FROM CHAINS TO TREES: EFFICIENT EXPLORATION AND CREDIT ASSIGNMENT IN PROTEIN DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models provide strong generative priors for conditional protein design, but steering them toward downstream objectives remains challenging. Reinforcement learning (RL) post-training is crucial for aligning generative models with specific targets, but its prohibitive computational cost remains a major barrier to widespread adoption. In protein design, this leads to two problems: (1) rollout budgets are spent repeatedly resampling similar prefixes; (2) terminal rewards provide sparse supervision for long denoising trajectories. We propose TSG-Diff, an RL framework that organizes reverse diffusion as a branching tree over intermediate checkpoint nodes. Improves training efficiency by recasting the denoising process as a search tree. Starting from a small set of initial rollouts, TSG-Diff expands selected checkpoints to diverse continuations while reusing partial trajectories. Terminal rewards on final designs are converted into group-relative leaf advantages and then propagated through the tree to produce segment-level credit for intermediate denoising decisions. We instantiate TSG-Diff for protein design with a composite reward combining binding affinity, interface confidence, and structural regularization. Our approach provides more informative supervision under a fixed rollout budget, and empirical results show improved optimization of binding-related objectives while preserving structural plausibility and naturalness. The code can be found at this anonymous link: https://anonymous.4open.science/r/TSG-Diff-F223.

1 INTRODUCTION

Diffusion generative models have emerged as a strong foundation for protein design Watson et al. (2023); Yim et al. (2023); Ingraham et al. (2023). In applications such as peptide binder design, our design objectives are not only to match the pretraining distribution, but also to optimize properties including binding affinity, interface quality, and structural plausibility Bennett et al. (2023); Vázquez Torres et al. (2024); Evans et al. (2021). Reinforcement learning (RL), a technique widely used in large language models (LLMs) post-training for enhancing reasoning capabilities and aligning with human preferences (Ouyang et al., 2022b; Shao et al., 2024), is a natural tool for this setting Black et al. (2023); Uehara et al. (2024).

However, applying RL to diffusion models is not straightforward Black et al. (2023); Prabhudesai et al. (2023); Uehara et al. (2024). A diffusion policy makes decisions throughout a long reverse-denoising trajectory, whereas the task reward is usually observed only for the final denoised structure. A terminal reward can tell the policy whether a completed design is good or bad, but it provides limited information about which intermediate denoising decisions led to that outcome. This sparse supervision is especially problematic in protein design, where the search space is extremely large and high-reward designs may lie in narrow regions that are rarely reached by the pretrained model.

Existing group-based RL methods for denoising generative models typically sample several independent rollouts for the same conditioning input and compare them only at the trajectory level Shao et al. (2024); Liu et al. (2025). Although such comparisons provide a useful relative baseline, they do not exploit the fact that many trajectories share informative partial denoising prefixes. Under a fixed rollout budget, repeatedly sampling full trajectories from scratch is therefore inefficient: computation is spent regenerating similar early denoising steps instead of exploring alternative continuations from promising intermediate states. A closely related recent line of work has also explored tree-

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.

structured GRPO for diffusion and flow post-training in visual domains Ding & Ye (2025); our focus here is diffusion fine-tuning for protein design with biophysical rewards.

This paper addresses these issues by viewing reverse diffusion as a tree-search problem. As shown in Figure 1, each node in the tree stores an intermediate checkpoint latent at a selected timestep in the diffusion schedule, each edge is a denoising segment between two checkpoints, and each leaf is a fully denoised design. Starting from a small set of initial trajectories, we repeatedly expand selected internal checkpoint nodes to generate new suffixes. This branching representation lets the algorithm reuse partial trajectories, allocate sampling budget to promising regions of the search space, and gather a richer set of competing continuations for the same protein-design condition.

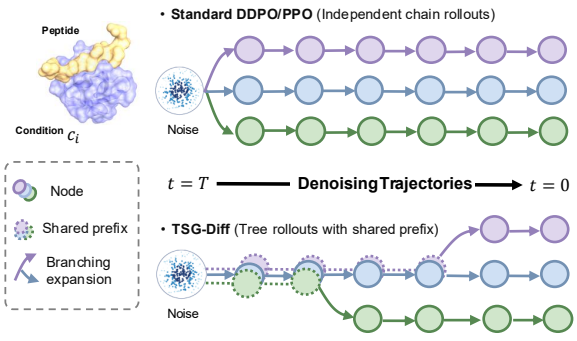


Figure 1: Overview of tree-structured rollout generation for diffusion models.

To optimize the diffusion policy on this tree, we combine the group-relative idea of GRPO Shao et al. (2024) with tree-search credit assignment. Final designs are scored by a composite reward that balances binding affinity, interface confidence, and structural regularization. We then normalize rewards both within each tree and across trees for the same conditioning input to obtain leaf-level relative advantages. To provide denser supervision than terminal rewards alone, we propagate these leaf advantages back through the tree: each internal checkpoint node receives a subtree value, and the difference between parent and child values defines a segment-level advantage for the denoising steps on that edge. The resulting update remains PPO-style at the step level Schulman et al. (2017), but it is informed by tree-level competition among alternative continuations. We refer to this overall procedure as Tree-search GRPO diffusion policy optimization (TSG-Diff). Our focus is diffusion fine-tuning for protein design, where the objective is not only to maximize task reward but also to preserve the structural prior encoded by the pretrained model Uehara et al. (2024). We instantiate TSG-Diff by fine-tuning BoltzGen (Stark et al., 2025), an all-atom diffusion model for protein and peptide binder design, using task rewards defined on generated protein complexes. TSG-Diff is well-suited to this setting because it improves exploration without discarding the naturalness and plausibility inherited from the base diffusion model.

Our contributions are fourfold:

- We propose Tree-search GRPO (TSG-Diff), an RL post-training framework that turns reverse diffusion into a checkpoint tree and reuses intermediate denoising states during rollout generation.
- We introduce a tree-based credit assignment scheme that combines group-relative leaf normalization with checkpoint-segment credit propagation, producing denser supervision than trajectory-level terminal rewards alone.
- We instantiate TSG-Diff for protein design with a composite reward over binding affinity, interface confidence, and structural regularization.
- We evaluate the method on protein-design tasks, including peptide binder design, to study reward optimization, structural quality, and diversity under a fixed rollout budget.

2 PRELIMINARIES

Let $\mathbf{x}$ denote a clean protein structure and let $\mathbf{c}$ denote the conditioning context, such as the receptor structure, design mask, and other geometric constraints. A conditional diffusion process gradually destroys $\mathbf{x}$ over timestep $t \in \{0, \dots, T\}$ by mixing data with noise according to a schedule (α_t, σ_t) as $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where the schedule is chosen so that $\mathbf{z}_0 = \mathbf{x}$ and $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For a fixed condition $\mathbf{c}$, sampling proceeds by initializing $\mathbf{z}_T$ from Gaussian noise and iteratively stepping to a lower noise level $s < t$ via $\mathbf{z}_s = \alpha_s \hat{\mathbf{x}} + \sigma_s \hat{\boldsymbol{\epsilon}}$, where $\hat{\epsilon}$ is the model's condition-dependent noise prediction and $\hat{\mathbf{x}}$ is recovered by inverting the forward equation. Diffusion models can be formulated as Markov decision processes (MDPs), where the reverse denoising procedure is interpreted as a sequence of decisions and the policy induces the full denoising trajectory. Formally,

we define an MDP as the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P_t^{tra} is a transition dynamic mapping: $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ mapping state-action pairs to a scalar signal r_t, and $\rho_0 \sim \Delta(\mathcal{S})$ is the initial state. A stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps each state $s \in \mathcal{S}$ to a distribution over actions.

Denoising as a Markov Decision Process. Following DDPO Black et al. (2023), we cast the denoising process of a diffusion model as an MDP:

$$\begin{aligned} &\mathbf{s}_t \triangleq (\mathbf{c}, t, \mathbf{z}_t), \quad \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) \triangleq p_\theta(\mathbf{z}_{t-1} \mid \mathbf{z}_t, \mathbf{c}, t), \quad P(\mathbf{s}_{t-1} \mid \mathbf{s}_t, \mathbf{a}_t) \triangleq \left(\delta_{\mathbf{c}}, \delta_{t-1}, \delta_{\mathbf{z}_{t-1}}\right) \\ &\mathbf{a}_t \triangleq \mathbf{z}_{t-1}, \quad r(\mathbf{s}_t, \mathbf{a}_t) \triangleq \begin{cases} r(\hat{\mathbf{x}_t}, \mathbf{c}), & \text{if } t = 1, \\ 0, & \text{otherwise}, \end{cases} \quad \rho_T(\mathbf{s}_T) \triangleq \left(p_{\text{data}}(\mathbf{c}), \delta_T, p_T(\mathbf{z}_T)\right). \end{aligned} \tag{1}$$

Here, $\mathbf{c}$ denotes the conditioning input and $\mathbf{z}_t$ is the diffusion latent at timestep t; the symbol $\mathbf{x}$ is reserved for clean or generated protein structures. The policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$ models the conditional reverse transition from $\mathbf{z}_t$ to $\mathbf{z}_{t-1}$ under $\mathbf{c}$. The notation δ_y refers to the Dirac delta distribution concentrated at y. Each trajectory starts from $\mathbf{z}_T \sim p_T$ and terminates at $\mathbf{z}_0$, which is decoded into a final structure $\hat{\mathbf{x}} = \text{dec}(\mathbf{z}_0)$ before receiving reward $r(\hat{\mathbf{x}}, \mathbf{c})$.

Optimization objective. To prevent the fine-tuned distribution from drifting too far from the pre-trained model p^{ref}, the following KL-regularized objective was introduced over terminal structures for each condition $\mathbf{c}$:

$$\max_{q(\cdot|\mathbf{c}) \in \Delta(\mathcal{X})} \mathbb{E}_{\mathbf{x} \sim q(\cdot|\mathbf{c})}[r(\mathbf{x}, \mathbf{c})] - \alpha D_{\text{KL}}[q(\cdot \mid \mathbf{c}) \| p^{\text{ref}}(\cdot \mid \mathbf{c})], \tag{2}$$

where $\mathcal{X}$ denotes the space of valid protein structures, and $\Delta(\mathcal{X})$ is the probability simplex over this space. Besides, p^{ref} is the pre-trained distribution and $\alpha > 0$ controls the strength of the regularization. The first term maximizes expected reward, while the KL term penalizes deviation from p^{ref}. The optimizer is the reward-tilted distribution $p_r^*(\mathbf{x} \mid \mathbf{c}) := \frac{1}{Z(\mathbf{c})} e^{r(\mathbf{x},\mathbf{c})/\alpha} p^{\text{ref}}(\mathbf{x} \mid \mathbf{c})$, where $Z(\mathbf{c})$ is the partition function (normalizing constant) defined as: $Z(\mathbf{c}) = \int_{\mathcal{X}} e^{r(\mathbf{x}',\mathbf{c})/\alpha} p^{\text{ref}}(\mathbf{x}' \mid \mathbf{c})\, d\mathbf{x}'$. Smaller α emphasizes reward maximization, whereas larger α keeps p_r closer to p^{ref}.

3 METHOD

To achieve a more effective allocation of the rollout budget and address the sparse supervision challenges in diffusion RL fine-tuning, we propose Tree-search Group Relative Policy Optimization (TSG-Diff). By sharing partial prefixes between rollouts, tree search method could obtain more terminal candidates under the same inference budget. Based on the tree structure, we further derive supervision signals of intermediate checkpoint segments from leaf-level rewards. Figure 2 presents an overview of our proposed TSG-Diff. In the following sections, we detail the tree-structured rollout procedure (§ 3.1), leaf-level group-relative advantages (§ 3.2), checkpoint-segment credit propagation (§ 3.3), the composite reward (§ 3.4), and the overall training algorithm (§ 3.5).

Problem statement. TSG-Diff organizes the reverse diffusion process as a tree, where each node represents a checkpoint latent state and each edge corresponds to a denoising segment between two checkpoint states $\mathcal{K} = \{\kappa_0, \kappa_1, \ldots, \kappa_Q\}$ satisfying $0 = \kappa_0 < \kappa_1 < \cdots < \kappa_Q = T$, where $\mathcal{K}$ are *time indices* of checkpoint states. A tree node v stores a checkpoint tuple $\xi_v \triangleq (\mathbf{c}, \kappa(v), \mathbf{z}_v), \kappa(v) \in \mathcal{K}$ where $\mathbf{z}_v$ is the latent available at time $\kappa(v)$. Each directed edge $u \to v$ satisfies $\kappa(u) > \kappa(v)$ and represents the denoising segment that transforms $\mathbf{z}_u$ into $\mathbf{z}_v$ over steps $t \in (\kappa(v), \kappa(u)]$. Root nodes are initialized at $\kappa_Q = T$, leaf nodes are at $\kappa_0 = 0$, and leaves decode to final protein structures that receive terminal rewards. In this work, our goal is to fine-tune a pretrained diffusion policy π^{ref} to obtain an improved policy π^*. Given a conditioning input $\mathbf{c}$, the policy induces a reverse-diffusion trajectory over latent checkpoint states, yielding a terminal distribution over denoised structures.

3.1 TREE-STRUCTURED ROLLOUT GENERATION FOR DIFFUSION MODELS

In contrast to the standard chain-view formulation, we organize rollouts as a collection of *checkpoint trees*, where different trajectories share denoising prefixes and branch into alternative suf-

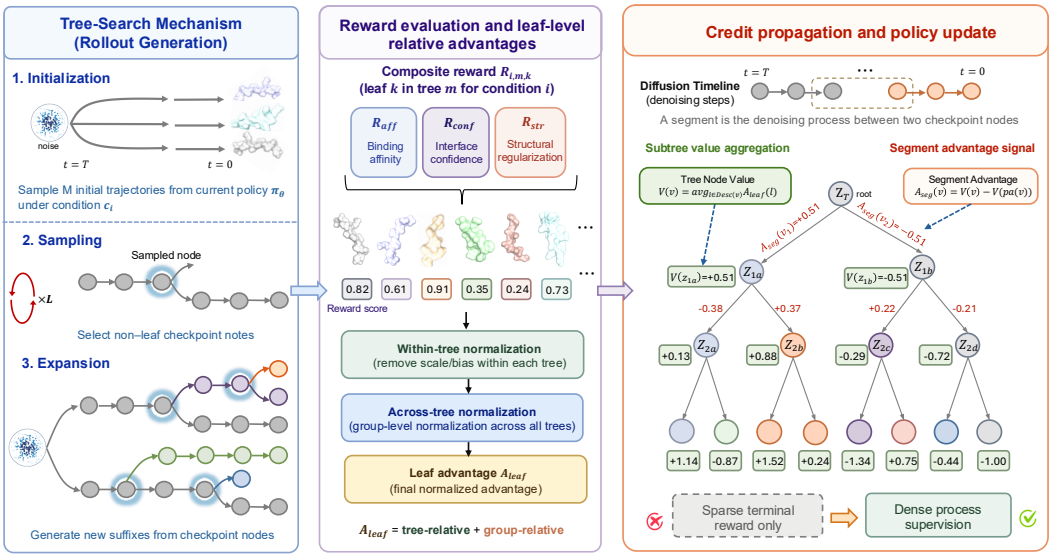


Figure 2: Overview of TSG-Diff. **Left:** M reverse-diffusion trajectories are initialized, non-leaf checkpoint nodes are sampled, and suffixes are expanded from them, repeated L times to form checkpoint trees that reuse denoising prefixes. **Middle:** each leaf receives a composite reward over binding affinity, interface confidence, and structural regularization; within- and across-tree normalization yield the leaf advantage A_{leaf}. **Right:** subtree-value aggregation (green) and parent-to-child differences (orange) turn A_{leaf} into segment advantages A_{seg}, and each denoising step is supervised by a convex combination of the two—dense process supervision from a sparse terminal reward under a fixed rollout budget.

fixes under the same conditioning input. A central challenge is that online RL requires repeatedly generating rollouts, computing rewards, and updating the policy under tight throughput constraints (Ouyang et al., 2022b; Shao et al., 2024). Search-based methods such as MCTS expand paths sequentially (Yao et al., 2023; Snell et al., 2024), but these sequential dependencies batch poorly on diffusion inference and severely bottleneck online RL efficiency (Schulman et al., 2017; Chen et al., 2024). To address this, we adopt an initialize-then-expand strategy similar to (Ji et al., 2025), in which multiple chains are first initialized in parallel and nodes are subsequently sampled for expansion. The rollout procedure proceeds as follows:

1. **Initialization.** For each conditioning input $\mathbf{c}_i$, we sample M independent reverse-diffusion trajectories $\{\tau_{i,m}\}_{m=1}^{M}$ from the current policy $\pi_\theta(\cdot \mid \mathbf{c}_i)$. Each trajectory yields checkpoint tuples $\{\xi_{i,m}^{(q)} = (\mathbf{c}_i, \kappa_q, \mathbf{z}_{i,m}^{(\kappa_q)})\}_{q=0}^{Q}$ that initialize one search tree $\mathcal{T}_{i,m}$.
2. **Sampling.** At each expansion round $\ell = \{1, \cdots, L\}$ of tree $\mathcal{T}_{i,m}$, we define the expansion set $\mathcal{V}_{i,m}^{(\ell)} \subset \text{Non-Leaf}(\mathcal{T}_{i,m})$ containing $|\mathcal{V}_{i,m}^{(\ell)}| = N$ selected non-leaf nodes.
3. **Expansion.** To branch out from each selected node $v \in \mathcal{V}_{i,m}^{(\ell)}$ at checkpoint time $\kappa(v)$, we reuse the prefix path $\tau_{T:\kappa(v)}$ from the root T down to v, and resume reverse diffusion from the stored latent $\mathbf{z}_v$ to generate a new suffix trajectory $\tau'_{\kappa(v):0} \sim \pi_\theta(\cdot \mid \xi_v)$, which is then attached to $\mathcal{T}_{i,m}$.

After L rounds of sampling and expansion, each initial rollout becomes a branching tree of alternative continuations. Since each round adds N new continuations, one tree contains $1 + LN$ rollout paths, including the original rollout. Thus, each conditioning input yields $M(1 + LN)$ candidate rollouts before leaf selection. We then evaluate the selected terminal leaves with the reward function and use them to update the policy π_θ. Appendix D provides a more detailed analysis of the computational budget for tree-search rollouts.

3.2 Leaf-level group-relative advantages

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) provides a critic-free alternative to PPO Schulman et al. (2017) by replacing the learned value baseline with within-group reward com-

parisons. For each conditioning input $\mathbf{c}_i$, GRPO samples a group of candidate trajectories $\{\tau_{i,g}\}_{g=1}^{G}$ from the current policy $\pi_{\theta_{\text{old}}}$.

After computing the terminal reward for all sampled leaves under condition $\mathbf{c}_i$, we convert them into relative advantages. To match the tree-search rollout, we use a two-level normalization: one within each tree and one across all trees under the same condition:

$$A_{\text{leaf}}(u_{i,m,k}) = \frac{r_{i,m,k} - \mu_{i,m}}{\sigma_{i,m} + \varepsilon_{\text{norm}}} + \frac{r_{i,m,k} - \mu_i}{\sigma_i + \varepsilon_{\text{norm}}}. \tag{3}$$

Here, $(\mu_{i,m}, \sigma_{i,m})$ represent the empirical mean and standard deviation of the rewards among the K leaves within the m-th tree, while (μ_i, σ_i) represent the global statistics aggregated across all M trees for condition $\mathbf{c}_i$. The explicit mathematical formulations for these statistics are provided in Appendix B.2. The resulting $A_{\text{leaf}}(u_{i,m,k})$ is the GRPO advantage for the k-th leaf of a fully expanded tree $\mathcal{T}_{i,m}^{(L)}$, which will be utilized in the subsequent credit propagation step. This two-level normalization turns the absolute reward into a *relative* preference among competing leaves while preserving local competition within each tree.

3.3 Checkpoint-segment credit propagation.

Leaf-level advantages supervise only the final denoised structures, whereas the policy makes decisions throughout the reverse diffusion trajectory. To transfer this signal to intermediate checkpoint states, we assign each internal node a subtree value and then convert parent-to-child changes in that value into segment-level advantages. For clarity, we suppress the sample index i and tree index m in the equations below. Let v be a non-root checkpoint node in a search tree, let $\text{Desc}(v)$ denote the set of selected leaves that descend from v, and let $\kappa(v) \in \mathcal{K}$ be the checkpoint time associated with v. We define the node value as the average leaf advantage among the selected descendants of v:

$$V(v) \triangleq \frac{1}{|\,\text{Desc}(v)|} \sum_{u \in \text{Desc}(v)} A_{\text{leaf}}(u). \tag{4}$$

For the checkpoint edge from the parent node $\text{pa}(v)$ to v, we then define the segment-level advantage by the difference of subtree values as $A_{\text{seg}}(\text{par}(v) \to v) \triangleq V(v) - V(\text{par}(v))$. This difference form attributes positive credit to a segment when entering the subtree rooted at v increases the expected quality of its selected descendants, and negative credit when it decreases that expectation. Finally, each reverse-diffusion step on that edge is supervised by a convex combination of the segment credit and the leaf advantage of the rollout it belongs to: for a rewarded leaf u and an edge $\text{pa}(v) \to v$ on its path, $A_t = \lambda\, A_{\text{seg}}(v) + (1-\lambda)\, A_{\text{leaf}}(u)$ for $t \in \big(\kappa(v),\, \kappa(\text{pa}(v))\big]$, where the credit-mixing coefficient $\lambda \in [0,1]$ gives trajectory-level credit at $\lambda = 0$ and pure segment credit at $\lambda = 1$; we use $\lambda = 0.5$ in all experiments. The leaf term is retained because $A_{\text{seg}}(v)$ vanishes whenever $\text{Desc}(v) = \text{Desc}(\text{pa}(v))$, which would leave the prefix shared by all rewarded leaves unsupervised.

3.4 Reward Modeling

For each training condition $\mathbf{c}_i$, the tree-search rollout policy produces fully denoised leaf structures $\{\hat{\mathbf{x}}_{i,m,k}\}$, where m indexes the tree and k indexes a sampled leaf. We assign a *terminal* reward to each leaf, since the downstream objectives—binding quality, interface reliability, and structural correctness—are most naturally defined on the final 3D structure. The composite reward combines three complementary terms:

$$R_{i,m,k} = \lambda_{\text{aff}} R_{\text{aff}}(\hat{\mathbf{x}}_{i,m,k}, \mathbf{c}_i) + \lambda_{\text{conf}} R_{\text{conf}}(\hat{\mathbf{x}}_{i,m,k}, \mathbf{c}_i) + \lambda_{\text{str}} R_{\text{str}}(\hat{\mathbf{x}}_{i,m,k}, \mathbf{x}_i^\star), \tag{5}$$

where $\lambda_{\text{aff}}, \lambda_{\text{conf}}, \lambda_{\text{str}} \geq 0$ trade off binding optimization against structural plausibility. (1) The *affinity reward* R_{aff} applies a sigmoid to a learned binding logit, providing the most task-aligned signal for optimization. (2) The *interface-confidence reward* R_{conf} aggregates iPTM, iIPTM, and an exponential transform of inter-chain PAE to suppress unstable or poorly supported interfaces. (3) The *structure regularizer* R_{str} is a smooth lDDT-style term that penalizes distortions of local geometry relative to the reference structure $\mathbf{x}_i^\star$, discouraging reward hacking through unrealistic conformations. Full definitions and hyperparameters are provided in Appendix B.1. Because R_{aff} and R_{conf} are read from the Boltz confidence and affinity heads, any Boltz-derived score is an optimization target rather than an unbiased evaluator; we therefore exclude such scores from our headline comparisons and evaluate with independent model weights instead (Section 4).

Algorithm 1 Tree-search GRPO for Diffusion Fine-tuning (TSG-Diff)

Require: Batch $\mathcal{B} = \{\mathbf{c}_i\}$; reward function R; tree params (M, N, L, K); checkpoint schedule $\mathcal{K} = \{\kappa_q\}_{q=0}^{Q}$; PPO params $(\epsilon, \beta, \varepsilon_{\text{norm}})$.

for condition $\mathbf{c}_i \in \mathcal{B}$ and $m = 1 \dots M$ **do** ▷ Initialize M trees per sample
  Sample trajectory $\mathbf{z}_{i,m}^{(T\to 0)} \sim \pi_\theta$ and initialize $\mathcal{T}_{i,m}$ from checkpoints $\xi_{i,m}^{(q)} = (\mathbf{c}_i, \kappa_q, \mathbf{z}_{i,m}^{(\kappa_q)})$
for round $\ell = 1 \dots L$ and tree $\mathcal{T}_{i,m}^{(\ell)}$ **do** ▷ Expand search trees
  Sample internal nodes $\mathcal{V}_{i,m}^{(\ell)} \subset \text{Int}(\mathcal{T}_{i,m}^{(\ell)})$ with $|\mathcal{V}_{i,m}^{(\ell)}| = N$
  for node $v \in \mathcal{V}_{i,m}^{(\ell)}$ **do**
    Resume reverse diffusion from ξ_v to generate suffix $\tau'_{\kappa(v):0} \sim \pi_\theta$ and attach to $\mathcal{T}_{i,m}^{(\ell)}$
for tree $\mathcal{T}_{i,m}^{(L)}$ **do** ▷ Evaluate leaves & compute advantages
  Sample final leaves $\mathcal{U}_{i,m} \subset \text{Leaf}(\mathcal{T}_{i,m}^{(L)})$ with $|\mathcal{U}_{i,m}| = K$
  For each $u \in \mathcal{U}_{i,m}$, decode structure $\hat{\mathbf{x}}_u$, and compute reward $r_u = r(\hat{\mathbf{x}}_u, \mathbf{c}_i)$
  For each $u \in \mathcal{U}_{i,m}$, compute leaf advantage $A_{\text{leaf}}(u)$ via Equation (3)
  For each internal node v, compute $V(v)$ and segment advantage $A_{\text{seg}}(\text{par}(v) \to v)$
Policy Update: Update θ via gradient ascent to maximize the objective $\mathcal{J}(\theta)$ defined in Equation (6).

3.5 Optimization

In chain-based GRPO, a trajectory-level advantage is broadcast across all denoising steps of each rollout. TSG-Diff instead computes leaf-level relative advantages and then propagates them to checkpoint segments, yielding a step-wise signal A_t^g that depends on the tree location of rollout path g. The resulting PPO-style objective is

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathbf{c}\sim p_{\text{data}},\, \{\tau^g\}\sim\pi_{\theta_{\text{old}}}(\cdot|\mathbf{c})}\left[\frac{1}{GT}\sum_{g=1}^{G}\sum_{t=1}^{T}\left(\min\left(r_t^g(\theta)\hat{A}_t^g,\ \text{clip}(r_t^g(\theta), 1-\varepsilon, 1+\varepsilon)\,\hat{A}_t^g\right) - \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\right)\right], \tag{6}$$

where $r_t^g(\theta) = \frac{\pi_\theta(\mathbf{a}_t^g|\mathbf{s}_t^g)}{\pi_{\theta_{\text{old}}}(\mathbf{a}_t^g|\mathbf{s}_t^g)}$ is the importance ratio. A_t^g is the segment-level credit inherited by step t from the tree-search rollout, rather than a single trajectory-level score shared across the full denoising chain. This is the key difference between TSG-Diff and standard group-based diffusion RL. The complete algorithm is depicted in Alg. 1.

4 Experiments

4.1 Experimental Setup

This section presents a comprehensive empirical evaluation of TSG-Diff. Our experiments address three central questions: **Q1**: Does tree-based rollout reuse improve diversity and reward efficiency under matched denoising compute? **Q2**: Can TSG-Diff preserve structural plausibility while optimizing binding-related objectives? **Q3**: How do the key architectural components contribute to overall performance? We evaluate TSG-Diff against baseline methods on two tasks: (1) *Peptide binder design*, which generates peptide structures conditioned on a specified receptor binding site; and (2) *Side-chain packing*, which predicts side-chain conformations for peptides bound to a specified receptor site. We also include ablations to study the effect of group-relative optimization, tree-search rollout expansion, and tree-based credit assignment.

Dataset. We follow the structural data mixture used by BoltzGen (Pacesa et al., 2024; Stark et al., 2025), which builds on the Boltz-2 data pipeline. Specifically, we use experimental structures from the Protein Data Bank (PDB) (Berman et al., 2000) and self-distilled structures from AlphaFold2 (AFDB) and Boltz-1. See Appendix C.1 for more dataset details.

Metrics. Following peptide co-design benchmarks such as PepFlow Li et al. (2024) and PPFlow Lin et al. (2024), we evaluate generated complexes with energy, geometry, confidence, and design-quality metrics. These measure whether a design improves native-relative stability and affinity,

Table 1: Comparison of peptide binder design methods across energy, geometry, and design-quality metrics. Stab., Aff., Des., and Div. abbreviate stability, affinity, designability, and diversity. Every metric is produced by an evaluator that shares no weights with the BoltzGen policy or its learned reward heads (Rosetta REF15, DSSP/TM-score, ESMFold). The Boltz-derived confidence scores used inside our reward are *reward-model-coupled* and are reported separately in Appendix C.4; independent confidence is reported in Table 2. Best in **bold**, second-best underlined.

	Energy		Geometry			Design		
	Stab. % (↑)	Aff. % (↑)	AAR % (↑)	SSR % (↑)	BSR % (↑)	Des. % (↑)	Div_{stru} (↑)	Div_{seq} (↑)
RFdiffusion	8.23	9.63	32.98	71.66	70.31	66.02	0.742	0.519
PPFLOW	11.62	11.33	38.72	68.57	68.41	69.24	0.639	0.500
PepGLAD	10.91	12.64	36.70	72.69	69.53	63.21	0.662	0.628
PepFlow w/Bb+Seq+Ang	13.39	14.83	**44.22**	75.07	71.05	**76.26**	0.585	0.421
BoltzGen	14.31	14.65	39.81	73.26	72.61	73.36	0.861	0.607
TSG-Diff (Ours)	**17.67**	**16.00**	42.71	**77.49**	**73.40**	71.46	**0.867**	**0.702**

preserves the peptide and interface geometry, receives high external confidence, and remains both refoldable and diverse. *Separating the reward model from the evaluator:* Our composite reward reads the affinity and confidence heads of the Boltz stack (Section 3.4), so any Boltz-derived confidence score is *reward-model-coupled*: improving it is partly the training objective and cannot serve as independent evidence of design quality. We therefore report Boltz iPTM and pLDDT only as an optimization diagnostic in Appendix C.4, and base every headline comparison on evaluators that share no weights with our policy or its reward heads: Rosetta REF15 for energy, DSSP and TM-score against the native complex for geometry, and ESMFold for designability (Table 1). For confidence we re-fold each design from scratch with Chai-1 (team et al., 2024), Protenix (Team et al., 2025), and AF2-Multimer via ColabFold (Evans et al., 2021; Mirdita et al., 2022), and add ESM-C pseudo-perplexity, a sequence-only metric that never sees a structure (Table 2). Unless otherwise stated larger values are better; for side-chain packing and pseudo-perplexity, lower is better. Details are in Appendix C.2 and C.3.

Baselines. We compare TSG-Diff with representative peptide/protein design baselines, including backbone-centric generators, full-atom peptide generators, the BoltzGen prior, and dedicated side-chain packing methods. For co-design, the baseline discussion covers RFDiffusion, PPFLOW, PepGLAD, PepFlow, and the BoltzGen base-prior baseline; for packing, it covers SCWRL4, DLPacker, AttnPacker, DiffPack, and PepFlow under the fixed-backbone protocol. Method-specific details and implementation choices are provided in Appendix C.5.

4.2 Primary Results

Performance comparison. As shown in Table 1, TSG-Diff performs competitively across peptide co-design objectives, obtaining the top value on most metrics and ranking second on AAR. It improves the energy metrics relative to the strongest baselines while maintaining strong interface geometry among diffusion methods, and retains high structural and sequence diversity while remaining competitive in Designability; PepFlow w/Bb+Seq+Ang achieves higher Designability, but with noticeably lower diversity. None of these metrics depends on the reward model. Table 2 asks whether the confidence gains survive when the evaluator shares no weights with the reward. TSG-Diff improves on all ten independent measurements, with large margins on Protenix ipTM, AF2-Multimer ipTM, Protenix binder pLDDT, and ESM-C pseudo-perplexity. Against external baselines TSG-Diff is best on seven of ten measurements. Overall, the decoupled evaluation supports improved binding energetics, interface geometry, and sequence naturalness with largely preserved diversity, but not a claim of uniform dominance on interface confidence.

Visualization. We further present four examples of generated peptides in Figure 3. We observe that TSG-Diff produces peptides with proper geometries and binding poses. The generated peptides recapitulate the native binding orientation and conform to the receptor pocket, demonstrating the ability to interact with the target binding sites and capture the right interface shape. To quantify the structural quality of each design, we annotate every panel with three complementary metrics: the Rosetta total energy, Boltz-2 confidence scores iPTM and pLDDT. Across the four examples our designs achieve high Boltz-2 pLDDT (≥ 0.95), indicating that the predicted peptide–target complexes are structurally plausible.

Table 2: Independent-evaluator assessment. Every design is re-folded from scratch by three predictors that share no weights with the BoltzGen policy or its learned reward heads: Chai-1 (team et al., 2024), Protenix (Team et al., 2025), and AF2-Multimer via ColabFold with an MSA for the target chain only (Evans et al., 2021; Mirdita et al., 2022). Cells report mean ipTM / pTM / binder pLDDT (pLDDT on a 0–100 scale; higher is better for all three); the last row is ESM-C pseudo-perplexity, for which lower is better. Values are not comparable *across* evaluators, as each has its own calibration and MSA regime; only within-row comparisons are meaningful.

Evaluator	RFdiffusion	PPFLOW	PepGLAD	PepFlow w/Bb+Seq+Ang	BoltzGen	TSG-Diff (Ours)
Chai-1	0.347 / **0.884** / 93.1	0.228 / 0.682 / 90.7	0.207 / 0.560 / 89.3	**0.494** / 0.877 / 92.3	0.274 / 0.848 / 91.8	0.357 / 0.862 / **93.3**
Protenix	0.726 / 0.883 / 55.6	0.639 / 0.865 / 50.6	0.618 / 0.844 / 47.1	0.658 / 0.897 / 56.9	0.645 / 0.895 / 55.2	**0.744** / **0.908** / **64.9**
AF2-Multimer	0.480 / **0.852** / 42.8	0.368 / 0.735 / 37.8	0.328 / 0.630 / 35.6	0.433 / 0.716 / 40.3	0.419 / 0.826 / 37.6	**0.516** / 0.828 / **43.6**
ESM-C ppl (↓)	13.73	18.66	17.21	16.44	16.55	**13.52**

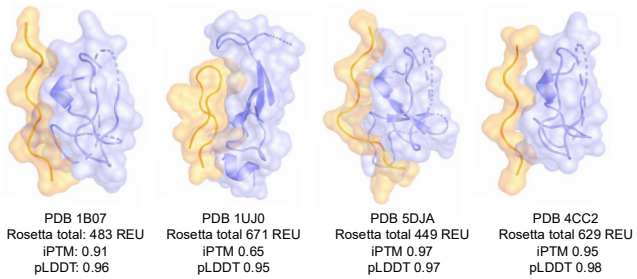


Figure 3: Visualization of generated peptides by our TSG-Diff. Generated peptides (in orange) for receptors (in purple). The PDB IDs of the receptors are 1B07, 1UJ0, 5DJA, and 4CC2.

Ablation study. To assess the contribution of each component of TSG-Diff, we ablate along three axes. (1) *-within-tree norm* / *-across-tree norm*: removing either normalization in Equation (3) isolates how local (intra-tree) competition and global (cross-tree) calibration shape the leaf-level advantage. (2) *-affinity*, *-confidence*, *-structure*: dropping each term from the composite reward exposes its contribution to binding quality, interface plausibility, and geometric realism. (3) *-segment credit* / *+trajectory credit*: keeping the checkpoint-tree sampler unchanged, we replace segment advantages with a single normalized terminal advantage broadcast to every step on the root-to-leaf path, isolating segment-level credit while preserving the tree-shaped rollouts. (4) *-tree reuse* / *+independent chains*: we instead sample independent full chains under the same step budget with trajectory-level GRPO credit; compared with (3), this isolates prefix reuse from finer-grained credit.

Table 3 shows that every ablation underperforms TSG-Diff. Both reward normalizations are load-bearing. Removing across-tree calibration is more damaging than removing within-tree competition, while within-tree normalization chiefly preserves intra-tree diversity. The composite reward terms affect their intended metric families: removing *affinity* causes the largest energy losses, removing *confidence* collapses Boltz iPTM and pLDDT with little energy loss, and removing the *structure regularizer* most harms geometry. Finally, the rollout ablations isolate the two algorithmic ingredients of TSG-Diff. Replacing segment-level with trajectory-level credit (while keeping the tree structure) drops Stability by 1.63, SSR by 1.87, and iPTM by 0.038. Switching further to independent chains reduces performance on the same metrics. Segment-level credit therefore accounts for the larger share of the gain, and prefix reuse across branches contributes a smaller but consistent variance reduction on top.
x

4.3 Side-chain Packing

For side-chain packing, the peptide backbone coordinates and amino-acid sequence are held at their native values, and the model generates only the side-chain torsion angles, conditioned on the receptor. This matches the standard protocol used by all baselines: SCWRL4 Krivov et al. (2009), DLPacker Misiura et al. (2022), AttnPacker McPartlon & Xu (2022), and DiffPack Zhang et al. (2023) take a fixed backbone as input by construction, while the corresponding PepFlow Li et al. (2024) variant (w/Bb+Seq+Ang) flow-matches only the angle modality with backbone and sequence held at native. We evaluate the BoltzGen prior under exactly this protocol as well, conditioning the

Table 3: Ablations on different components of TSG-Diff. Metric abbreviations follow Table 1. WT and AT denote within-tree and across-tree normalization, while Traj. and Ind. denote trajectory-level credit assignment and independent chains, respectively.

Ablations		Energy		Geometry			Confidence		Design		
		Stab. % (↑)	Aff. % (↑)	AAR % (↑)	SSR % (↑)	BSR % (↑)	iPTM (↑)	pLDDT (↑)	Des. % (↑)	Div_{stru} (↑)	Div_{seq} (↑)
Full		**17.67**	**16.00**	<u>42.71</u>	**77.49**	**73.40**	**0.625**	**0.798**	**71.46**	**0.867**	**0.702**
Reward normalization	w/o WT norm	16.39	15.05	41.49	76.87	72.83	0.593	0.768	71.07	0.853	0.679
	w/o AT norm	15.84	14.71	41.05	75.43	72.78	0.582	0.758	70.51	0.862	0.659
Composite reward	w/o Aff.	14.69	13.89	**43.18**	76.21	72.85	0.611	0.789	71.27	0.866	0.691
	w/o Conf.	16.46	15.18	41.21	74.83	71.94	0.539	0.722	69.32	0.858	0.683
	w/o Struct.	15.63	14.92	38.48	70.84	68.31	0.591	0.762	70.29	0.864	0.694
Credit assignment and rollout structure	Traj. credit	16.04	14.83	41.18	75.62	71.97	0.587	0.762	70.42	0.860	0.668
	Ind. chains	15.43	14.47	40.52	74.95	72.46	0.572	0.748	69.74	0.857	0.654

Table 4: Evaluation of different methods on the side-chain packing task. Best and second-best are **bolded** and <u>underlined</u>.

	Angle MAE °(↓)				Angle Accuracy % (↑)		
	χ_1	χ_2	χ_3	χ_4	All residues	Core residues	interface area residues
SCWRL4	28.83	36.99	46.20	62.76	53.26	67.30	40.99
DLPacker	27.44	33.72	43.69	63.71	53.21	68.52	37.61
AttnPacker	<u>25.95</u>	32.51	45.87	<u>58.38</u>	57.88	**71.43**	43.02
DiffPack	27.43	<u>28.86</u>	43.92	60.46	<u>61.79</u>	67.00	**48.67**
PepFlow w/Bb+Seq+Ang	26.36	31.77	**40.31**	59.63	59.39	66.09	45.91
BoltzGen prior	26.90	29.89	45.77	61.24	58.62	68.23	42.74
TSG-Diff (ours)	**24.58**	**27.26**	<u>42.68</u>	**56.72**	**62.41**	<u>69.53</u>	<u>47.63</u>

all-atom sampler on the native backbone and sequence so that it denoises only the torsion angles. Since TSG-Diff fine-tunes an all-atom diffusion prior with a binding-oriented reward, this experiment provides a stress test for Q2: whether RL post-training preserves the all-atom structural fidelity inherited from the base model, rather than collapsing toward reward-favouring modes. We report two metrics: (1) *Angle MAE*, the mean absolute error between predicted and ground-truth χ angles, with standard symmetry handling for χ-symmetric residues; and (2) *Angle Accuracy*, the fraction of torsion angles within 20° of the native value. Following the metric methodology of Zhang et al. (2023), we report results separately for core, interface-area, and all residues, where core residues are those with relative SASA below 20% in the complex and interface-area residues are peptide residues with at least one heavy-atom contact within 5 Å of the receptor. For each peptide, we generate 64 conformations and report the per-sample mean.

Table 4 presents the results. TSG-Diff achieves the lowest MAE on χ_1, χ_2, and χ_4, and the second-lowest on χ_3. On angle accuracy, it attains the best all-residue score of 62.41%, surpassing the strongest packer baseline DiffPack, and ranks second on both core and interface-area residues. The hardest angle, χ_4, sees the largest absolute gain: TSG-Diff reduces the MAE by 1.66° over the previous best AttnPacker. These results show that TSG-Diff sharpens the all-atom prior of BoltzGen rather than degrading it. Tree-search GRPO fine-tuning under a binding reward improves task reward while remaining competitive with dedicated side-chain packers.

5 Conclusion

We presented TSG-Diff, a tree-search RL framework for fine-tuning diffusion models in protein design. By organizing reverse diffusion as a checkpoint tree and combining group-relative leaf normalization with subtree-value propagation, TSG-Diff converts terminal rewards into segment-level credit while reusing denoising prefixes across rollouts, addressing both the budget inefficiency and sparse supervision of trajectory-level RL. Instantiated on BoltzGen, it improves binding-related objectives without sacrificing structural plausibility or naturalness. More broadly, the tree-search view offers a practical recipe for optimizing diffusion-based generators under tight rollout budgets, with adaptive checkpoint selection and multi-objective extensions as natural next steps.

AI USE STATEMENT

AI-assisted tools were used to improve the clarity and presentation of the manuscript, support literature retrieval and the identification of related work, and assist in monitoring the execution of the experiment.

REFERENCES

Rebecca F. Alford, Andrew Leaver-Fay, Jeliazko R. Jeliazkov, Matthew J. O'Meara, Frank P. DiMaio, Hahnbeom Park, Maxim V. Shapovalov, P. Douglas Renfrew, Vikram K. Mulligan, Kalli Kappel, et al. The rosetta all-atom energy function for macromolecular modeling and design. *Journal of Chemical Theory and Computation*, 13(6):3031–3048, 2017. doi: 10.1021/acs.jctc.7b00125.

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.

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1): 235–242, 2000.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Tong Wang, Samuel Marks, Charbel-Raphael Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biyik, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open problems and fundamental limitations of reinforcement learning from human feedback. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Lingjiao Chen, Jared Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more llm calls all you need? towards the scaling properties of compound ai systems. *Advances in Neural Information Processing Systems*, 37:45767–45790, 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.

Zheng Ding and Weirui Ye. Treegrpo: Tree-advantage grpo for online rl post-training of diffusion models. *arXiv preprint arXiv:2512.08153*, 2025.

Richard Evans, Michael O'neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, et al. Protein complex prediction with alphafold-multimer. *biorxiv*, pp. 2021–10, 2021.

Samuel Genheden and Ulf Ryde. The mm/pbsa and mm/gbsa methods to estimate ligand-binding affinities. *Expert opinion on drug discovery*, 10(5):449–461, 2015.

Scott A Hollingsworth and Ron O Dror. Molecular dynamics simulation for all. *Neuron*, 99(6): 1129–1143, 2018.

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.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. *arXiv preprint arXiv:2509.21240*, 2025.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, 1976. doi: 10.1107/S0567739476001873.

Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983. doi: 10.1002/bip.360221211.

Robert Karlsson. Spr for molecular interaction analysis: a review of emerging application areas. *Journal of Molecular Recognition*, 17(3):151–161, 2004.

Xiangzhe Kong, Yinjun Jia, Wenbing Huang, and Yang Liu. Full-atom peptide design with geometric latent diffusion. *Advances in Neural Information Processing Systems*, 37:74808–74839, 2024.

Georgii G Krivov, Maxim V Shapovalov, and Roland L Dunbrack Jr. Improved prediction of protein side-chain conformations with scwrl4. *Proteins: Structure, Function, and Bioinformatics*, 77(4): 778–795, 2009.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024.

Jiahan Li, Chaoran Cheng, Zuofan Wu, Ruihan Guo, Shitong Luo, Zhizhou Ren, Jian Peng, and Jianzhu Ma. Full-atom peptide design based on multi-modal flow matching. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 27615–27640. PMLR, 2024. URL `https://proceedings.mlr.press/v235/li24o.html`.

Haitao Lin, Odin Zhang, Huifeng Zhao, Dejun Jiang, Lirong Wu, Zicheng Liu, Yuwei Huang, and Stan Z Li. PPFlow: Target-aware peptide design with torsional flow matching. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

Daniel J Lizotte and Eric B Laber. Multi-objective markov decision processes for data-driven decision support. *Journal of Machine Learning Research*, 17(210):1–28, 2016.

Matt McPartlon and Jinbo Xu. An end-to-end deep learning method for rotamer-free protein side-chain packing. *bioRxiv*, pp. 2022–03, 2022.

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.

Mikita Misiura, Raghav Shroff, Ross Thyer, and Anatoly B Kolomeisky. Dlpacker: Deep learning for prediction of amino acid side chain conformations in proteins. *Proteins: Structure, Function, and Bioinformatics*, 90(6):1278–1290, 2022.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022a.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022b.

Martin Pacesa, Lennart Nickel, Christian Schellhaas, Joseph Schmidt, Ekaterina Pyatova, Lucas Kissling, Patrick Barendse, Jagrity Choudhury, Srajan Kapoor, Ana Alcaraz-Serna, et al. Bindcraft: one-shot design of functional protein binders. *BioRxiv*, pp. 2024–09, 2024.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation (2023). *arXiv preprint arXiv:2310.03739*, 2023.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, volume 33 of *AAAI'19/IAAI'19/EAAI'19*, pp. 4780–4789, Honolulu, Hawaii, USA, January 2019. AAAI Press. ISBN 978-1-57735-809-1.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

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*, pp. 2025–11, 2025.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3): 2497–2562, 2023.

ByteDance AML AI4Science Team, Xinshi Chen, Yuxuan Zhang, Chan Lu, Wenzhi Ma, Jiaqi Guan, Chengyue Gong, Jincai Yang, Hanyu Zhang, Ke Zhang, et al. Protenix-advancing structure prediction through a comprehensive alphafold3 reproduction. *BioRxiv*, pp. 2025–01, 2025.

Chai Discovery team, Jacques Boitreaud, Jack Dent, Matthew McPartlon, Joshua Meier, Vinicius Reis, Alex Rogozhonikov, and Kevin Wu. Chai-1: Decoding the molecular interactions of life. *BioRxiv*, pp. 2024–10, 2024.

Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*, 2024.

Susana Vázquez Torres, Philip JY Leung, Preetham Venkatesh, Isaac D Lutz, Fabian Hink, Huu-Hien Huynh, Jessica Becker, Andy Hsien-Wei Yeh, David Juergens, Nathaniel R Bennett, et al. De novo design of high-affinity binders of bioactive helical peptides. *Nature*, 626(7998):435–442, 2024.

Lingle Wang, Yujie Wu, Yuqing Deng, Byungchan Kim, Levi Pierce, Goran Krilov, Dmitry Lupyan, Shaughnessy Robinson, Markus K Dahlgren, Jeremy Greenwood, et al. Accurate and reliable prediction of relative ligand binding potency in prospective drug discovery by way of a modern free-energy calculation protocol and force field. *Journal of the American Chemical Society*, 137 (7):2695–2703, 2015.

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.

Thomas Wiseman, Samuel Williston, John F Brandts, and Lung-Nan Lin. Rapid measurement of binding constants and heats of binding using a new titration calorimeter. *Analytical biochemistry*, 179(1):131–137, 1989.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pp. 2226–2240. PMLR, 2023.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and Ming Ding. Bfs-prover: Scalable best-first tree search for llm-based automatic theorem proving. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32588–32599, 2025.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

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.

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. doi: 10.1093/nar/gki524.

Yangtian Zhang et al. Diffpack: A torsional diffusion model for autoregressive protein side-chain packing. In *Advances in Neural Information Processing Systems*, volume 36, pp. 48150–48172, 2023.

A Related Works

A.1 Tree-based Search

Tree-based search is widely used in sequential decision problems (Świechowski et al., 2023) and RL algorithms (Wu et al., 2023) for a good exploration-exploitation trade-off, and is directly applicable to MDPs (Puterman, 1994; Lizotte & Laber, 2016). MCTS has been used to construct offline DPO datasets from high- and low-quality trajectory pairs (Xie et al., 2024; Lai et al., 2024), and for test-time scaling by searching for better answers during inference (Yao et al., 2023; Xin et al., 2025), both in a one-shot, non-interactive manner. Its reliance on multi-turn sequential rollouts is poorly suited for parallelized inference engines on diffusion models, bottlenecking throughput. Tree-GRPO (Ji et al., 2025) uses a tree structure to improve rollout efficiency, but applying MCTS-style search to online RL creates a systems bottleneck: online RL repeatedly generates rollouts and updates the policy, while MCTS expands nodes sequentially, with each rollout determining the next step. This clashes with inference engines optimized for batched generation, making MCTS prohibitively slow when thousands of rollouts are needed per training step. To mitigate this, Tree-GRPO adopts an initialize-then-expand strategy: it first generates M complete reasoning chains in parallel, then selectively expands promising intermediate nodes, preserving batched-generation efficiency while exploiting tree-structured diversity and reuse.

A.2 Diffusion Models with Reinforcement Learning

Recent RL methods treat diffusion generation as a sequential decision-making problem. DDPO (Black et al., 2023) trains the reverse diffusion process using rewards defined on the final denoised output, enabling direct optimization for task-specific objectives without differentiable rewards or inference-time guidance. DanceGRPO (Xue et al., 2025) and FlowGRPO (Liu et al., 2025) further adapt GRPO to diffusion models by exploiting the tractable Gaussian transitions between adjacent denoising steps. Compared with guidance, constrained decoding, or reward-based post hoc selection, RL fine-tuning directly updates the model toward downstream objectives. However, existing methods typically rely on chain-based rollouts and trajectory-level rewards, which can make exploration inefficient and supervision sparse. TSG-Diff builds on this line of work, but introduces tree-search rollouts and checkpoint-level credit propagation for more efficient exploration and finer-grained feedback.

B Model Details

B.1 Composite Reward Formulation

The fine-tuned policy should maximize the expected value of a property of interest while preserving prior information from the pretrained model through KL regularization. We formalize this diffusion fine-tuning objective as

$$\arg\max_\pi \mathbb{E}_{x\sim p_1^\pi}[r(x)] - \alpha D_{KL}(p_1^\pi || p_1^{\text{pre}}) \tag{7}$$

where $r : \mathcal{X} \to \mathbb{R}$ is a scalar reward function and $\alpha > 0$ determines the KL-regularization strength.

For each training condition $\mathbf{c}_i$ (e.g., receptor, design mask, and other structural context), the tree-search rollout policy produces a set of fully denoised leaf structures $\{\hat{\boldsymbol{x}}_{i,m,k}\}_{k=1}^{K_{i,m}}$, where $m \in \{1, \dots, M\}$ indexes the tree and $K_{i,m}$ is the number of terminal leaves in that tree. We assign a *terminal* reward to each leaf, rather than to intermediate noisy states, because the downstream objectives of interest—binding quality, interface reliability, and structural correctness—are most naturally defined on the final 3D structure.

Formally, let $\hat{\boldsymbol{x}}_{i,m,k}$ denote the final structure associated with leaf k from tree m under condition $\mathbf{c}_i$, and let $\boldsymbol{x}_i^\star$ denote the corresponding reference structure available during training. We define the composite reward as

$$R_{i,m,k} = R(\hat{\boldsymbol{x}}_{i,m,k}, \mathbf{c}_i, \boldsymbol{x}_i^\star) = \lambda_{\text{aff}} R_{\text{aff}}(\hat{\boldsymbol{x}}_{i,m,k}, \mathbf{c}_i) + \lambda_{\text{conf}} R_{\text{conf}}(\hat{\boldsymbol{x}}_{i,m,k}, \mathbf{c}_i) + \lambda_{\text{str}} R_{\text{str}}(\hat{\boldsymbol{x}}_{i,m,k}, \boldsymbol{x}_i^\star), \tag{8}$$

where $\lambda_{\text{aff}}, \lambda_{\text{conf}}, \lambda_{\text{str}} \geq 0$ control the trade-off between optimizing binding-related properties and preserving structural plausibility. In our setting, the structure term acts as a regularizer that prevents the policy from exploiting the learned binding scores by producing geometrically implausible structures.

(1) Affinity reward. The first component directly encourages binding. We use a learned affinity head f_{aff} that maps the final structure and conditioning context to a scalar binding logit,

$$R_{\text{aff}}(\hat{\boldsymbol{x}}_{i,m,k}, \mathbf{c}_i) = s(f_{\text{aff}}(\hat{\boldsymbol{x}}_{i,m,k}, \mathbf{c}_i)), \tag{9}$$

where $s(u) = (1+\exp(-u))^{-1}$ is the logistic sigmoid. Thus, $R_{\text{aff}} \in [0,1]$ can be interpreted as the predicted probability that the generated design forms a favorable binder under the given condition. Compared with purely geometry-based rewards, this term is the most task-aligned signal for binding optimization.

(2) Interface-confidence reward. Affinity alone is not sufficient, because a model can assign high scores to structures with an uncertain or poorly supported interface. We therefore add a confidence-based reward that measures whether the predicted interface is self-consistent according to the model's confidence head:

$$R_{\text{conf}}(\hat{\boldsymbol{x}}_{i,m,k}, \mathbf{c}_i) = \alpha_1 \,\text{iPTM}_{d\to t} + \alpha_2 \,\text{iIPTM}_d + \alpha_3 \exp\left(-\frac{\text{PAE}_{\text{inter}}}{\tau_{\text{pae}}}\right). \tag{10}$$

Here, $\text{iPTM}_{d\to t}$ measures the interface agreement between the designed binder d and the target t, iIPTM_d measures confidence over the designed interface itself, and $\text{PAE}_{\text{inter}}$ denotes the predicted aligned error on inter-chain residue pairs. The exponential transform converts low interface PAE into a high reward, making this term monotone with interface quality. This component is particularly important in early RL training, when the affinity head may be locally informative but not yet sufficient to suppress unstable interfaces.

(3) Structure regularization. To preserve local structural correctness, we use a smooth lDDT-style reward as a regularizer. Let $\mathcal{P}$ be the set of valid local atom pairs,

$$\mathcal{P} = \{(a,b) \,:\, a \neq b,\ d^\star_{ab} < \rho_{ab},\ a,b \text{ are resolved}\}, \tag{11}$$

where $d^\star_{ab} = \|\mathbf{x}^\star_a - \mathbf{x}^\star_b\|_2$ is the reference distance and ρ_{ab} is a local-pair selection threshold. For notational simplicity, we omit the dependence of the atom coordinates on (i,m,k) and on the reference structure $\boldsymbol{x}^\star_i$. We then define

$$R_{\text{str}}(\hat{\boldsymbol{x}}_{i,m,k}, \boldsymbol{x}^\star_i) = \frac{1}{|\mathcal{P}|} \sum_{(a,b)\in\mathcal{P}} \frac{1}{4} \sum_{\delta\in\{0.5,1,2,4\}} s(\delta - |\|\hat{\mathbf{x}}_a - \hat{\mathbf{x}}_b\|_2 - \|\mathbf{x}^\star_a - \mathbf{x}^\star_b\|_2|), \tag{12}$$

which is a smooth approximation of the standard lDDT criterion. This reward is high when the predicted local geometry matches the reference structure and low when the denoised sample distorts local distances. In practice, this term stabilizes RL by discouraging reward hacking through unrealistic conformations.

B.2 Advantage Normalization Statistics

As introduced in Eq. 3, the terminal advantage $A_{\text{leaf}}(u_{i,m,k})$ for the k-th leaf relies on both tree-local and condition-global statistics. For a given condition $\mathbf{c}_i$, let M be the number of generated trees, each containing K fully expanded leaves.

The tree-local mean and standard deviation for the m-th tree are computed as:

$$\mu_{i,m} = \frac{1}{K}\sum_{k=1}^{K} r_{i,m,k}, \qquad \sigma_{i,m} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(r_{i,m,k} - \mu_{i,m})^2}. \tag{13}$$

The condition-global statistics are aggregated across all $M \times K$ leaves associated with the condition:

$$\mu_i = \frac{1}{MK}\sum_{m=1}^{M}\sum_{k=1}^{K} r_{i,m,k}, \qquad \sigma_i = \sqrt{\frac{1}{MK}\sum_{m=1}^{M}\sum_{k=1}^{K}(r_{i,m,k} - \mu_i)^2}. \tag{14}$$

B.3 Detailed Algorithm

Algorithm 2 provides the full pseudocode for the TSG-Diff training loop. It covers tree initialization from M independent root trajectories, L rounds of checkpoint-node expansion, reward computation for all terminal leaves, the within-tree and across-tree leaf-advantage normalization, mixed leaf and checkpoint-segment credit propagation, and the PPO-style policy update over denoising steps.

Algorithm 2 Tree-search GRPO for diffusion fine-tuning

Require: batch $\mathcal{B} = \{\mathbf{c}_i\}$; reward $R(\cdot, \cdot)$; tree hyperparameters M, N, L; checkpoint time schedule $\mathcal{K} = \{\kappa_0 = 0 < \kappa_1 < \cdots < \kappa_Q = T\}$; PPO hyperparameters $\epsilon_-, \epsilon_+, \beta$; credit-mixing coefficient λ; normalization constant $\varepsilon_{\text{norm}}$

set $\pi_{\text{old}} \leftarrow \pi_\theta$

Initialize M trees per sample

for each condition $\mathbf{c}_i \in \mathcal{B}$ **do**

  for $m = 1$ to M **do**

    sample $\mathbf{z}_{i,m}^{(\kappa_Q)} \sim p_T(\cdot)$ and generate $\mathbf{z}_{i,m}^{(t-1)} \sim \pi_{\text{old}}(\cdot \mid \mathbf{z}_{i,m}^{(t)}, \mathbf{c}_i, t)$ for $t = T, \ldots, 1$

    initialize $\mathcal{T}_{i,m}$ from checkpoint tuples $\xi_{i,m}^{(q)} = (\mathbf{c}_i, \kappa_q, \mathbf{z}_{i,m}^{(\kappa_q)})$ for $q = 0, \ldots, Q$

Expand trees

for $\ell = 1$ to L **do**

  for each tree $\mathcal{T}_{i,m}$ **do**

    sample $\mathcal{V}_{i,m}^{(\ell)} \subset \text{Int}(\mathcal{T}_{i,m})$ with $|\mathcal{V}_{i,m}^{(\ell)}| = N$

    for each $v \in \mathcal{V}_{i,m}^{(\ell)}$ **do**

      generate a suffix from $\xi_v = (\mathbf{c}_i, \kappa(v), \mathbf{z}_v)$ by sampling $\mathbf{z}^{(t-1)} \sim \pi_{\text{old}}(\cdot \mid \mathbf{z}^{(t)}, \mathbf{c}_i, t)$ for $t = \kappa(v), \ldots, 1$, and attach its checkpoint tuples to $\mathcal{T}_{i,m}$

Compute rewards for all final leaves

for each tree $\mathcal{T}_{i,m}$ **do**

  set $\mathcal{W}_{i,m} = \text{Leaf}(\mathcal{T}_{i,m})$ and $K_{i,m} = |\mathcal{W}_{i,m}|$

  for each $w_{i,m,k} \in \mathcal{W}_{i,m}$ **do**

    reconstruct $H_{i,m,k}$, decode $\hat{\boldsymbol{x}}_{i,m,k}$, and set $r_{i,m,k} = R(\hat{\boldsymbol{x}}_{i,m,k}, \mathbf{c}_i)$

Compute Tree-search GRPO leaf advantages

for each sample i **do**

  compute $\mu_{i,m} = \text{mean}_{w \in \mathcal{W}_{i,m}} r(w)$, $\sigma_{i,m} = \text{std}_{w \in \mathcal{W}_{i,m}} r(w)$ for each tree m

  compute $\mu_i = \text{mean}_{m,\, w \in \mathcal{W}_{i,m}} r(w)$ and $\sigma_i = \text{std}_{m,\, w \in \mathcal{W}_{i,m}} r(w)$ across all leaves

  for $m = 1$ to M **do**

    for each $w_{i,m,k} \in \mathcal{W}_{i,m}$ **do**

      set $A_{\text{leaf}}(i, m, k) = \dfrac{r_{i,m,k} - \mu_{i,m}}{\sigma_{i,m} + \varepsilon_{\text{norm}}} + \dfrac{r_{i,m,k} - \mu_i}{\sigma_i + \varepsilon_{\text{norm}}}$

Propagate credit to checkpoint segments

For each rewarded leaf w, set $V(w) = A_{\text{leaf}}(w)$

For each non-leaf checkpoint node v, define $V(v) = |\,\text{Desc}_{\mathcal{W}}(v)|^{-1} \sum_{w \in \text{Desc}_{\mathcal{W}}(v)} A_{\text{leaf}}(w)$

for each checkpoint edge $\text{pa}(v) \to v$, set $A_{\text{seg}}(v) = V(v) - V(\text{pa}(v))$ and for each leaf w descending from v, set $A_t = \lambda A_{\text{seg}}(v) + (1 - \lambda) A_{\text{leaf}}(w)$ for all $t \in (\kappa(v),\, \kappa(\text{pa}(v))]$

PPO update over denoising steps

for each sampled denoising step t, let $r_t(\theta) = \frac{\pi_\theta(\mathbf{z}^{(t-1)} | \mathbf{z}^{(t)}, \mathbf{c}_i, t)}{\pi_{\text{old}}(\mathbf{z}^{(t-1)} | \mathbf{z}^{(t)}, \mathbf{c}_i, t)}$ and minimize

$$\mathcal{L}_{\text{PPO}} = -\mathbb{E}_t[\min(r_t(\theta) A_t,\, \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)\, A_t) - \beta\, \text{KL}_t]\,.$$

Optimize

take an optimizer step on $\nabla_\theta \mathcal{L}_{\text{PPO}}$ and log reward, A_{leaf}, and A_t statistics

C Experimental Details

The experiments were conducted on a computing cluster with 4 NVIDIA A6000 GPUs. The total computation time for training was approximately 62 hours. We trained with a per-device batch size

of 2 (effective batch size of 16) using bf16 mixed-precision and gradient clipping at 10.0. We used the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1 \times 10^{-8}$) with a weight decay of 3×10^{-3}, following an AlphaFold-style learning-rate schedule: a linear warmup from 0 to a peak learning rate of 5×10^{-4} over the first 1,000 steps, after which the learning rate is held constant until step 50,000 and then decayed by a factor of 0.95 every 50,000 steps.

C.1 Dataset

We train and evaluate on receptor–peptide complexes following the peptide co-design and side-chain packing protocols used by prior full-atom peptide benchmarks such as PepFlow and PPFlow (Li et al., 2024; Lin et al., 2024). For surface–structure joint design, the receptor and binding-site context are provided to the model, and the peptide sequence, backbone, and side chains are generated jointly. The native peptide is used only for evaluation metrics such as amino-acid recovery, peptide RMSD/TM-score, secondary-structure recovery, and binding-site recovery. For side-chain packing, the receptor, peptide backbone, and peptide sequence are fixed to the native complex, and the model predicts only peptide side-chain conformations; torsion-angle errors and angle accuracies are then computed against the native side chains. Across both tasks, generated complexes are further assessed with Rosetta energy terms, external Boltz confidence metrics, designability, and diversity as detailed in Appendix C.2.

C.2 Evaluation Metrics Details

Following peptide co-design benchmarks such as PepFlow Li et al. (2024) and PPFlow Lin et al. (2024), we group the evaluation metrics into geometry, energy, confidence, and design quality. For each metric, values in the stated preferred direction indicate better performance, while movement in the opposite direction indicates worse performance.

*Geometry. **AAR*** is the position-wise amino-acid recovery of the generated peptide sequence against the native peptide; higher values indicate better sequence recovery. ***PepRMSD*** and ***PepTM*** compare the generated peptide backbone to the native peptide by C_α RMSD after Kabsch superposition Kabsch (1976) and TM-score Zhang & Skolnick (2005), respectively. Lower PepRMSD is better because it means the generated backbone is closer to the native peptide, while higher PepTM is better because it indicates stronger global structural agreement. ***TargetRMSD*** and ***TargetTM*** apply the same measures to the refolded receptor chain, checking whether the target structure is modelled consistently; lower TargetRMSD and higher TargetTM indicate better receptor agreement. ***SSR*** is the fraction of peptide residues with matching DSSP secondary-structure labels Kabsch & Sander (1983), so higher values indicate better secondary-structure preservation. ***BSR*** measures interface preservation as the fraction of native receptor contact residues, defined by heavy-atom distance below 5 Å, recovered by the generated peptide; higher values indicate better recovery of the native binding site.

Energy. All energy terms use the Rosetta full-atom REF15 scoring function Alford et al. (2017). ***RosettaTotal*** is the total score of the complex, and lower values indicate a more favourable overall complex energy. ***RosettadG*** is the interface $\Delta G_{\mathrm{separated}}$, and lower values indicate more favourable binding after separating the partners. From these raw scores we derive two native-relative rates. ***Stability*** is the percentage of generated designs whose RosettaTotal is more favourable than that of the native complex, and ***Affinity*** is the percentage whose RosettadG is more favourable than that of the native complex. Higher Stability and Affinity percentages are better, but they are strict threshold metrics and should be interpreted together with the raw Rosetta scores.

Confidence. We distinguish two families. ***Reward-model-coupled confidence*** comprises Boltz iPTM and pLDDT, which are computed by the same model family that supplies R_{aff} and R_{conf} during training; improving them is partly the training objective itself, so they are an optimization diagnostic rather than an unbiased evaluator, and we report them only in Appendix C.4. ***Reward-model-independent confidence*** comprises the refolding metrics of Appendix C.3, obtained by re-predicting each generated complex with structure predictors whose weights are disjoint from our policy and reward heads. Within either family, higher iPTM/ipTM, pTM, and pLDDT indicate more confident complex and local-structure predictions.

Design quality. Designability is the percentage of generated peptides whose ESMFold Lin et al. (2023) refolding falls within 2 Å C_α RMSD of the generated backbone, probing whether the structure is realizable by a plausible sequence; higher values are better. *Structural diversity* is one minus the mean pairwise peptide TM-score Zhang & Skolnick (2005) among generations for the same target, so higher values indicate greater structural variety. *Sequence diversity* is computed analogously from pairwise peptide sequence dissimilarity among generations for the same target, with higher values indicating less redundant generated sequences. Diversity metrics are most informative when read together with the quality metrics above.

Side-chain packing. For the side-chain packing task, *Angle MAE* is the mean absolute error between predicted and ground-truth side-chain torsion angles, with standard handling for symmetric χ angles; lower values are better. *Angle Accuracy* is the fraction of torsion angles within 20° of the native value; higher values are better. We report these metrics over all peptide residues, core residues, and interface-area residues.

C.3 Evaluation with Independent Model Weights

A reviewer-facing concern with reward-driven fine-tuning is that the evaluator may share weights with the reward model, in which case a reported gain can reflect optimization of the scorer rather than improvement of the design. Our composite reward reads the Boltz affinity and confidence heads, so every Boltz-derived confidence score is coupled to the objective. To evaluate confidence without this coupling, we discard the generated complex's predicted confidences entirely and re-fold each design from scratch with three external structure predictors, none of which shares weights with the BoltzGen policy or with our learned reward heads.

- **Chai-1** (team et al., 2024) is an all-atom biomolecular structure predictor trained independently of the Boltz model family. We run it in single-sequence mode on the receptor–peptide complex.
- **Protenix** (Team et al., 2025) is an open reproduction and extension of the AlphaFold3 architecture, again trained with its own weights. We use the default inference configuration.
- **AF2-Multimer** (Evans et al., 2021) is run through ColabFold (Mirdita et al., 2022). We supply an MSA for the target chain only and leave the designed binder in single-sequence mode, which is the standard protocol for evaluating *de novo* binders, whose sequences have no meaningful homologs. This is the most conservative of the three evaluators: it was trained before any of the generative models compared here and its architecture, training data, and confidence heads are unrelated to Boltz.

For each evaluator we report the mean interface pTM (ipTM), the mean global pTM, and the mean pLDDT restricted to the designed binder chain, on a 0–100 scale. Binder-restricted pLDDT is used rather than complex pLDDT because the receptor is given and would otherwise dominate the average. We emphasize that absolute values are *not* comparable across evaluators: the three predictors have different confidence calibrations and different MSA regimes, which is why, for example, Chai-1 binder pLDDT sits near 90 while AF2-Multimer binder pLDDT sits near 40. Only comparisons within a row are meaningful.

We additionally report **ESM-C pseudo-perplexity**, computed by masking each residue of the designed peptide in turn and averaging the negative log-likelihood of the true token under the ESM-C protein language model. This metric sees only the sequence, never a structure, and therefore cannot be affected by any structural reward; lower values indicate sequences that are more natural under a large protein language model. It provides a check against the failure mode in which a policy produces structurally confident but biologically implausible sequences.

Table 2 reports the results. The comparison we regard as cleanest is TSG-Diff against the BoltzGen prior it is fine-tuned from, since the two differ only by our RL post-training: TSG-Diff improves on all ten independent measurements, although the AF2-Multimer pTM difference ($0.826 \rightarrow 0.828$) is within run-to-run variation and we do not claim it. Against external baselines the result is strong but not uniform. TSG-Diff is best on seven of ten measurements, while PepFlow leads on Chai-1 ipTM and RFdiffusion leads on pTM under both Chai-1 and AF2-Multimer. The disagreement between Chai-1 and the other two evaluators on the interface ranking is itself informative: it indicates that

confidence rankings derived from a single predictor are not stable, which is a further argument for reporting a panel of independent evaluators rather than any one of them.

C.4 Reward-Model-Coupled Confidence Metrics

For completeness we report the Boltz-derived confidence metrics that earlier versions of this work placed in the main comparison. These are *reward-model-coupled*: R_{conf} in Eq. 10 is built from the same iPTM and interface-PAE heads, so TSG-Diff is directly trained to increase them. A higher score for TSG-Diff in Table 5 is therefore expected by construction and should not be read as evidence that our designs are better than the baselines; we include it only to document that the optimization behaves as intended and that the reward did not saturate or collapse. All claims about confidence in this paper are instead supported by Table 2.

Table 5: Reward-model-coupled confidence metrics, computed with the Boltz evaluator. **These numbers are not a fair comparison** between TSG-Diff and the baselines, because R_{conf} optimizes the same model's heads during training; they are reported as an optimization diagnostic only. Independent-weight confidence is reported in Table 2.

	Boltz confidence (reward-model-coupled)	
	iPTM (↑)	pLDDT (↑)
RFdiffusion	0.464	0.559
PPFLOW	0.478	0.517
PepGLAD	0.540	0.571
PepFlow w/Bb+Seq+Ang	0.532	0.689
BoltzGen	0.536	0.720
TSG-Diff (Ours)	0.625	0.798

C.5 Baselines

We summarize the baseline methods used in the two evaluation tasks. The binder design baselines cover both backbone-centric generators and full-atom peptide generators, while the packing baselines cover classical, neural, attention-based, diffusion-based, and flow-based side-chain prediction methods.

Peptide binder design. In this task, the receptor and binding-site context are provided, and the method generates the peptide sequence, backbone, and side-chain conformations. We use the following comparison methods.

- **RFDiffusion** (Watson et al., 2023) is a backbone diffusion model for protein design. We use it as a backbone-centric binder generator; generated backbones are assigned amino-acid sequences with ProteinMPNN (Dauparas et al., 2022). Since the model does not explicitly generate full side-chain conformations during diffusion, side-chain information is introduced after backbone generation.
- **PPFLOW** (Lin et al., 2024) is a target-aware peptide design model that uses conditional flow matching on torus manifolds to model peptide torsion-angle geometry. It conditions on the receptor and binding-site context, and is included as a strong peptide-specific backbone-generation baseline.
- **PepGLAD** (Kong et al., 2024) combines geometric latent diffusion with receptor-specific transformations to generate full-atom peptides. Because it explicitly models peptide atoms in a receptor-conditioned latent space, it provides a full-atom comparison to backbone-centric generators.
- **PepFlow** (Li et al., 2024) is a multimodal flow-matching model for receptor-conditioned, full-atom peptide design. It jointly models peptide backbone geometry, side-chain conformations, and residue identities on their corresponding geometric manifolds. We use the w/Bb+Seq+Ang configuration for compatibility with prior peptide co-design and packing protocols.

- **BoltzGen** (Stark et al., 2025) is the all-atom diffusion prior from which TSG-Diff is fine-tuned. We include it as the direct base-model baseline to isolate the effect of tree-search GRPO post-training.

Side-chain packing. In this task, the receptor, peptide backbone, and peptide sequence are fixed to their native values, and each method predicts only peptide side-chain conformations or torsion angles. This setup matches the standard fixed-backbone protocol used by side-chain packing benchmarks.

- **SCWRL4** (Krivov et al., 2009) is a widely used side-chain packing method based on a backbone-dependent rotamer library and a statistical energy function.
- **DLPacker** (Misiura et al., 2022) is a 3D CNN-based model for predicting residue side-chain conformations. We use the official implementation and released model weights.
- **AttnPacker** (McPartlon & Xu, 2022) uses equivariant attention over backbone 3D geometry to predict all side-chain coordinates jointly.
- **DiffPack** (Zhang et al., 2023) is a diffusion-based generative model that autoregressively samples side-chain torsion angles on a toric manifold.
- **PepFlow** (Li et al., 2024) is evaluated in its side-chain packing setting by holding the receptor, peptide backbone, and peptide sequence fixed and flow-matching only the angle modality.

C.6 HYPERPARAMETERS

Our method introduces several hyperparameters spanning the optimizer, the reverse-diffusion rollout, the tree-search expansion, the GRPO update, and the reward function. To select the operating point, we conducted a random search over the values listed in Table 6.

Table 6: Search space for the proposed TSG-Diff model. The values used in the reported experiments are marked in **bold**.

Group	Parameter	Search Space
Optimization	Peak learning rate	1×10^{-4}, 3×10^{-4}, $\mathbf{5\times10^{-4}}$, 1×10^{-3}
	LR warmup steps	500, **1000**, 2000
	Weight decay	1×10^{-3}, $\mathbf{3\times10^{-3}}$, 1×10^{-2}
	Gradient clip	1.0, 5.0, **10.0**
	Per-device batch size	1, **2**, 4
Reverse-diffusion rollout	Denoising steps per trajectory (T)	20, **50**, 80
	Reverse-transition noise level	0.5, **0.7**, 0.9
	Reverse-step scale	1.0, **1.5**, 2.0
GRPO update	KL coefficient β	1×10^{-3}, $\mathbf{1\times10^{-2}}$, 1×10^{-1}
	PPO clip range	0.1, **0.2**, 0.3
	Advantage clip max	2.0, **5.0**, 10.0
	PPO step-window size	5, **10**, 15
	Credit-mixing coefficient λ	0.25, **0.5**, 0.75
Tree-search rollout	Branch factor at root (M)	2, **3**, 4
	Branches per intermediate node (N)	**2**, 3, 4
	Tree depth (L)	2, **3**, 4
Reward weights	lDDT reward weight	**0.1**, 0.5, 1.0
	Affinity reward weight	0.1, 0.5, **1.0**
	Interface-confidence reward weight	0.1, **0.5**, 1.0
	Energy reward weight	**0.0**, 0.1, 0.5

D COMPUTATIONAL BUDGET OF TREE SEARCH ROLLOUTS

We measure rollout cost using neural function evaluations (NFEs). One NFE corresponds to one reverse denoising step, represented as $\epsilon_\theta(x_t, t, c)$. For a single conditioning input, let M denote the number of independent root trajectories, N denote the number of checkpoint nodes expanded per tree in each round, L denote the number of expansion rounds, and K denote the number of terminal leaves selected from each tree for reward evaluation and the GRPO update.

Candidate leaves versus training group. The sampler first draws M independent full denoising trajectories, one for each tree root. These trajectories provide one terminal leaf per tree, together with intermediate checkpoint nodes along the path. The sampler then performs L expansion rounds. In each round and for each tree, it samples N non-terminal checkpoint nodes with replacement and denoises a new suffix from each selected checkpoint to the terminal state. Thus, each successful expansion adds one candidate terminal leaf, and the number of generated candidate leaves per conditioning input is

$$G_{\text{cand}} = M(1 + LN). \tag{15}$$

This candidate count is not necessarily the GRPO training group size. After tree construction, the implementation samples K leaves from each tree, reconstructs the selected paths, evaluates rewards only for these leaves, and uses them in the policy update. The effective group size used by reward normalization and PPO-style clipping is

$$G_{\text{eff}} = MK. \tag{16}$$

Expected NFE cost. T is the number of denoising steps in a complete trajectory. A standard chain-based GRPO sampler with group size G costs

$$B_{\text{chain}}(G) = GT. \tag{17}$$

For tree search, the M root trajectories cost MT NFEs. Each later expansion starts from an intermediate checkpoint rather than from the initial noisy state. Let $s_{\ell,m,n}$ be the denoising state index of the n-th checkpoint selected in round ℓ of tree m, where state 0 is the initial noisy state and state T is terminal. The corresponding suffix costs $T - s_{\ell,m,n}$ NFEs, so the exact sampling cost for one conditioning input is

$$B_{\text{tree}} = MT + \sum_{\ell=1}^{L} \sum_{m=1}^{M} \sum_{n=1}^{N} (T - s_{\ell,m,n}) \,. \tag{18}$$

For a simple estimate, suppose the selected expansion nodes are approximately uniform over the non-terminal checkpoint states $\{c_0, \ldots, c_{Q-1}\}$, with $0 = c_0 < c_1 < \cdots < c_Q = T$. Then

$$\mathbb{E}\left[T - s_{\ell,m,n}\right] \approx \sum_{q=0}^{Q-1} \Pr(s_{\ell,m,n} = c_q)(T - c_q) = \frac{1}{Q} \sum_{q=0}^{Q-1} (T - c_q). \tag{19}$$

Here $\Pr(s_{\ell,m,n} = c_q)$ denotes the probability that the selected checkpoint for the n-th expansion in round ℓ of tree m is the q-th checkpoint state c_q. Under the uniform-checkpoint approximation, this probability is $1/Q$ for each non-terminal checkpoint. Since there are MLN such expansion suffixes, linearity of expectation gives

$$\mathbb{E}[B_{\text{tree}}] \approx MT + MLN \cdot \frac{1}{Q} \sum_{q=0}^{Q-1} (T - c_q) \,. \tag{20}$$

If the checkpoint states are evenly spaced, then $c_q = qT/Q$. Therefore

$$\begin{aligned} \frac{1}{Q} \sum_{q=0}^{Q-1} (T - c_q) &= \frac{1}{Q} \sum_{q=0}^{Q-1} \left(T - \frac{qT}{Q} \right) \\ &= T - \frac{T}{Q^2} \sum_{q=0}^{Q-1} q \\ &= T - \frac{T}{Q^2} \frac{Q(Q-1)}{2} = T \frac{Q+1}{2Q}. \end{aligned} \tag{21}$$

Substituting Eq. (21) into Eq. (20) gives

$$\mathbb{E}[B_{\text{tree}}] \approx MT\left(1 + LN\frac{Q+1}{2Q}\right) \approx MT\left(1 + \frac{LN}{2}\right). \tag{22}$$

Trade-offs under a fixed budget. Under a fixed sampling budget, M controls breadth across independent noisy roots, N and L control how many alternative suffixes are generated by resuming from intermediate checkpoints, and K controls how many leaves are actually rewarded and used in the GRPO update. Decreasing M while increasing N or L can create more continuations per root, but it also narrows the exploration scope because more selected trajectories share prefixes. Increasing M improves independent exploration but reduces the amount of prefix reuse available under the same denoising budget. We study this trade-off empirically in the ablations.

Budget-normalized effective sample size. The candidate count alone does not characterize the statistical value of tree rollouts, because leaves sampled from the same tree share part of their reverse-diffusion prefix. A simple design-effect approximation makes this dependence explicit. Let $\rho \in [0, 1]$ denote the average pairwise correlation between the rewards, or equivalently the advantage estimates, of two selected leaves from the same tree. Assuming different trees are independent and the K selected leaves within each tree are exchangeable, the effective sample size of the tree-structured training group is approximately

$$\text{ESS}_{\text{tree}} \approx \frac{MK}{1 + (K-1)\rho}. \tag{23}$$

When $\rho = 0$, the selected leaves behave like MK independent rollouts. When $\rho > 0$, shared prefixes reduce the effective number of independent samples.

Combining this approximation with the NFE budget gives a useful measure of statistical efficiency per diffusion-network evaluation. Define the average fractional suffix cost

$$\alpha = \frac{1}{TQ}\sum_{q=0}^{Q-1}(T - c_q), \tag{24}$$

so that Eq. (20) can be written as

$$\mathbb{E}[B_{\text{tree}}] \approx MT(1 + LN\alpha). \tag{25}$$

The resulting effective sample size per NFE is therefore

$$\frac{\text{ESS}_{\text{tree}}}{\mathbb{E}[B_{\text{tree}}]} \approx \frac{K}{T(1 + LN\alpha)\{1 + (K-1)\rho\}}. \tag{26}$$

For MK independent chain rollouts, the corresponding efficiency is

$$\frac{\text{ESS}_{\text{chain}}}{B_{\text{chain}}(MK)} = \frac{MK}{MKT} = \frac{1}{T}. \tag{27}$$

Thus, under the same NFE accounting, tree search is more statistically efficient than independent chain sampling when

$$\frac{K}{(1 + LN\alpha)\{1 + (K-1)\rho\}} > 1. \tag{28}$$

This condition summarizes the rollout-budget trade-off. Increasing K provides more rewarded leaves per root, but the gain saturates when sibling leaves are highly correlated. Increasing N or L creates more candidate suffixes and can improve the chance of selecting high-quality leaves, but it also increases the NFE budget through the factor $1 + LN\alpha$. Earlier checkpoint expansion has larger suffix cost but tends to reduce sibling correlation, whereas later checkpoint expansion is cheaper but produces more strongly correlated leaves.

E Limitations and Future Work

While TSG-Diff significantly improves the sample efficiency and credit assignment of RL fine-tuning for diffusion models, it has several limitations that present exciting avenues for future research.

Proxy Reward Noise and Reward Hacking. First, our framework relies on an automated, composite reward function (e.g., binding affinity or structure confidence oracles) to guide the policy. However, the inherent noise leaves the policy vulnerable to reward hacking Casper et al. (2023), where it learns to exploit inaccuracies in the oracle to achieve high scores with biologically implausible structures. Addressing this gap will require grounding the reward models more closely in physical reality, perhaps through uncertainty-aware ensembles or by incorporating actual wet-lab feedback via RLHF Ouyang et al. (2022a).

In Silico Evaluation versus In Vitro Validation. All reported results rely on computational proxies. We reduce dependence on the training objective by re-folding designs with independent predictors (Table 2), applying physics-based energy scoring, and evaluating sequence naturalness with a protein language model. However, predictors trained on overlapping PDB-derived data may share correlated errors, and these metrics do not establish biological activity, absolute affinity, specificity, solubility, expression yield, or stability. We therefore present our results strictly as *in silico* screening, not experimentally validated binding.

Validation should proceed through three stages: (i) explicit-solvent molecular dynamics to test interface stability (Hollingsworth & Dror, 2018); (ii) MM/PBSA or MM/GBSA screening followed by alchemical free-energy calculations (Genheden & Ryde, 2015; Wang et al., 2015); and (iii) SPR and ITC assays to measure binding affinity and kinetics (Karlsson, 2004; Wiseman et al., 1989). The first two stages could serve as higher-cost computational rewards, whereas binding assays provide the essential biological validation. Closing this loop with measured affinities is an important direction for future work.

Tree Expansion and the Exploration-Exploitation Trade-off. Second, the current iteration of TSG-Diff employs a static, deferred evaluation strategy, waiting to construct the fully expanded search tree over all L rounds before computing the terminal advantages. Rather than a mere technical limitation, this full-expansion approach is a deliberate choice prioritizing broad exploration to maintain unbiased group-relative baselines. In protein design, prematurely exploiting early high-reward branches severely risks mode collapse, i.e., generating only minor structural variations of a single binding motif rather than discovering diverse, distinct binding interfaces. By expanding the tree uniformly, TSG-Diff naturally mitigates this risk. However, this heavy exploration comes at a computational cost, as budgets are inevitably spent on unpromising branches. A highly promising direction for future work is shifting this balance toward tree exploitation with an incremental, evolutionary search Real et al. (2019), where sub-optimal branches are pruned early. The main technical hurdle will be navigating this trade-off: designing dynamic advantage estimators to stabilize gradients during asynchronous expansion, while simultaneously introducing entropy-regularized exploration terms Peters et al. (2010) to explicitly force the policy to explore disjoint regions of the functional protein space.