target-aligned binder Cα RMSD < 2.5 Å. Our AF2 ipTM/min-iPAE gate is therefore a case-specific evaluation criterion, not an identical reproduction of that protocol. The 17 hits are computational candidates; the confidence thresholds do not establish experimental binding or neutralization. App. D.10 and Fig. 12 show the target-side interface views from the three predicted complexes.

5 Conclusion

We introduced Control-Proteina, a preference-conditioned approach to coordinating protein design across molecular contexts. By exchanging sequence populations, specialized generative models can contribute to a shared design objective without requiring a model trained directly on every combination of scenarios. The results span dual-target binding, ligand-dependent motif switching, and a three-target viral case study. `Ctrl-P` solves 18 of 20 target-pair tasks and solves 8 of 12 motif-switching tasks, compared with 7 for SwitchCraft, while finding 2.4 versus 1.0 unique successes per task on average. In the three-target case study, it finds 17 candidates meeting the joint AF2 gate within 12 hours, compared with none from the evaluated baselines.

These results support sequence-population conditioning as a general interface for composing design requirements at inference time. We conclude by noting some remaining limitations for future work. First, the model is not trained to condition on concatenated preferences. While this worked in our case study, this could be further improved to generalize to even more scenarios. Second, it is still unclear how to design *against* a scenario, and how to represent negative preferences. Finally, the present evidence remains computational. Successes depend on the chosen prediction models. Experimental testing and evaluation across additional contexts will be needed to establish how reliably predicted multi-scenario behavior translates into protein function.

6 USE OF AI

In this work, we used generative AI to write code that implements methods, supports data analysis and performs data cleaning and processing. We have not used generative AI tools for generating synthetic datasets, developing conceptual frameworks, or to propose research methodology and experiments, and writing proofs or formulating mathematical claims are not applicable to this work. Additionally, we used generative AI tools for creating figures, suggesting certain hyperparameters to sweep, and creating first drafts of parts of the appendix. We have reviewed all AI-assisted work. LLM-generated code was regularly verified and tested for correctness, and all figures and text created by agents were edited and proof-read by us. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. MultiDiffusion: Fusing diffusion paths for controlled image generation. In *International Conference on Machine Learning*, pp. 1737–1752. PMLR, 2023.

Jasper Butcher, Rohith Krishna, Raktim Mitra, Rafael Isaac Brent, Yanjing Li, Nathaniel Corley, Paul T. Kim, Jonathan Funk, Simon Valentin Mathis, Saman Salike, et al. De novo design of all-atom biomolecular interactions with RFdiffusion3. *bioRxiv*, 2025. doi: 10.1101/2025.09.18.676967. URL https://www.biorxiv.org/content/10.1101/2025.09.18.676967v2.

Hengyuan Cao, Shizhuo Cheng, Mingxuan Liu, Weicheng Huang, Yunhong Lu, Chenxi Cai, Yan Zhang, and Min Zhang. Chamaileon: Cross-context binder design with contextualized modeling and mixed sampling. In *International Conference on Machine Learning*, 2026. Spotlight.

Yehlin Cho, Griffin Rangel, Gaurav Bhardwaj, and Sergey Ovchinnikov. Protein hunter: exploiting structure hallucination within diffusion for protein design. *bioRxiv*, 2025. doi: 10.1101/2025.10.10.681530.

Lars E. Clark, Selma Mahmutovic, Donald D. Raymond, Taleen Dilanyan, Takaaki Koma, John T. Manning, Sundaresh Shankar, Silvana C. Levis, Ana M. Briggiler, Delia A. Enria, Kai W. Wucherpfennig, Slobodan Paessler, and Jonathan Abraham. Vaccine-elicited receptor-binding site antibodies neutralize two new world hemorrhagic fever arenaviruses. *Nature Communications*, 9:1884, 2018. doi: 10.1038/s41467-018-04271-z.

J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning–based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56, 2022. doi: 10.1126/science.add2187.

Kieran Didi, Danny Reidenbach, Matthew Penner, Supriya Ravichandran, Marshall Case, Mike Nichols, Erik Swanson, Alex Reis, Maggie Prescott, Yue Qian, et al. Latent generative search unlocks de novo design of untapped biomolecular interactions at scale. *bioRxiv*, pp. 2026–09, 2026a.

Kieran Didi, Zuobai Zhang, Guoqing Zhou, Danny Reidenbach, Zhonglin Cao, Sooyoung Cha, Tomas Geffner, Christian Dallago, Jian Tang, Michael M. Bronstein, Martin Steinegger, Emine Kucukbenli, Arash Vahdat, and Karsten Kreis. Scaling atomistic protein binder design with generative pretraining and test-time compute. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=qmCpJtFZra. Oral.

Roland L. Dunbrack. Rēs ipsae loquunt: what's wrong with AlphaFold's iptm score and how to fix it. *bioRxiv*, 2025. doi: 10.1101/2025.02.10.637595.

Janani Durairaj, Yusuf Adeshina, Zhonglin Cao, Xuejin Zhang, Vladas Oleinikovas, Thomas Duignan, Zachary McClure, Xavier Robin, Gabriel Studer, Daniel Kovtun, et al. Plinder: The protein-ligand interactions dataset and evaluation resource. *BioRxiv*, pp. 2024–07, 2024.

Mu Gao and Jeffrey Skolnick. ialign: a method for the structural comparison of protein–protein interfaces. *Bioinformatics*, 26(18):2259–2265, 2010.

Timur Garipov, Sebastiaan De Peuter, Ge Yang, Vikas Garg, Samuel Kaski, and Tommi Jaakkola. Compositional sculpting of iterative generative processes. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Tomas Geffner, Kieran Didi, Zhonglin Cao, Danny Reidenbach, Zuobai Zhang, Christian Dallago, Emine Kucukbenli, Karsten Kreis, and Arash Vahdat. La-Proteina: Atomistic protein generation via partially latent flow matching. *arXiv preprint arXiv:2507.09466*, 2025a.

Tomas Geffner, Kieran Didi, Zuobai Zhang, Danny Reidenbach, Zhonglin Cao, Jason Yim, Mario Geiger, Christian Dallago, Emine Kucukbenli, Arash Vahdat, and Karsten Kreis. Proteina: Scaling flow-based protein structure generative models. In *The Thirteenth International Conference on Learning Representations*, 2025b. Oral.

Cameron J. Glasscock, Robert J. Pecoraro, Ryan McHugh, Lindsey A. Doyle, Wei Chen, Olivier Boivin, Beau Lonnquist, Emily Na, Yuliya Politanska, Hugh K. Haddox, David Cox, Christoffer Norn, Brian Coventry, Inna Goreshnik, Dionne Vafeados, Gyu Rie Lee, Raluca Gordân, Barry L. Stoddard, Frank DiMaio, and David Baker. Computational design of sequence-specific DNA-binding proteins. *Nature Structural & Molecular Biology*, 32(11):2252–2261, 2025. doi: 10.1038/s41594-025-01669-4.

Bowen Jing, Mihir Bafna, Anisha Parsan, Heyuan Michael Ni, David Kwabi-Addo, Bryan Bryson, Adam Klivans, and Bonnie Berger. SwitchCraft: A programmatic framework for designing state-switching proteins. In *International Conference on Machine Learning*, 2026.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

Rohith Krishna, Jue Wang, Woody Ahern, Pascal Sturmfels, Preetham Venkatesh, Indrek Kalvet, Gyu Rie Lee, Felix S. Morey-Burrows, Ivan Anishchenko, Ian R. Humphreys, Ryan McHugh, Dionne Vafeados, Xinting Li, George A. Sutherland, Andrew Hitchcock, C. Neil Hunter, Alex Kang, Evans Brackenbrough, Asim K. Bera, Minkyung Baek, Frank DiMaio, and David Baker. Generalized biomolecular modeling and design with RoseTTAFold All-Atom. *Science*, 384 (6693):eadl2528, 2024. doi: 10.1126/science.adl2528.

Marc J. Lajoie, Scott E. Boyken, Alexander I. Salter, Jilliane Bruffey, Anusha Rajan, Robert A. Langan, Audrey Olshefsky, Vishaka Muhunthan, Matthew J. Bick, Mesfin Gewe, Alfredo Quijano-Rubio, JayLee Johnson, Garreck Lenz, Alisha Nguyen, Suzie Pun, Colin E. Correnti, Stanley R. Riddell, and David Baker. Designed protein logic to target cells with precise combinations of surface antigens. *Science*, 369(6511):1637–1643, 2020. doi: 10.1126/science.aba6527.

Anna Lauko, Samuel J. Pellock, Kiera H. Sumida, Ivan Anishchenko, David Juergens, Woody Ahern, Jihun Jeung, Alexander F. Shida, Andrew Hunt, Indrek Kalvet, Christoffer Norn, Ian R. Humphreys, Cooper Jamieson, Rohith Krishna, Yakov Kipnis, Alex Kang, Evans Brackenbrough, Asim K. Bera, Banumathi Sankaran, K. N. Houk, and David Baker. Computational design of serine hydrolases. *Science*, 388(6744):eadu2454, 2025. doi: 10.1126/science.adu2454.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pp. 423–439. Springer, 2022.

Martin Pacesa, Lennart Nickel, Christian Schellhaas, Joseph Schmidt, Ekaterina Pyatova, Lucas Kissling, Patrick Barendse, Jagrity Choudhury, Srajan Kapoor, Ana Alcaraz-Serna, Yehlin Cho, Kourosh H. Ghamary, Laura Vinué, Brahm J. Yachnin, Andrew M. Wollacott, Stephen Buckley, Adrie H. Westphal, Simon Lindhoud, Sandrine Georgeon, Casper A. Goverde, Georgios N. Hatzopoulos, Pierre Gönczy, Yannick D. Muller, Gerald Schwank, Daan C. Swarts, Alex J. Vecchio, Bernard L. Schneider, Sergey Ovchinnikov, and Bruno E. Correia. One-shot design of functional protein binders with BindCraft. *Nature*, 646(8084):483–492, 2025. doi: 10.1038/s41586-025-09429-6.

Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stark, David Kwabi-Addo, Dominique Beaini, Tommi Jaakkola, and Regina Barzilay. Boltz-2: Towards accurate and efficient binding affinity prediction. *bioRxiv*, 2025. doi: 10.1101/2025.06.14.659707.

Protenix Team, Milong Ren, Jinyuan Sun, Jiaqi Guan, Cong Liu, Chengyue Gong, Yuzhe Wang, Lan Wang, Qixu Cai, Wenzhi Ma, Yuxuan Zhang, Zhenyu Liu, Hanyu Zhang, Xinshi Chen, and Wenzhi Xiao. PXDesign: Fast, modular, and accurate de novo design of protein binders. *bioRxiv*, 2025. doi: 10.1101/2025.08.15.670450.

Sheli R. Radoshitzky, Jonathan Abraham, Christina F. Spiropoulou, Jens H. Kuhn, Dan Nguyen, Wenhui Li, Jane Nagel, Paul J. Schmidt, Jack H. Nunberg, Nancy C. Andrews, Michael Farzan, and Hyeryun Choe. Transferrin receptor 1 is a cellular receptor for new world haemorrhagic fever arenaviruses. *Nature*, 446(7131):92–96, 2007. doi: 10.1038/nature05539.

Marta Skreta, Lazar Atanackovic, Avishek Joey Bose, Alexander Tong, and Kirill Neklyudov. The superposition of diffusion models using the Itô density estimator. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hannes Stark, Felix Faltings, MinGyu Choi, Yuxin Xie, Eunsu Hur, Timothy O'Donnell, Anton Bushuiev, Talip Uçar, Saro Passaro, Weian Mao, Mateo Reveiz, Roman Bushuiev, Tally Portnoi, Tomáš Pluskal, Josef Sivic, Karsten Kreis, Arash Vahdat, Shamayeeta Ray, Jonathan T. Goldstein, Andrew Savinov, Jacob A. Hambalek, Anshika Gupta, Diego A. Taquiri-Diaz, Yaotian Zhang, Samuel J. Snyder, A. Katherine Hatstat, Angelika Arada, Nam Hyeong Kim, Haoyu Fan, Ethel Tackie-Yarboi, Dylan Boselli, Lee Schnaider, Chang C. Liu, Gene-Wei Li, Denes Hnisz, David M. Sabatini, William F. DeGrado, Jeremy Wohlwend, Gabriele Corso, Regina Barzilay, and Tommi Jaakkola. BoltzGen: Toward universal binder design. *bioRxiv*, 2025. doi: 10.1101/2025.11.20.689494.

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, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, 2023. doi: 10.1038/s41586-023-06415-8.

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.

Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

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.

Appendix

A Extended Related Work **14**
- A.1 Single-Objective Protein Design . . . 14
- A.2 Multi-Objective and Multi-State Protein Design . . . 15
- A.3 Combining Generative Models . . . 15

B Extended Methods **16**
- B.1 Model and Algorithm Details . . . 16
- B.2 Baselines . . . 16

C Multi-Target Benchmark **17**
- C.1 Benchmark Construction . . . 17
- C.2 Comparison to CROSS Benchmark . . . 18

D Additional Experiments **18**
- D.1 ProteinMPNN Temperature Profiling . . . 18
- D.2 Preference Conditioning Behavior . . . 18
- D.3 Folding Model Seeding . . . 19
- D.4 Temperature Schedule . . . 20
- D.5 Metric Calibration . . . 21
- D.6 Sequence Coordination . . . 21
- D.7 Round-by-Round Search Dynamics . . . 22
- D.8 Inverse Folding Proposals . . . 23
- D.9 Population Search . . . 24
- D.10 Viral Case Study: Predicted Interfaces . . . 25

A Extended Related Work

A.1 Single-Objective Protein Design

Structure-based de novo design has converged on two broad strategies. The first treats design as conditional generation: a diffusion or flow model is trained to sample structures given a specification, and sequences are recovered afterwards with an inverse folding model. RFdiffusion (Watson et al., 2023) established this route by fine-tuning a structure prediction network into a denoising diffusion model over backbone frames, and demonstrated motif scaffolding and binder design with experimental validation. Subsequent work extended the representation beyond backbones: RoseTTAFold All-Atom and RFdiffusionAA (Krishna et al., 2024) model small molecules, nucleic acids and covalent modifications atomistically, while Proteina (Geffner et al., 2025b) and La-Proteina (Geffner et al., 2025a) scale flow-based generation to longer proteins and to joint sequence–structure generation via partially latent representations. BoltzGen (Stark et al., 2025) unifies design and structure prediction in a single all-atom model spanning many binder modalities.

The second strategy treats design as optimization through a structure predictor, often called hallucination. BindCraft (Pacesa et al., 2025) backpropagates through AlphaFold2 to optimize a binder sequence directly against a target, reporting high experimental success rates without screening. Protein Hunter (Cho et al., 2025) instead cycles between all-atom structure prediction and inverse folding, exploiting the tendency of diffusion-based predictors to hallucinate well-folded structures from uninformative inputs. PXDesign (Protenix Team et al., 2025) combines both, pairing a diffusion generator with confidence-model filtering.

Proteina-Complexa (Didi et al., 2026b), the base model we finetune, argues that this dichotomy is false: it pretrains a fully atomistic flow-based generator on the large synthetic Teddymer dataset and

then performs inference-time optimization with that generative prior, recovering the strengths of both families. What all of these methods share, and what we depart from, is that a design is scored against a single specification. The notion of a context in which the design should *not* act has no place in the objective.

A.2 Multi-Objective and Multi-State Protein Design

Designing a protein that behaves differently in different contexts has a long history in the rational-design literature, where switches were built from carefully engineered structural elements—for example LOCKR-style latches that expose a functional peptide only when displaced by a key, and the colocalization-dependent logic gates of Lajoie et al. (2020) that activate only in the presence of a specified antigen combination. These constructions are powerful but bespoke, and do not generalize to arbitrary targets or motifs.

Two recent learning-based methods tackle the problem generically. Chamaileon (Cao et al., 2026) formulates multi-target and multi-state binder design as cross-context binding landscape modeling, training with an in-context co-design objective and sampling with a mixture-of-paths strategy that optimizes one sequence across contexts. SwitchCraft (Jing et al., 2026) takes an optimization view, constructing a compositional loss over multiple predicted states and minimizing it by gradient descent through a co-folding model.

Our framing subsumes both. A scenario is a (context, outcome) pair, and multi-target binding, multi-state binding, and allosteric motif switching are all combinations of such pairs; we evaluate against Chamaileon on the first and SwitchCraft on the last. The methodological difference is where coordination lives. Chamaileon builds it into the model's training and sampling, and SwitchCraft into a differentiable loss, which requires the predictor to be differentiable and the optimization to be run per design. We place coordination in the conditioning signal instead, which leaves the underlying sampler and its hyperparameters untouched.

A.3 Combining Generative Models

Outside protein design, composing pretrained generative models is a well-studied problem. Liu et al. (2022) compose diffusion models by summing their scores, interpreting the result as a product of experts over concepts, and MultiDiffusion (Bar-Tal et al., 2023) binds several diffusion trajectories through a least-squares consistency objective at each step to satisfy multiple spatial constraints. Garipov et al. (2023) define compositions such as harmonic means and contrasts of iterative generative processes and sample from them with classifier guidance. SuperDiff (Skreta et al., 2025) derives a superposition framework from the continuity equation and provides logical AND and OR operators via a scalable Itô density estimator, with an automated reweighting of the component vector fields.

Within protein sequence design, the analogous tool is tied decoding in ProteinMPNN (Dauparas et al., 2022), where positions are coupled across chains or structures and their logits averaged before sampling. This is the natural multistate baseline, and we adopt it both as a baseline and as an alternative proposal kernel inside our search algorithms.

The common thread is that each method intervenes in the generative process: it averages logits at the decoding step, or modifies the drift of the SDE, or adds a guidance term to the score. This is precisely what is expensive for protein generative models, whose samplers are tuned heuristically and often far from the process the model was trained to reverse—reduced inference noise in RFdiffusion (Watson et al., 2023), and separate integration schedules with time-dependent rescaling for structure and latents in Proteina and Proteina-Complexa (Geffner et al., 2025b; Didi et al., 2026b). Composition identities derived for the unmodified process do not transfer cleanly to these samplers; App. D.6 illustrates a trade-off between sequence-guided structure recovery and binder design quality. By moving coordination into the conditioning, our approach is agnostic to the sampler: any model that accepts preference conditioning can participate, including distilled or few-step models, and the coordination quality becomes a function of training rather than of a hand-tuned guidance weight.

B Extended Methods

B.1 Model and Algorithm Details

Fixed-Order PMPNN Decoding Usually, ProteinMPNN decodes sequences in a random order, and even at sampling temperature $\tau = 0$ the sampled sequences are random. However, because we want the temperature to correspond with preference strength, $\tau = 0$ should be deterministic and encode a strong preference for a single sequence. We therefore use a fixed decoding order which greedily decodes positions with the lowest entropy. We found that this still gives good sequences with good structure recovery at low temperatures (see App. D.1), while giving deterministic samples at $\tau = 0$.

Training Data The protein binder model is trained with Teddymer and uses ProteinMPNN to sample population pools during training. Because we use ProteinMPNN to sample the sequence populations, we replace the native Teddymer sequences with greedy ProteinMPNN sequences and repack the side chains. This way, as the ProteinMPNN sampling temperature approaches 0, the sequence population concentrates around the training target sequence. This encourages the model to learn to generate sequences that correspond with the population conditioning. We sample 10 ProteinMPNN populations per training instance, each at a random temperature $\tau \in [0, 5]$, as we found that at temperature $\tau = 5$ the ProteinMPNN populations were close to random. See App. D.1 for details. We also add a special ∞ condition to the model, which masks the population condition. This corresponds to the limit $\tau \to \infty$ where preference conditioning is removed while scenario conditioning remains.

The ligand-motif model is trained in a similar way based on the Complexa-Ligand training from (Didi et al., 2026a). Starting from a La-Proteina (Geffner et al., 2025a) checkpoint trained for monomer generation, we first finetune for motif scaffolding, followed by population conditioning ad finally, LoRA finetuning for ligand binding to avoid overfitting. We generate the ProteinMPNN pools in the same way as for Teddymer, with the difference that we use LigandMPNN for the PLINDER data.

Fold-Model Seeding Conditioning on a sequence preference is inherently similar to folding, as the model needs to reason about how different sequences in the population would fold. However, we found that the finetuned Proteina-Complexa model had trouble with folding. We elaborate on this in Sec. D.3. To remedy this, we "seed" the model with the prediction from a folding model. Concretely, when sampling against scenario C_1 given a population derived from a design with sequence s for scenario C_0, we first use a folding model to predict the structure of s in context C_1. We then partially renoise this seed and denoise it with the generative model. We use a sweet spot for the noise that allows the design model to both freely modify the structure, while also understanding the structural neighborhood implied by the provided population.

Scoring Objectives Each scenario's objective may have a different scale, such as the ipSAE for binding and the motif-RMSD for motif scaffolding. Because we select based on the aggregated objective $\min_j f_j$, we must ensure the scores are comparable. For the motif-switching task, we therefore normalize each score with a sigmoid centered on its success threshold, mapping it to $(0, 1)$ with 0.5 at the threshold. For multi-target binding, we use raw ipSAE for both scenarios. The scale of the sigmoid is chosen based on the distribution of scores when sampling from the model. See App. D.5 for calibration.

B.2 Baselines

Here we provide details on the configurations used to run the baselines. Note that the GPU-hr accounting includes the time to evaluate the final samples, so that simple sampling baselines like Proteina-Complexa are still bound by the time required to score and rank the final designs.

1. **Proteina-Complexa** We sampled 325 samples against each target in each task for a total of 650 samples per task. The approximate GPU-hr per task was 3.88h.

2. **Proteina-Complexa + Beam Search** We sampled 5 trajectories against each target in each task, with a beam width of 4, a branching factor of 4, and 4 steps per trajectory, amounting to around 4.2GPU-hr/task.
3. **Proteina-Complexa + SuperDiff** We sampled 485 samples per pair, accounting for 3.84 GPU-hr/task.
4. **BindCraft** We sampled 22 trajectories per pair, for a total of 4.2 GPU-hr/task.
5. **Chamaileon** We sampled 50 designs per task, using their beam search settings, for a total of 4.5 GPU-hr/task. These same settings were also used to reproduce the results on their CROSS benchmark in App. C.2.
6. **SwitchCraft** We sampled 100 designs per task, following their setting, amounting to around 13 GPU-hr/task.

C Multi-Target Benchmark

To evaluate multi-target binder design we need pairs of targets (C_1, C_2) that span a controlled range of *interface* similarity: pairs whose binding sites are structurally similar and pairs whose binding sites are dissimilar.

C.1 Benchmark Construction

We build our benchmark by (i) selecting a pool of designable targets, (ii) characterizing each target by the interfaces of binders that succeed against it, and (iii) forming similar/dissimilar target pairs from an all-pairs interface similarity comparison.

Target pool. We start from a screen of 1299 cluster-representative, single-chain targets, each of which was subjected to a large-scale binder design run producing 50 candidate binders per target. A designed binder is counted as a *success* against its target if it satisfies the AlphaProteo success criteria Zambaldi et al. (2024). We use the per-target number of successful binders as a model-difficulty proxy and keep every target with at least five successes, yielding 394 targets.

Interface characterization. For each retained target we keep all of its successful designed binders. We deduplicate binders by structure with TM-align (Zhang & Skolnick, 2005), collapsing binders at TM-score (Zhang & Skolnick, 2004) ≥ 0.60 and keeping the lowest-i_pAE representative; this reduces 4794 passing complexes to 2758 unique target-binder complexes. For each complex we extract the two-sided binding interface consisting of all heavy atoms within 10Å across the target–binder boundary. Each target is thus represented by the set of interfaces of the distinct binders that succeed against it.

Interface similarity. We compare all pairs of binder interfaces with iAlign (Gao & Skolnick, 2010), using the IS-score as the interface similarity measure. The target-level similarity between two targets is the maximum IS-score over all pairs of their respective binder interfaces, i.e. two targets are considered similar if *any* pair of successful binders binds them through a shared interface geometry. Over the $75{,}466$ cross-target pairs the IS-score null distribution has median ≈ 0.20 and a thin high-similarity tail (p99.9 ≈ 0.51).

Pair selection. We first *deduplicate the target pool* with a union-find over targets: two targets are merged if their max-IS exceeds 0.40 *and* they share a PDB entry or structural cluster, or exceeds 0.45 for any pair. Each connected group is collapsed to a single representative—the target with the most successful binders, breaking ties by most unique binders, then lowest median i_pAE—so that no two benchmark targets describe essentially the same binding site; this removes 72 of the 394 targets, leaving 322. We additionally keep only compact, well-structured target chains, requiring each to fall below the 75th percentile of shape anisotropy, aspect ratio, normalized radius of gyration, and loop fraction.

We then form candidate pairs from the surviving representatives and discard candidate pairs if their two targets originate from the same PDB entry. Requiring that no target is reused across pairs, we

take the 10 highest-max-IS pairs as the *similar* set (SIM, max-IS $\approx$ 0.33–0.44) and 10 pairs from a low-similarity band as the *dissimilar* set (DIS, max-IS $\approx$ 0.16–0.27). This yields 20 pairs over 40 distinct targets. Every target carries a consensus set of hotspot residues (aggregated across its successful binders) and a binder length range, which together define the conditioning context C used at generation time.

C.2 Comparison to CROSS Benchmark

We found that a previous multi-target benchmark, CROSS, which also consisted of target pairs, was easily saturated by sampling with Complexa against a single target in each target pair. Table 5 reports the cross-reactivity success rates on the 100-entry CROSS benchmark. Each entry is a pair of target conformations (conf_0, conf_1) with a native binder; for each entry we draw a single design conditioned on one conformation and fold its binder sequence from scratch against both conformations (success per conformation: ipAE < 10 and pLDDT > 70; "both" requires the single sequence to pass on both conformations). Conditioning Complexa on either conformation already yields dual-conformation success on 96% of entries, far above the 7% reached by the CROSS-benchmark method (Chamaileon).

Table 5: Cross-reactivity results on the 100-entry CROSS benchmark. We report the fraction of entries whose designed binder succeeds against conf_0, against conf_1, and against both conformations (the cross-reactivity metric). "Target conf_0" and "Target conf_1" are single-target Complexa designs conditioned on the respective conformation.

Method	conf_0 success	conf_1 success	both success
Target conf_0	**0.98**	**0.96**	**0.96**
Target conf_1	0.96	**0.96**	**0.96**
Chamaileon	0.08	0.08	0.07

D Additional Experiments

Here we present various additional results and analyses to complement the main experiments.

D.1 ProteinMPNN Temperature Profiling

Before generating the ProteinMPNN sequence populations used to condition the model, we profiled how the sampled sequence distributions behave as a function of the ProteinMPNN sampling temperature τ. We ran ProteinMPNN over a grid of temperatures $\tau \in [0, 100]$, sampling a pool of sequences using our deterministic order at each temperature, and measured, for each temperature, the marginal diversity of the effective number of amino acids per position, 2^H, the mean per-position entropy H, the on-manifold fidelity of the samples, the ProteinMPNN score, and the designability of the sampled sequences via ESMFold. Figure 4 summarizes these four quantities.

As τ increases, marginal diversity rises monotonically, and both the ProteinMPNN score and the RMSD degrade, with a sharp designability cliff between $\tau = 0.7$ and $\tau = 1.0$. We therefore sampled our populations only up to $\tau = 5$ (red dashed line), which spans essentially the full accessible diversity range while excluding the far tail where the distribution collapses to near-uniform noise.

D.2 Preference Conditioning Behavior

We systematically probe how the model conditions on preferences during a single turn of optimization. Given two contexts A and B, we first sample designs against A, filtering for high-scoring designs. We generate sequence populations for these designs and then have the model design new samples against the cross-target B conditioned on the A preferences and look at how the model trades off B objectives against A objectives. As the temperature sweeps from 0 towards ∞ and the preference strength decreases, the model trades off binding towards A for binding towards B, as shown in Figure 5.

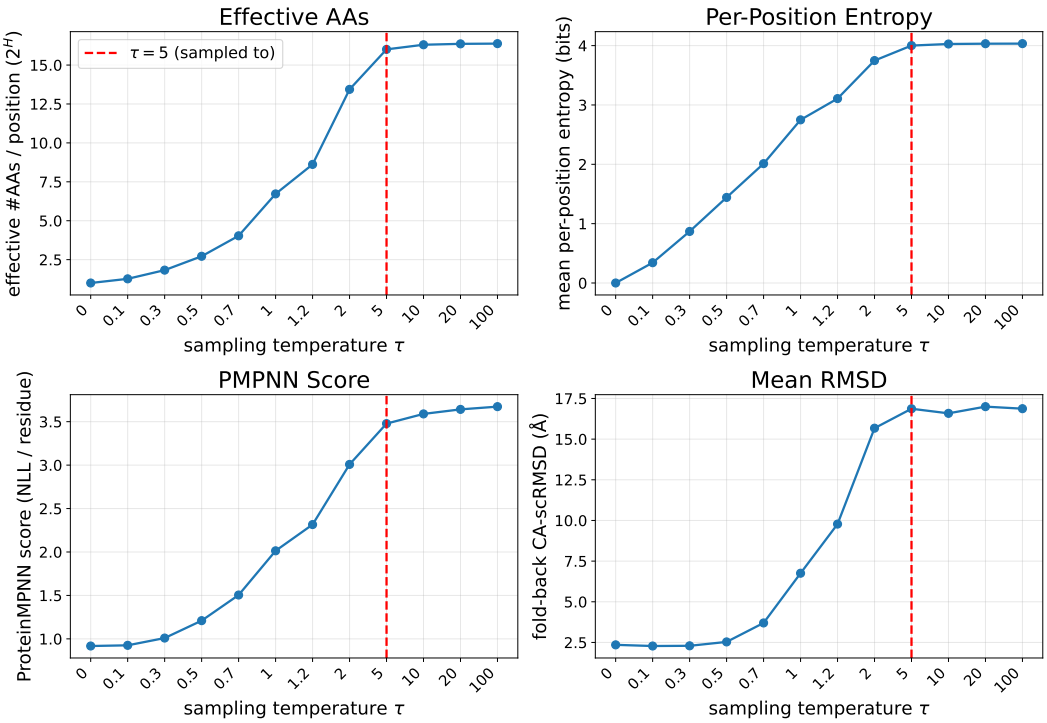

Figure 4: ProteinMPNN pool statistics as a function of the sampling temperature τ. **Effective AAs**: effective number of amino acids per position (2^H; uniform-over-20 $\rightarrow$ 20, collapsed $\rightarrow$ 1). **Per-Position Entropy**: mean per-position entropy H in bits. **PMPNN Score**: ProteinMPNN negative log-likelihood per residue. **Mean RMSD**: ESMFold fold-back CA-scRMSD to the native binder backbone. Diversity rises with τ while sequence fidelity and fold-back designability degrade; the red dashed line marks $\tau = 5$, the maximum temperature used when generating the conditioning populations during training.

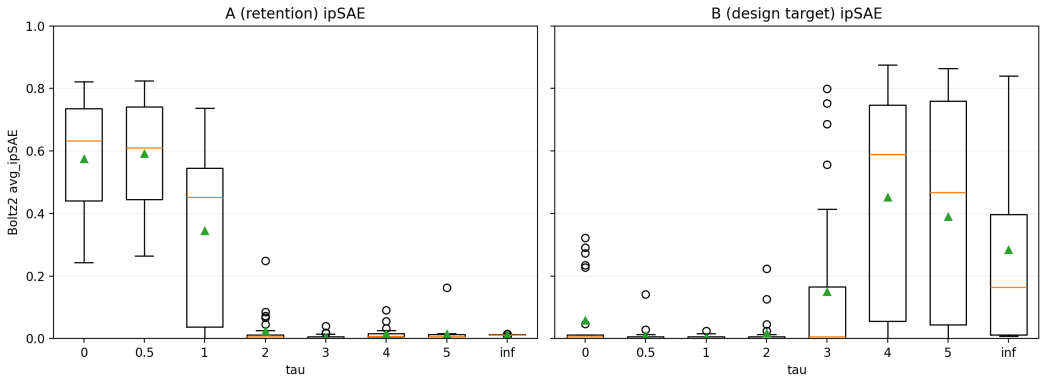

Figure 5: Interface ipSAE (Boltz-2) of designs generated against the cross-target B, conditioned on a preference population sampled from the original A designs at temperature τ. As τ increases, A-retention (left) degrades while B-binding (right) improves, illustrating the trade-off between preserving the original preference and satisfying the new design target.

D.3 Folding Model Seeding

At the lowest temperature, where the model conditions on a single sequence and therefore essentially performs folding, we found that the structure it generates does not generally agree with the predictions from a folding model. We therefore seed the model by starting it from an intermediately

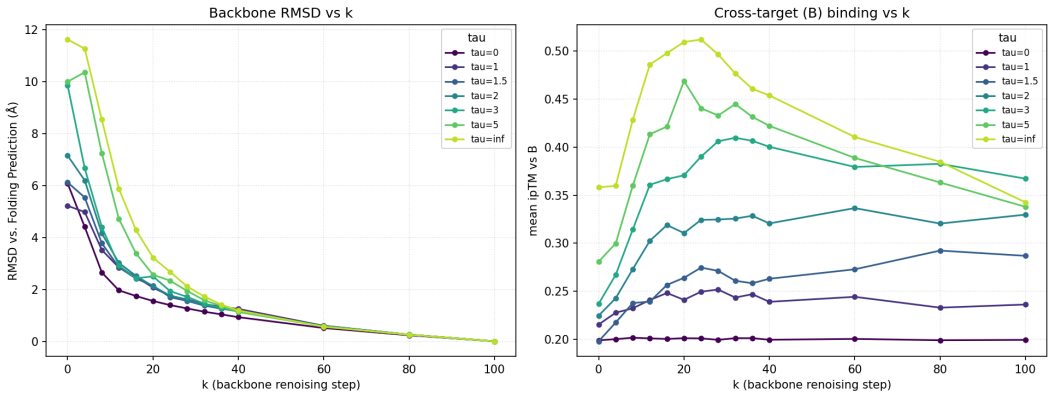

Figure 6: Effect of the backbone renoising step k used to seed generation from an external folding model's prediction against the cross-target B. **Left**: RMSD between the generated backbone and the folding prediction it was seeded from, as a function of k, for pool temperatures $\tau \in \{0, 1, 1.5, 2, 3, 5, \infty\}$. Smaller k means more noise was added to the folding-model seed before denoising resumed (k=0: generation from scratch; larger k: closer to the folding prediction). Even at $\tau = 0$ (folding), the model does not agree with the folding prediction at $k = 0$, showing it is a poor folder from scratch; agreement improves progressively with k, falling below 2 Å by $k \approx 20$. **Right**: mean ipTM against the cross-target B as a function of k, for the same temperatures, peaking around the same $k \approx 20$. Pooled over 50 initial A-binders, 5 samples each.

noised folding model prediction. This is accomplished by renoising the folding model's prediction and giving that as a starting point to the design model. We sweep intermediate noise levels and find a sweet spot as shown in Figure 6.

Concretely, we start with two protein targets A and B. For each of 50 initial A-binders we take an external folding model's prediction of the structure adopted by the PMPNN-designed sequence against B, and use it as the seed for our generative model. We *renoise* that folding prediction up to an intermediate step k of the generative model's sampling procedure and then denoise from there, conditioned on B and a pool temperature τ; k is a point along the generative process, not a noise-level, so a smaller k means the seed has been noised to an earlier step of the generative process, and therefore has had more noise applied to it. Figure 6 (left) shows that even at $\tau = 0$ the model's output does not recover the folding prediction at $k = 0$, but agreement improves progressively as we renoise from later steps, dropping below 2 Å RMSD to the folding prediction by around $k = 20$. The right panel shows that this same renoising step is where cross-target binding to B peaks across temperatures, so $k \approx 20$ gives both a backbone close to a plausible B fold and, at higher τ, retains enough flexibility for the pool conditioning to still trade off against the A preference.

D.4 Temperature Schedule

The PMPNN temperature τ also controls the strength of the preferences it represents. This influences `Ctrl-P`'s exploratory behavior. At each round, the active model conditions on the preferences of the other models. A low τ creates a concentrated preference around the other models' designs (exploitation) while a high τ gives a loose preference allowing the active model to prioritize its own target and explore more. Rather than fixing τ during the negotiation, we anneal it per candidate off the trajectory's current score: trajectories far from a dual success stay hot (weak preference, more exploration), and cool down as scores against both scenarios climb. Figure 7 shows the schedule that performed best in our sweep and used by `Ctrl-P`. Table 6 ablates this score-conditioned schedule against the two strongest *fixed*-temperature configurations using the same setup and success criterion as Table 1. The annealed schedule solves 10/10 SIM pairs versus 9/10 for either fixed configuration but solves substantially more DIS pairs (8/10 vs. 2–3/10), showing that adapting the preference strength to each trajectory's progress is beneficial.

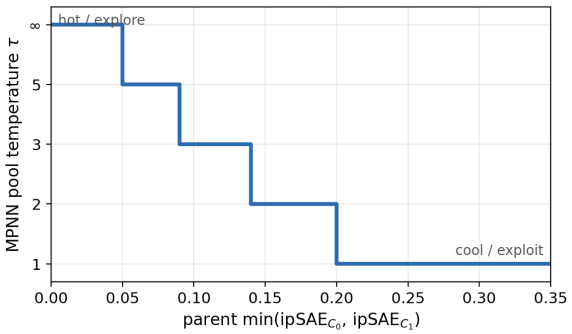

Figure 7: The score-conditioned temperature schedule used by `Ctrl-P` in the multi-target benchmark. The plot shows the MPNN pool temperature τ applied to a candidate's redesign as a step function of its parent trajectory's current min-ipSAE against both targets. Trajectories far from success stay hot ($\tau = \infty$, unconditional pool, maximal exploration) and cool in discrete steps ($\tau \in \{5, 3, 2, 1\}$) as the parent's min-ipSAE climbs, strengthening the preference for the parent's own sequence as the trajectory approaches a dual success.

Table 6: Ablating the score-conditioned temperature-annealing schedule used by `Ctrl-P` (Table 1) against the two strongest fixed-temperature configurations on the multi-target benchmark (same setup and success criterion as Table 1).

	DIS		SIM	
Configuration	solved / 10	avg. unique / task	solved / 10	avg. unique / task
Fixed, unconditional ($\tau = \infty$)	2	0.5	9	6.5
Fixed $\tau = 3$	3	0.8	9	6.2
Temperature schedule (ours)	8	1.5	10	5.3

D.5 Metric Calibration

During the search, candidates are selected by a $\min$ over per-scenario metrics. We normalize these so that heterogeneous metrics, like motif C_α-RMSD (Å), cross-exclusivity C_α-RMSD (Å), and ligand interface-PAE, can be compared on a common $[0, 1]$ "goodness" axis. For the multi-target benchmark we just use the raw metrics since we use the ipSAE for both scenarios.

Each metric v is mapped through a sigmoid gate

$$g(v) = \frac{1}{1 + \exp\left(-\frac{\ln 9}{s}\, x\right)}, \qquad x = \begin{cases} t - v & \text{lower-is-better,} \\ v - t & \text{higher-is-better,} \end{cases} \tag{6}$$

parameterized by a threshold t and a scale s. The threshold is that metric's established pass/fail criterion ($t = 1.0$ Å for motif RMSD, $t = 2.0$ for ligand interface-PAE), and the scale is the distance past the threshold at which a candidate is treated as clearly good or bad; the $\ln 9$ factor fixes $g = 0.5$ exactly at the threshold and $g \approx 0.9$ one scale beyond it, with the sigmoid never saturating so that a candidate already past its gate still has a monotone incentive to improve. Figure 8 overlays each calibrated gate on the empirical distribution of its metric, pooled over all candidate evaluations from the 12-task motif-switching benchmark.

D.6 Sequence Coordination

To illustrate the difficulty of modifying the inference process of protein generative models, we present some early experiments trying to guide Proteina-Complexa towards a given sequence. In our experiment, we show that, while it is easy to recover the desired sequence, the generated structure often does not correspond to it. Part of the reason is that in Complexa the backbone denoises much faster than the local latents which encode the sequence. Thus, it is difficult for the sequence to guide the structure. Matching the local latent schedule to the backbone schedule improves structure recovery but worsens binding quality in this experiment.

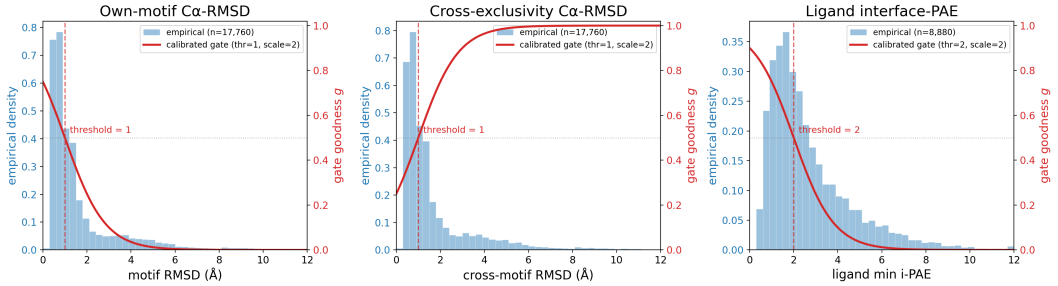

Figure 8: Calibrated selection gates (red) overlaid on the empirical distribution (blue) of each selector metric, pooled over all candidate evaluations from the 12-task motif-switching benchmark. **Left**: own-motif C_α-RMSD (lower-is-better, threshold 1.0 Å). **Center**: cross-exclusivity C_α-RMSD (higher-is-better, threshold 1.0 Å), rewarding designs that place the other motif far from its reference. **Right**: ligand interface-PAE (lower-is-better, threshold 2.0). Each gate is a sigmoid $g(v)$ equal to 0.5 at its threshold (dashed line) and ≈ 0.9 one scale ($s = 2.0$) past it; the left axis shows the empirical density and the right axis the gate goodness $g \in [0, 1]$.

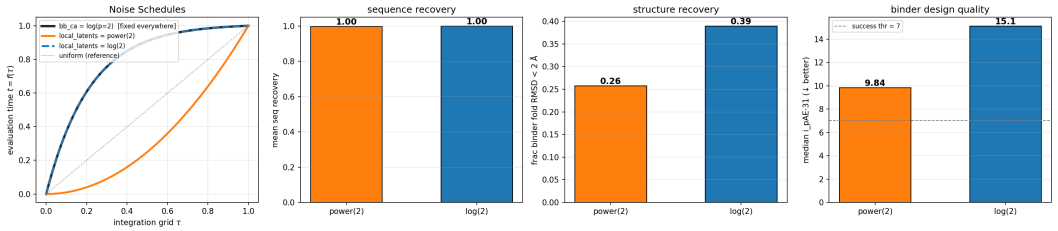

Figure 9: Effect of the local-latent noise schedule on sequence-guided recovery versus unconditioned design quality. **Left**: the two candidate local-latent schedules. **Center-left, center-right**: sequence recovery is near 1.0 under both schedules; switching from `power(2)` to `log(2)` improves structure recovery (0.26 $\rightarrow$ 0.39) in the sequence-guided recovery task. **Right**: the same schedule change degrades unconditioned binder design quality, raising the median interface PAE ($i_pAE \times 31$: 9.8 $\rightarrow$ 15.1).

Concretely, we trained a sequence prediction head to predict the clean sequence from a noisy intermediate state. We then added the gradient of this sequence head to Complexa's predicted score, in order to guide the inference process towards a given sequence. We then generated 8 binders to each individual target from our multi-target benchmark. For each binder, we generated 8 new samples conditioned on the original binder's sequence using our sequence guidance. We compared two schedules for the local latents: `power(2)` and `log(2)`. In the plotted experiment, `log(2)` matches the backbone schedule. We looked at (1) how well did the new samples recover the sequence of the original binder, which it was guided towards, and (2) how well did it recover the structure. To assess the effect on binding quality, we separately generated binders using the two schedules and measured median ipAE with ESMFold2.

Figure 9 shows that switching the local-latent schedule to track the backbone schedule increases structure recovery (0.26 $\rightarrow$ 0.39) with sequence recovery already saturated near 1.0 under either schedule, but substantially degrades unconditioned design quality (median $i_pAE \times 31$: 9.8 $\rightarrow$ 15.1).

D.7 Round-by-Round Search Dynamics

We take a closer look at the round-by-round mechanics of `Ctrl-P` on the full multi-target benchmark, split by SIM and DIS pairs.

DIS successes come later. Figure 10 (left) shows the cumulative fraction of trajectories that have reached a first dual success by round r. SIM successes accrue quickly and saturate by round 3–4;

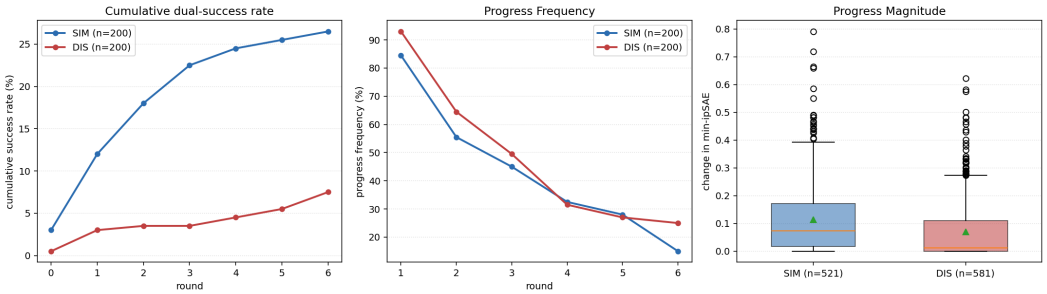

Figure 10: Per-round search dynamics of `Ctrl-P` on the multi-target benchmark. **Left:** cumulative fraction of trajectories that have reached dual success (ipSAE > 0.5 and pLDDT > 0.9 for both targets) by round r. SIM plateaus early near a much higher ceiling; DIS accrues successes more slowly and keeps gaining through the final round. **Center:** fraction of trajectories, per round, where a sampled child beats the carried-over incumbent. **Right:** distribution of the per-step change in $\min(\text{ipSAE}_{C_0}, \text{ipSAE}_{C_1})$ among progress events. DIS progress steps are systematically smaller than SIM's, explaining why DIS successes take longer to accumulate.

DIS successes accrue more slowly and are still increasing at the final round, with over half of all eventual DIS successes only appearing in the last two rounds.

Progress frequency is similar. Figure 10 (center) shows the fraction of trajectories, per round, where a sampled child beats the carried-over incumbent. This is nearly identical between SIM and DIS at every round, both dropping from $\sim$90% at round 1 to $\sim$20–25% by round 6. So DIS trajectories are not stalling more often than SIM's.

Progress magnitude is smaller for DIS. Figure 10 (right) shows the distribution of the per-step change in $\min(\text{ipSAE}_{C_0}, \text{ipSAE}_{C_1})$ conditioned on progress having been made. DIS progress steps are systematically smaller than SIM's (median 0.012 vs. 0.074), with a much heavier concentration of near-zero, marginal improvements. This is the primary driver of DIS's harder, slower-converging dynamics: DIS trajectories improve about as often as SIM's, but each improvement is a smaller nudge, so it takes more rounds to accumulate enough gain to cross the dual-success threshold.

D.8 Inverse Folding Proposals

Our two preference alignment algorithms can be used with any model that can condition on preferences. Since we already seed `Ctrl-P` with a folding model prediction, a natural comparison is therefore to tied PMPNN decoding (Dauparas et al., 2022), which combines the PMPNN logits from two structures. By averaging the logits it can be seen as merging preferences in a similar way to `Ctrl-P`. The preferences it merges are also related to `Ctrl-P`, since we use PMPNN to construct the sequence populations that represent preferences. Concretely, when sampling a new candidate x at each round, we fold the current sequence against both targets and redesign it from a joint decode over both folds. Given targets A and B, let $z_{A,i}$ and $z_{B,i}$ be the PMPNN logits for position i. We then sample from

$$\text{softmax}\left(\frac{\beta_A z_{A,i} + \beta_B z_{B,i}}{2T}\right), \tag{7}$$

where β_A and β_B control the weight on each target. By default we use $\beta_A = \beta_B = 1$.

For the multi-target benchmark, resampling the sequence with PMPNN tends to improve the binder. For the SwitchCraft benchmark, we template the folding model with the desired motifs to drive the designs to satisfy the motif, and we use LigandMPNN instead of PMPNN in order to condition on the target ligand as well.

On multi-target binder design (Table 7), `Ctrl-P` solves more DIS pairs than using an inverse folding model (8 vs. 6/10). The two are comparable on SIM (10/10 pairs each, 5.3 vs. 6.0 unique successes per task). On the motif-switching benchmark (Table 8), the inverse folding model trails both `Ctrl-P` and SwitchCraft (4/12 tasks, 5/240 unique successes, 2.1%, vs. 8/12, 29/360, 8.1% for `Ctrl-P`).

Table 7: Inverse folding as an alternative proposal to `Ctrl-P` on the multi-target benchmark (same setup as Table 1).

	DIS		SIM	
Proposal	solved / 10	avg. unique	solved / 10	avg. unique
Tied-MPNN	6	1.4	10	6.0
`Ctrl-P`	8	1.5	10	5.3

Table 8: Inverse folding as an alternative proposal to `Ctrl-P` on the motif-switching benchmark (same setup as Table 3).

Proposal	Tasks solved	Unique successes	Success rate	GPU-hours
Tied-MPNN	4/12	5/240	2.1%	96.1
`Ctrl-P`	8/12	29/360	8.1%	160.2

D.9 Population Search

As an alternative to negotiation, we also evaluated `Ctrl-P` with a population search algorithm. Instead of independently refining trajectories, population search keeps one growing candidate pool and a global selector that concentrates each round's budget on the best-scoring designs so far (Fig. 11).

Consider the same J scenarios and objectives as in our default stochastic negotiation. Initialize a pool of candidates X with samples designed against single scenarios at random. At round k, given candidate pool X:

1. Pick M best candidates out of X, and sample N new candidates per selected one using $p_\theta(x|C, S, \tau)$, designing against the worst objective and conditioning on the preferences of the rest.
2. Add all new candidates to the pool X.

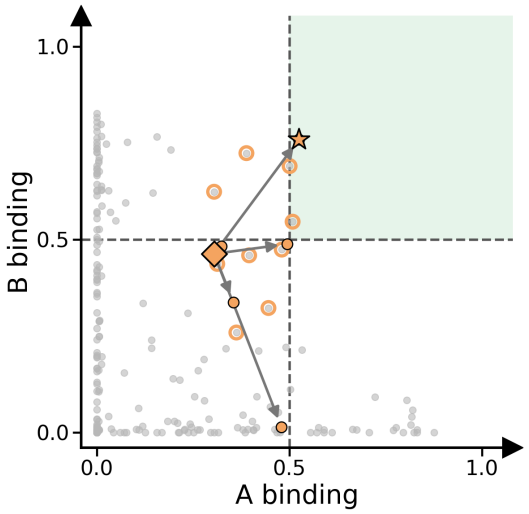

Figure 11: Illustration of population search, in the space of A vs. B binding (higher is better along each arrowed axis); the shaded quadrant is the dual-success region. The proposal kernel grows a shared pool (gray dots); at a given step the M best members (orange rings) become parents, and expanding the chosen parent (orange diamond) yields N children (orange dots), one of which reaches a fresh dual success (orange star). Compared with negotiation (Figure 2), population search concentrates compute on the current frontier of a common pool rather than refining independent trajectories.

On the multi-target benchmark (Table 9), population finds more unique successes on SIM (9.8 vs. 5.3 on average), but does not increase the number of solved DIS tasks, getting slightly fewer successes.

On the motif-switching benchmark (Table 10), population does not improve on negotiation: it solves fewer tasks (4/12 vs. 8/12) and finds fewer unique successes per task on average (0.4 vs. 2.4).

Table 9: Negotiation vs. population search on the multi-target benchmark (same setup and criterion as Table 1).

	DIS		SIM	
Search	solved / 10	avg. unique / task	solved / 10	avg. unique / task
Negotiation	8	1.5	10	5.3
Population	8	1.4	10	9.8

Table 10: Negotiation vs. population search on the motif-switching benchmark (same setup and criterion as Table 3).

Search	solved / 12	avg. unique
Negotiation	8	2.4
Population	4	0.4

D.10 Viral Case Study: Predicted Interfaces

Figure 12 complements the preference-pooling schematic in Sec. 4.3 with target-side views of the separate predicted interfaces.

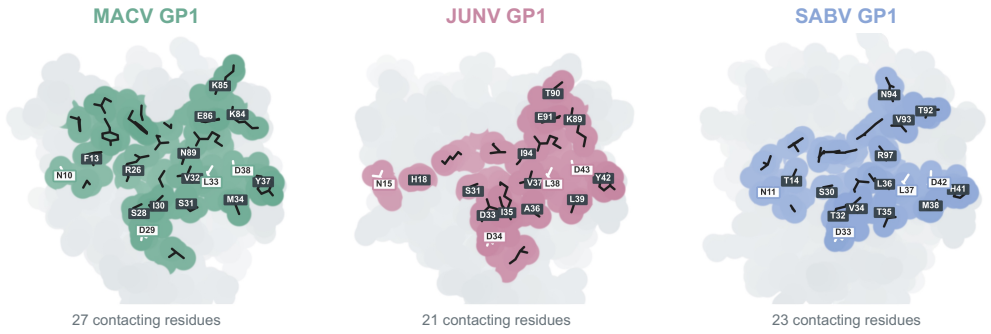

Figure 12: **Viral Target Contact Points.** Target interface views from a designed binder's separate AF2-predicted complexes with MACV, JUNV, and SABV GP1. The interface views highlight 27, 21, and 23 contacting target residues, respectively. White sticks and light labels indicate matching amino-acid identities at corresponding contacted positions across all three targets; dark sticks and labels indicate differing identities. Target colors identify the three contexts and do not encode electrostatic potential. These are predicted interfaces, not experimentally established binding sites.