ATOMBLOOM: ALL-ATOM BRANCHING FLOWS FOR 3D BINDER DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Designing target-specific small molecules, peptides, and antibodies requires generating atomistic 3D structures whose cardinality, atom types, and geometry are jointly adapted to the target protein environment. Existing atom-level diffusion and flow models typically fix atom count throughout generation, while hierarchical latent models determine residue or fragment types before all-atom decoding. We introduce AtomBloom, a pocket-specific generative framework that directly generates ligand atoms and allows them to split or disappear along the generative trajectory. This formulation jointly generates atom counts, types, and 3D geometry within a unified framework for small molecules, peptides, and antibodies. For practical generation, AtomBloom combines exact interval event probabilities with dynamic rescaling that adapts the predictor's coordinate scale along the trajectory. We further introduce model-rolled self-conditioning during training and propagate previous predictions along branching lineages during inference. Across small-molecule, peptide, and antibody benchmarks, AtomBloom achieves the best overall ranking on small molecules. For peptides and antibodies, AtomBloom-R adaptively generates each residue's atoms at fixed residue count and improves sequence recovery, structural accuracy, and binding scores. Further adapting residue count yields additional binding gains, with favorable peptide designs occurring at both longer and shorter lengths than the reference. These results support treating binder cardinality as a design variable alongside atom types and geometry.

1 INTRODUCTION

Designing high-affinity binders with plausible 3D structures for a specific protein target is a central problem in drug discovery and biomolecular engineering (Peng et al., 2022; Liu et al., 2022; Schneuing et al., 2024; Guan et al., 2023; Qu et al., 2024; Kong et al., 2024; Lin et al., 2024c; Li et al., 2024; Watson et al., 2023; Luo et al., 2022; Kong et al., 2023a; Lin et al., 2024b; Martinkus et al., 2024). Binder design typically seeks a suitable molecular composition and 3D structure. Equally important is binder size, which shapes the binding interface and constrains accessible conformations. However, the appropriate size is often hard to determine in advance, as it depends on the pocket environment and the binder's structural and chemical properties. Binder size is therefore a key design variable.

Existing methods handle binder size at different levels of molecular representation. Atom-level diffusion and flow models directly generate atom types and 3D coordinates while keeping a prespecified atom count fixed throughout generation (Liu et al., 2022; Schneuing et al., 2024; Lin et al., 2025; Guan et al., 2023; Song et al., 2023). Hierarchical methods instead generate a coarse-grained structure composed of residues or molecular fragments, then decode it into an all-atom structure (Kong et al., 2024; 2025). Because different residues and fragments contain different numbers of atoms, these methods allow atom count to vary with the selected composition. Once all-atom decoding begins, however, the count remains fixed and cannot adapt to the emerging geometry. These formulations motivate an alternative in which atom count and atomistic geometry evolve jointly during generation.

Allowing atoms to be added or removed during generation provides flexibility to accommodate the target environment. Additional atoms can extend receptor contacts, whereas fewer atoms may better fit a crowded pocket. These cardinality changes reshape local interactions and packing, influencing subsequent updates to atom types and coordinates.

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

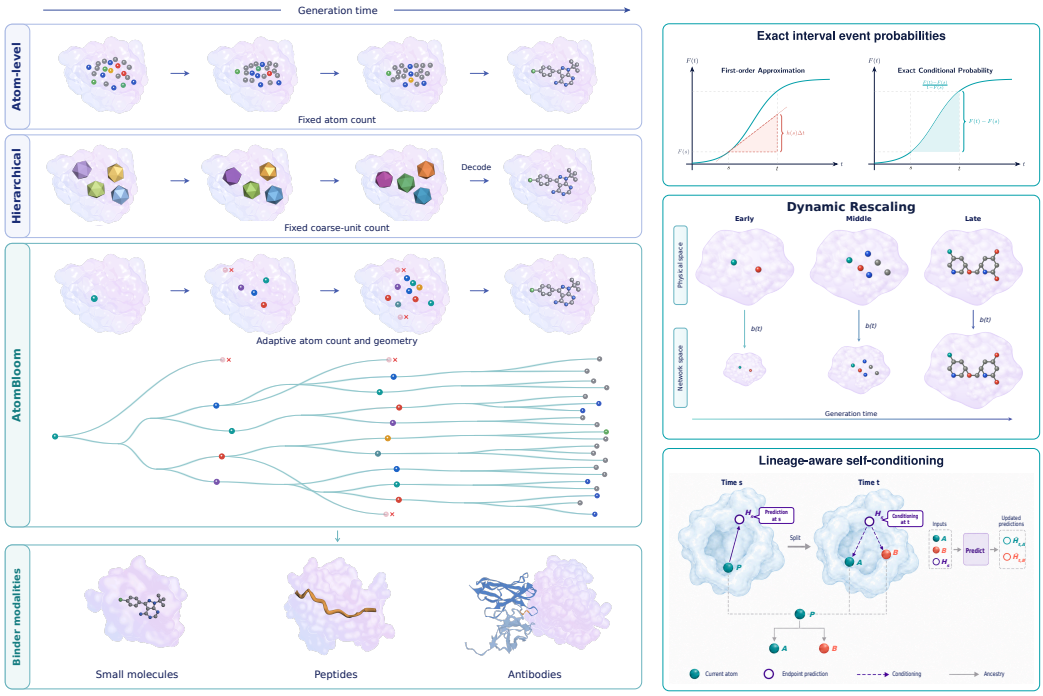


Figure 1: **Overview of AtomBloom for target-specific binder design. Left.** Unlike atom-level and hierarchical methods with fixed atom or coarse-unit counts, AtomBloom jointly adapts atom count, types, and geometry through splitting, deletion, and transport across small-molecule, peptide, and antibody design. **Right.** Exact interval event probabilities preserve cardinality accuracy, dynamic rescaling adapts the predictor's coordinate scale over time, and lineage-aware self-conditioning preserves cross-step structural context. Structures are schematic.

Realizing this joint evolution introduces three challenges. First, iterative generative processes typically discretize time into finite intervals. Once cardinality becomes part of the evolving state, atom creation and removal also need to be discretized accurately within each interval. Errors at individual intervals can accumulate into systematic deviations in the final atom count. Second, the same geometric predictor handles noisy intermediate states and fine molecular geometry at different stages of generation, motivating a coordinate scale that changes along the trajectory. Finally, atom creation and removal change cardinality and local atomic environments between steps, making earlier structural information harder to preserve in subsequent updates. Earlier predictions provide a stable reference for later steps, promoting consistency along the trajectory.

In this work, we introduce AtomBloom, an all-atom variable-cardinality generative model for target-specific binder design. As illustrated in Figure 1, we formulate the generation process with Branching Flows (Billera et al., 2026), allowing atoms to evolve along branches rather than a fixed-dimensional path. Starting from a single root in the target pocket, we expand or contract the atom set through splitting and deletion, jointly evolving cardinality and all-atom geometry within the same flow. To maintain accurate cardinality evolution under finite-step sampling, we compute exact event probabilities within each generation interval. Dynamic rescaling compresses coordinates more strongly early in generation and returns to their original scale at the endpoint. To preserve structural context across changes in the atom set, we pass each parent's structural prediction to its children through lineage-aware self-conditioning.

Our main contributions are as follows.

- We propose AtomBloom, an all-atom generative model that treats cardinality as an explicit design variable, jointly determining binder size, atom types, and coordinates.
- We derive exact interval event probabilities and introduce dynamic rescaling and lineage-aware self-conditioning to support accurate cardinality evolution, local geometry, and structural consistency during generation.

- We evaluate AtomBloom across small-molecule, peptide, and antibody design, achieving the best overall ranking on small molecules. For peptides and antibodies, AtomBloom-R improves structural recovery and binding quality at fixed residue count, while the full AtomBloom model yields further binding improvements by adapting residue count.

2 Related Work

For protein-conditioned binder design, prior models generate atoms directly (Peng et al., 2022; Guan et al., 2023; Qu et al., 2024) or couple sequence and structure through residue or fragment representations (Kong et al., 2024; Luo et al., 2022; Kong et al., 2025). Unlike the block-based UniMoMo framework (Kong et al., 2025), AtomBloom generates atoms without a learned block representation or latent decoder. To evolve cardinality alongside atom states, we build on Branching Flows (Billera et al., 2026), which introduce splitting and deletion into continuous-state generation. We develop its pocket-conditioned finite-step realization and extend self-conditioning (Chen et al., 2023) through model-rolled training and lineage-based propagation, maintaining correspondence as atoms split or disappear. Appendix A provides the full discussion.

3 AtomBloom

3.1 Problem Formulation

Let c denote the target protein environment. We represent a binder as a finite set of ligand atoms

$$\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^{N}, \qquad \mathbf{y}_i = (\mathbf{x}_i, a_i), \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}^3$ is the coordinate of atom i, $a_i \in \{1, \dots, C\}$ is its atom type, and the terminal cardinality N is not specified before generation. Our objective is to learn $p_\theta(\mathcal{Y} \mid c)$ jointly over cardinality, atom types, and coordinates.

3.2 Preliminaries on Branching Flows

Branching Flows (BF) (Billera et al., 2026) extend a base generative process to finite sets by coupling state evolution with splitting and deletion. We outline its dynamics and training objective.

3.2.1 Generative Dynamics

At time $t \in [0, 1]$, the evolving set is $\mathcal{X}_t = \{\mathbf{z}_{t,i}\}_{i=1}^{N_t}$. Let $\mathsf{S}_i\mathcal{X}$ replace element i with two descendants and let $\mathsf{D}_i\mathcal{X}$ remove it. For a test function φ, the learned infinitesimal generator is

$$\begin{aligned}(\mathcal{G}_{\theta,t}\varphi)(\mathcal{X}) = (\mathcal{B}_{\theta,t}\varphi)(\mathcal{X}) \\ + \sum_i h_{\mathrm{sp}}(t)\lambda_{\theta,i}(\mathcal{X}, t)\big[\varphi(\mathsf{S}_i\mathcal{X}) - \varphi(\mathcal{X})\big] \\ + \sum_i h_{\mathrm{del}}(t)d_{\theta,i}(\mathcal{X}, t)\big[\varphi(\mathsf{D}_i\mathcal{X}) - \varphi(\mathcal{X})\big].\end{aligned} \tag{2}$$

Here, $\mathcal{B}_{\theta,t}$ is the base generator, parameterized by a model output F_θ appropriate to the chosen base process. The quantity $\lambda_{\theta,i} \geq 0$ predicts the expected number of remaining splits for element i, while $d_{\theta,i} \in [0, 1]$ predicts whether it will be deleted.

Splitting and deletion times follow $T_{\mathrm{sp}} \sim \mu_{\mathrm{sp}}$ and $T_{\mathrm{del}} \sim \mu_{\mathrm{del}}$. For $e \in \{\mathrm{sp}, \mathrm{del}\}$, let F_e and f_e denote the corresponding cumulative distribution and density. Their hazard is $h_e(t) = f_e(t)/(1 - F_e(t))$. Starting from a tractable initial set $\mathcal{X}_0$, base dynamics update states, splitting adds elements, and deletion removes them, producing the terminal set $\mathcal{X}_1$.

3.2.2 Conditional Generator Matching

BF learns the marginal generator through conditional processes whose terminal set is observed during training. Given a terminal set $\mathcal{X}_1$, a latent construction Γ is sampled that specifies a binary coalescent forest, event times, and base-process endpoints. The terminal elements form the leaves of the forest.

To incorporate deletion into the conditional process, BF introduces auxiliary leaves whose branches terminate through deletion before $t = 1$. This construction induces a conditional path $p_{t|\Gamma}$ from which $\mathcal{X}_t$ can be sampled.

The cut of the forest at time t identifies the active elements. Let $v_i(t)$ be the active node associated with element i, and let $n(v)$ be the number of terminal and auxiliary leaves below node v. The conditional targets are

$$\lambda_i(\Gamma, t) = \max\big(n(v_i(t)) - 1, 0\big), \qquad d_i(\Gamma, t) = \mathbf{1}\{v_i(t) \text{ is auxiliary}\}, \tag{3}$$

together with a base-process target $F_i(\Gamma, t)$. These targets define the conditional generator $\mathcal{G}_{\Gamma,t}$. BF matches the learned generator to these conditional generators by minimizing

$$\mathcal{L}_{\mathrm{BF}} = \mathbb{E}[D_{\mathrm{base}}(F_\theta, F(\Gamma, t)) + D_{\mathrm{split}}(\lambda_\theta, \lambda(\Gamma, t)) + D_{\mathrm{del}}(d_\theta, d(\Gamma, t))]\,. \tag{4}$$

The model observes $\mathcal{X}_t$ and t, but not Γ. At the population optimum, it therefore recovers the conditional expectation

$$\overline{\mathcal{G}}_t(\cdot \mid \mathcal{X}_t) = \mathbb{E}[\mathcal{G}_{\Gamma,t}(\cdot \mid \mathcal{X}_t) \mid \mathcal{X}_t]\,, \tag{5}$$

which is evaluated from the current finite set during generation.

3.3 Pocket-Conditioned All-Atom Branching

We instantiate BF for structure-based binder design by representing each particle as a ligand heavy atom and conditioning the process on the fixed target pocket c. The evolving ligand is

$$\mathcal{X}_t = \{\mathbf{z}_{t,i}\}_{i=1}^{N_t}, \qquad \mathbf{z}_{t,i} = (\mathbf{x}_{t,i}, \mathbf{h}_{t,i}), \tag{6}$$

where $\mathbf{x}_{t,i} \in \mathbb{R}^3$ is an atom coordinate and $\mathbf{h}_{t,i}$ is a continuous relaxation of its one-hot atom type. Starting from one ligand particle in the pocket, the process jointly evolves atom states and cardinality.

3.3.1 Pocket-Conditioned Generator

Given the current ligand and target pocket, the predictor returns terminal atom states together with splitting and deletion outputs

$$\left(\widehat{\mathcal{X}}_{1|t}, \widehat{\boldsymbol{\ell}}_t, \widehat{\mathbf{d}}_t\right) = f_\theta(\mathcal{X}_t, c, t). \tag{7}$$

Each $\widehat{\mathbf{z}}_{1|t,i} \in \widehat{\mathcal{X}}_{1|t}$ predicts the terminal state of the current branch. The discrete-event parameters are

$$\lambda_{\theta,i}(\mathcal{X}_t, c, t) = \exp(\widehat{\ell}_{t,i}), \qquad d_{\theta,i}(\mathcal{X}_t, c, t) = \mathrm{sigmoid}(\widehat{d}_{t,i}). \tag{8}$$

For peptides and antibodies, auxiliary heads predict boundaries between adjacent atoms and amino acid identities for the resulting groups. Late in generation, we fix these groups and identities and adjust each group's atom count to match its predicted residue type. We then assign generated atoms to canonical residue graphs to determine atom identities and connectivity (Appendix E).

We use an Ornstein–Uhlenbeck (OU) base process and its bridge (Goldys & Maslowski, 2008),

$$\mathrm{d}\mathbf{Z}_\tau = -\kappa_{\mathrm{OU}}\mathbf{Z}_\tau\,\mathrm{d}\tau + \sigma_{\mathrm{OU}}\,\mathrm{d}\mathbf{W}_\tau. \tag{9}$$

Conditioning this process on the predicted terminal state gives the transition

$$\mathbf{z}_{t,i} \sim \mathbb{P}_{\mathrm{OU}}\big(\mathbf{Z}_t \in \cdot \;\big|\; \mathbf{Z}_s = \mathbf{z}_{s,i}, \mathbf{Z}_1 = \widehat{\mathbf{z}}_{1|s,i}\big)\,, \qquad s < t < 1. \tag{10}$$

We set $\kappa_{\mathrm{OU}} = 0.5$ and $\sigma_{\mathrm{OU}} = 1$, and use $\mu_{\mathrm{sp}} = \mathrm{Beta}(1, 2)$ and $\mu_{\mathrm{del}} = \mathrm{Uniform}(0, 1)$.

3.3.2 Conditional Paths from Binder Structures

Given a binder structure, the leaves of the BF conditional path are its ligand heavy atoms. A terminal leaf v_j has anchor $\mathbf{b}(v_j) = (\mathbf{x}_j, \mathbf{e}(a_j))$, where $\mathbf{e}(a_j)$ is the one-hot encoding of atom type a_j. Let $\mathcal{D}(v)$ contain all terminal and auxiliary leaves below node v. Each internal node is anchored at the mean of its descendants

$$\mathbf{b}(v) = \frac{1}{|\mathcal{D}(v)|} \sum_{v_j \in \mathcal{D}(v)} \mathbf{b}(v_j). \tag{11}$$

At a forest cut, the anchor of each active node provides its terminal atom-state target, its descendant count provides $\lambda_i(\Gamma, t)$, and auxiliary leaves provide $d_i(\Gamma, t)$. OU bridges between these anchors then define the continuous state at time t.

3.3.3 TRAINING OBJECTIVE

For active node $v_i(t)$, write $\mathbf{b}(v_i(t)) = (\mathbf{x}(v_i(t)), \mathbf{h}(v_i(t)))$. We train the terminal coordinate and atom-type predictions with

$$\mathcal{L}_{\text{pos}} = \mathbb{E}\left[\frac{\|\widehat{\mathbf{x}}_{1|t,i} - \mathbf{x}(v_i(t))\|_2^2}{3}\right], \quad \mathcal{L}_{\text{type}} = \mathbb{E}\left[\|\widehat{\mathbf{h}}_{1|t,i} - \mathbf{h}(v_i(t))\|_2^2\right]. \tag{12}$$

Event losses use the conditional-generator targets.

$$\mathcal{L}_{\text{split}} = \mathbb{E}\left[e^{\widehat{\ell}_{t,i}} - \lambda_i(\Gamma, t)\widehat{\ell}_{t,i}\right], \quad \mathcal{L}_{\text{del}} = \mathbb{E}\left[\text{BCEWithLogits}(\widehat{d}_{t,i}, d_i(\Gamma, t))\right]. \tag{13}$$

The complete objective is

$$\mathcal{L} = w_{\text{pos}}\mathcal{L}_{\text{pos}} + w_{\text{type}}\mathcal{L}_{\text{type}} + w_{\text{split}}\mathcal{L}_{\text{split}} + w_{\text{del}}\mathcal{L}_{\text{del}} + \sum_m w_m \mathcal{L}_{\text{aux},m}. \tag{14}$$

The auxiliary terms depend on the binder modality and are detailed in Appendix G. We implement f_θ with MEET (Jiao et al., 2026), an E(3)-equivariant memory-efficient transformer. Architectural and training details are in Appendix H, with complete training and sampling algorithms in Appendix C.

3.4 EXACT INTERVAL EVENT PROBABILITIES

The generator describes continuous-time dynamics; sampling uses a finite grid $0 = t_0 < t_1 < \cdots < t_L = 1$. Each interval $(s, t]$ updates cardinality before atom-state transport, using predictions at s.

The original BF finite-step kernel (Billera et al., 2026) uses the hazards at s to approximate interval event probabilities:

$$\widetilde{p}_{\text{sp}}(s,t) = h_{\text{sp}}(s)(t-s), \qquad \widetilde{p}_{\text{del}}(s,t) = 1 - \exp[-h_{\text{del}}(s)(t-s)]. \tag{15}$$

To account for hazard variation within the interval, we compute event probabilities directly from the event-time distributions. For an event $e \in \{\text{sp}, \text{del}\}$ that has not occurred before s, its probability of occurring in $(s, t]$ is

$$p_e(s,t) = \Pr(s < T_e \le t \mid T_e > s) = 1 - \frac{1 - F_e(t)}{1 - F_e(s)}. \tag{16}$$

Across adjacent intervals, these probabilities compose exactly as $1 - p_e(s,u) = (1 - p_e(s,t))(1 - p_e(t,u))$ for $s < t < u$. For each atom i at time s, write $\lambda_i = \lambda_{\theta,i}(\mathcal{X}_s, c, s)$ and $d_i = d_{\theta,i}(\mathcal{X}_s, c, s)$. With these predictions held fixed over $(s, t]$, the latent-event construction (Appendix B) gives

$$K_i \sim \text{Poisson}(\lambda_i p_{\text{sp}}(s,t)), \qquad D_i \sim \text{Bernoulli}(d_i p_{\text{del}}(s,t)). \tag{17}$$

The atom contributes $M_i = (1 - D_i)(1 + K_i)$ descendants, giving $N_t = \sum_i M_i$. The ancestry map $\pi_{s\to t} : \{1, \ldots, N_t\} \to \{1, \ldots, N_s\}$ identifies the time-s ancestor of each atom at time t.

Proposition 1. Let M_i and $\widetilde{M}_i$ be the exact and approximate descendant counts. Under our split and deletion schedules, the approximation has positive conditional bias. Writing $r = (t-s)/(1-s)$, for sufficiently small r,

$$\mathbb{E}\left[\widetilde{M}_i - M_i \mid \mathcal{F}_s\right] = \left(\lambda_{\theta,i} + \frac{d_{\theta,i}}{2}\right) r^2 + \mathcal{O}(r^3). \tag{18}$$

Appendix B proves this local bias. Figure 2(b) shows the corresponding inflation of terminal atom counts under coarse sampling.

3.5 DYNAMIC RESCALING

Early in generation, the predictor processes noisy configurations; later, it resolves fine molecular geometry. We therefore use a time-dependent coordinate scale, applying stronger compression early and returning to the original scale at the endpoint. At time t, we divide pocket-centered ligand and pocket coordinates by $b(t)$. The model predicts terminal coordinates $\widehat{\mathbf{y}}_{1|t,i}$ in this same coordinate system, and we compute the coordinate loss against targets divided by $b(t)$. We choose the scale using the OU bridge coefficients α_t, β_t, and γ_t for the initial state, terminal state, and noise:

$$b(t) = \sqrt{b_0^2\alpha_t^2 + \beta_t^2 + \gamma_t^2}, \qquad b_0 = b(0) = 10, \quad b(1) = 1. \tag{19}$$

Before OU transport, we recover $\widehat{\mathbf{x}}_{1|t,i} = b(t)\widehat{\mathbf{y}}_{1|t,i}$. Appendix F provides further details.

3.6 LINEAGE-AWARE SELF-CONDITIONING

Standard self-conditioning (Chen et al., 2023) feeds a detached clean-state prediction back into the predictor as auxiliary context. During iterative generation, earlier predictions also influence the states encountered at later steps. To expose the predictor to this dependence during training, we use a model-rolled variant with two times $s < t$. We sample these times on the same branching tree, which provides the ancestry map $\pi_{s\to t}$. Given $\mathcal{X}_s$, we first obtain a detached prediction $\widehat{\mathcal{X}}_{1|s} = \mathrm{sg}[f_\theta^{\mathrm{end}}(\mathcal{X}_s, c, s)]$. Here, f_θ^{end} denotes the terminal-state output. Keeping the sampled branching history fixed, we propagate the continuous states toward these predicted endpoints,

$$\widetilde{\mathbf{z}}_{t,j} \sim \mathbb{P}_{\mathrm{OU}}\big(\mathbf{Z}_t \in \cdot \;\big|\; \mathbf{Z}_s = \mathbf{z}_{s,\pi_{s\to t}(j)}, \mathbf{Z}_1 = \widehat{\mathbf{z}}_{1|s,\pi_{s\to t}(j)}\big)\,. \tag{20}$$

Through the same ancestry map, each descendant receives its ancestor's detached prediction as self-conditioning context H_t. We then train

$$\widehat{Y}_t = f_\theta\Big(\widetilde{\mathcal{X}}_t, c, t, H_t\Big)\,, \qquad \mathcal{L}_{\mathrm{SC}} = \mathcal{L}(\widehat{Y}_t, Y_t), \tag{21}$$

where Y_t denotes the original supervision targets at time t. This exposes the predictor to states influenced by its earlier predictions while preserving the sampled genealogy and target correspondence.

To implement H_t while preserving E(3) equivariance, we represent each time-s prediction as a virtual node at its predicted endpoint, carrying its predicted atom type and splitting information. Each time-t atom is connected to the virtual node associated with its ancestor $\pi_{s\to t}(j)$. All descendants of a time-s atom share this node, while deleted branches contribute no later conditioning context.

4 EXPERIMENTS

We evaluate AtomBloom on peptide, antibody, and small-molecule binder design, using the same variable-cardinality formulation and training one model per modality. For peptide and antibody design, we examine adaptive cardinality at two levels. For comparison with existing methods under the same residue-count constraint, we introduce AtomBloom-R, which fixes the residue count to the reference and initializes one root per residue. Each root grows a variable set of atoms, with amino acid identity determined from the evolving all-atom state. The full AtomBloom model starts from a single root and additionally allows residue count to emerge during generation. AtomBloom-R evaluates adaptive atom generation under the reference residue-count constraint, while its comparison with AtomBloom assesses the additional flexibility of adaptive residue counts.

Data and Splits We evaluate peptides on 93 LNR complexes, antibodies on 60 RAbD complexes, and small molecules on 100 CrossDocked2020 complexes, following prior splits and protocols (Kong et al., 2025; Peng et al., 2022). Appendix I details the data, metrics, and baselines.

4.1 *De Novo* PEPTIDE DESIGN

Setup. We evaluate on LNR, following UniMoMo (Kong et al., 2025). We report amino-acid recovery (AAR), Cα RMSD after target-complex and ligand self-alignment (C-RMSD and L-RMSD), Rosetta interface energy (ΔG), improvement over the native binder (IMP), and local geometry. Baselines are RFdiffusion, PepFlow, PepGLAD, and UniMoMo (Appendix I).

Results. Table 1 examines two levels of adaptive cardinality in peptide generation. *(1) Given a fixed residue count, can determining amino-acid types from evolving all-atom structures improve binder design?* UniMoMo determines residue types before all-atom decoding, thereby fixing the corresponding atom counts. AtomBloom-R instead determines residue types from atom sets that evolve in both cardinality and geometry. Under the same residue-count constraint, AtomBloom-R improves amino-acid recovery, structural accuracy, Rosetta interface energy, and local geometry over UniMoMo (all). It reduces L-RMSD from 1.27 to 0.76 Å and increases IMP from 40.86% to 70.97%. These results show that AtomBloom-R improves structure recovery and binding scores under the same residue-count constraint. *(2) Can the constraint on residue count be further relaxed?* Allowing residue count to adapt further increases IMP from 70.97% to 79.57% and lowers predicted interface energies, while preserving reference-domain structural recovery and increasing sequence and structural diversity. Appendix K further supports the value of exploring non-reference lengths to discover lower-energy candidates.

Table 1: Peptide binder design on 93 LNR targets at Sample@100. AtomBloom-R fixes the reference residue count while adaptively generating each residue's atoms. AtomBloom additionally adapts residue count. Bold and underline denote the best and second-best results.

Model	AAR (%) ↑	C-RMSD (Å) ↓	L-RMSD (Å) ↓	ΔG ↓	IMP (%) ↑	Clash_{in} ↓	$\text{Clash}_{\text{out}}$ ↓	JSD_{bb} ↓	JSD_{sc} ↓	Seq. Div. ↑	Struct. Div. ↑
Reference	–	–	–	-37.25	–	0.31%	0.88%	–	–	–	–
RFdiffusion	34.68	4.69	1.88	-13.47	5.38	**0.06%**	13.58%	0.273	0.798	0.155	0.616
PepFlow	35.47	2.87	1.79	-21.71	15.22	2.72%	4.62%	0.240	0.693	0.530	0.507
PepGLAD	38.62	2.74	1.60	-23.12	18.28	1.82%	1.66%	0.474	0.398	**0.687**	**0.698**
UniMoMo (single)	37.59	2.48	1.48	-28.72	29.03	1.53%	0.94%	0.390	0.365	0.626	0.629
UniMoMo (all)	39.45	2.19	1.27	-34.35	40.86	0.45%	0.93%	0.205	0.180	0.617	0.573
AtomBloom-R	**42.88**	**1.92**	**0.76**	-43.94	70.97	0.08%	0.44%	0.109	0.156	0.189	0.103
AtomBloom	40.55	1.18†	0.69†	**-45.38**	**79.57**	0.13%	**0.38%**	**0.102**	**0.155**	0.346	0.158

Table 2: CDR-H3 recovery and structural plausibility on 60 RAbD targets. Recovery is reported at 1, 10, and 100 samples. AtomBloom-R uses the reference CDR-H3 length. Structural rationality is evaluated at Sample@100. The reference Clash_{in} and $\text{Clash}_{\text{out}}$ are 0.08% and 0.02%. Bold and underline denote the best and second-best results.

Model	Sample@1			Sample@10			Sample@100			Structural Plausibility			
	AAR ↑	RMSD ↓	IMP ↑	AAR ↑	RMSD ↓	IMP ↑	AAR ↑	RMSD ↓	IMP ↑	Clash_{in} ↓	$\text{Clash}_{\text{out}}$ ↓	JSD_{bb} ↓	JSD_{sc} ↓
MEAN	29.13	1.87	6.67	–	–	–	–	–	–	0.96%	0.16%	0.529	–
dyMEAN	31.65	8.21	11.86	–	–	–	–	–	–	1.02%	2.98%	0.542	0.702
GeoAB-R	**32.04**	**1.67**	6.67	–	–	–	–	–	–	0.59%	0.11%	0.529	–
DiffAb	24.60	2.77	10.34	38.42	2.08	34.48	49.74	1.46	60.34	0.31%	0.25%	0.268	–
GeoAB-D	29.74	1.73	6.67	38.20	1.58	20.00	45.96	1.50	40.00	0.75%	0.07%	0.430	–
UniMoMo (single)	20.44	2.71	15.00	39.04	1.90	35.00	48.78	1.39	63.33	0.25%	0.06%	0.278	0.284
UniMoMo (all)	21.44	2.52	13.33	42.05	1.44	41.67	52.34	1.04	65.00	0.18%	0.03%	0.224	**0.221**
AtomBloom-R	30.52	1.74	13.33	**45.50**	**1.19**	**50.00**	**54.29**	**0.83**	71.67	**0.10%**	**0.03%**	0.182	0.231
AtomBloom	20.01	1.99†	**16.67**	37.54	1.30†	41.67	48.15	0.92†	**78.33**	0.13%	0.04%	**0.180**	0.235

4.2 Antibody Design

Setup. We evaluate CDR-H3 design following UniMoMo (Kong et al., 2025), reporting AAR, RMSD, and IMP at Sample@1, Sample@10, and Sample@100, alongside clash rates and dihedral JSDs. Baselines include MEAN, dyMEAN, GeoAB, DiffAb, and UniMoMo (Appendix I).

Results. Table 2 supports the peptide-design findings. With residue count fixed, AtomBloom-R achieves the strongest sequence recovery at larger sampling budgets and substantially reduces RMSD over existing generative methods. Allowing residue count to adapt enables AtomBloom to find candidates with lower ΔG than the native binder for more targets. Low RMSDs and clash rates show that this flexibility preserves structural recovery and local plausibility.

4.3 Small-Molecule Design

Setup. We follow CBGBench (Lin et al., 2024d) and compare with 14 methods spanning autoregressive, diffusion, fragment, voxel, Bayesian-flow, and latent models. The overall score aggregates ranks for substructures, chemical properties, geometry, and interactions, with interactions weighted twice as heavily (Kong et al., 2025). Appendix I lists baselines and metric definitions.

Results. Small-molecule design tests whether AtomBloom extends beyond residue-based binders to more diverse structures while preserving chemical plausibility and favorable target interactions.

Table 3: Overall CBGBench ranking over 15 methods. Higher weighted score and lower rank are better.

Model	Substructure ↓	Chemical ↓	Interaction ↓	Geometry ↓	Weighted Score ↑	Overall ↓
LIGAN	9.33	7.75	**3.58**	8.75	8.40	7
3DSBDD	9.33	6.50	9.08	11.50	5.90	10
GraphBP	14.17	7.00	14.08	14.50	2.23	15
Pocket2Mol	11.33	8.25	7.67	11.50	5.72	11
TargetDiff	6.00	7.00	6.08	6.50	8.67	6
DiffSBDD	11.17	5.75	11.92	10.25	4.80	13
DiffBP	13.67	9.00	9.58	8.25	4.98	12
FLAG	11.50	7.50	11.25	12.00	4.30	14
D3FG	7.67	3.25	10.25	11.50	6.42	9
DecompDiff	5.50	5.75	8.58	5.75	8.17	8
MolCRAFT	5.00	7.00	4.67	3.50	10.03	3
VoxBind	7.00	4.25	5.00	5.00	9.75	4
UniMoMo (single)	3.67	3.50	8.17	4.75	9.35	5
UniMoMo (all)	**2.33**	**3.00**	6.08	3.50	10.80	2
AtomBloom	**2.33**	4.25	4.67	**2.75**	**11.27**	**1**

Table 4 reports key metrics, with category ranks in Table 3 and full metrics in Appendix N. AtomBloom achieves the best aggregate ranks in substructure modeling and molecular geometry, with accurate ring-type and functional-group distributions, low bond-length divergence, and few

†For variable-length outputs, RMSD is computed over the reference domain using the order-preserving correspondence in Appendix J. These values provide a descriptive measure of reference-region agreement and are excluded from the best/second-best ranking of fixed-length methods.

Table 4: Key small-molecule metrics. FG denotes functional group. Energies are in kcal/mol. Bold and underline denote the best and second-best results. Full metrics are in Table 10.

Model	Substructure JSD ↓			Chemical		Geometry		Vina Score		Vina Dock	
	Atom Type	Ring Type	FG Type	QED ↑	SA ↑	JSD_{BL} ↓	$Ratio_{cca}$ ↓	E ↓	IMP (%) ↑	E ↓	IMP (%) ↑
LIGAN	0.1167	0.3163	0.2468	0.46	0.66	0.4645	0.0096	**-6.47**	**62.13**	-7.70	**72.71**
3DSBDD	0.0860	0.3188	0.2682	0.48	0.63	0.5024	0.2482	–	3.99	-6.45	31.46
GraphBP	0.1642	0.5061	0.6259	0.44	0.64	0.5182	0.8634	–	0.00	-4.57	10.86
Pocket2Mol	0.0916	0.3550	0.2961	0.39	0.65	0.5433	0.0576	-5.23	31.06	-7.05	48.07
TargetDiff	0.0533	0.2345	0.2876	0.49	0.60	0.2659	0.0483	-5.71	38.21	-7.41	51.99
DiffSBDD	0.0529	0.3853	0.5520	0.49	0.34	0.3501	0.1083	–	12.67	-5.53	29.76
DiffBP	0.2591	0.4531	0.5346	0.47	0.59	0.3453	0.0449	–	8.60	-7.34	49.24
FLAG	0.1032	0.2432	0.3634	0.41	0.58	0.4215	0.6777	–	0.04	-3.65	11.78
D3FG	0.0644	0.1869	0.2511	0.49	0.66	0.3727	0.2115	–	3.70	-6.78	28.9
DecompDiff	0.0431	0.2431	0.1916	0.49	0.66	0.2576	0.0462	-5.18	19.66	-7.10	48.31
MolCRAFT	0.0490	0.2469	0.1196	0.48	0.66	**0.2250**	0.0264	-6.15	54.25	**-7.79**	56.22
VoxBind	0.0942	0.2401	**0.1053**	0.54	0.65	0.2701	0.0103	-6.16	41.80	-7.68	52.91
UniMoMo (single)	0.0352	0.1252	0.1562	0.53	0.68	0.3362	0.0042	-5.50	26.37	-7.19	49.60
UniMoMo (all)	**0.0280**	0.1073	0.1356	**0.55**	0.70	0.3223	**0.0040**	-5.72	30.40	-7.25	51.59
AtomBloom	0.0379	**0.1047**	0.1275	0.53	**0.73**	0.2572	0.0045	-6.35	43.66	-7.48	58.44

steric clashes. It also achieves the best synthetic accessibility, competitive drug-likeness and Vina energies, and one of the highest docking improvement rates. Together, these results yield the best overall ranking and show that adaptive atom count is compatible with chemical plausibility, geometric quality, and favorable target interactions.

5 Analysis

5.1 Ablation Studies

Adaptive residue count. We examine whether favorable peptides differ in length from the reference binder. Figure 2(a) shows their distribution per target, together with the reference count and the count of the lowest-ΔG candidate among 100 samples. Across 93 targets, this candidate is longer than the reference for 66.7% of targets, shorter for 21.5%, and equal in length for 11.8%. Favorable candidates occur at both longer and shorter lengths, and the lowest-energy candidate is also the longest for only 6 of 93 targets. Further diagnostics show a mild within-target length preference and frequent receptor contacts by extra residues, supporting adaptive-length search (Appendix K).

Finite-step cardinality accuracy. We compare exact interval event probabilities with the first-order hazard approximation using the same transport and refinement procedure. Figure 2(b) shows that the approximation inflates generated atom count under coarse sampling, whereas exact probabilities maintain more consistent cardinality across inference budgets. The gap widens with fewer sampling steps, consistent with the finite-step bias in Proposition 1.

Dynamic rescaling. To test whether the coordinate scale should change during generation, we compare dynamic rescaling with fixed scales of $b = 1$ and 10 (Figure 2(c)). Dynamic rescaling improves binding performance over both alternatives across all three modalities, while preserving small-molecule bond-length quality and improving peptide and antibody backbone geometry. These gains favor adapting the scale along the trajectory over keeping either the early or terminal scale throughout generation. Raw results are in Appendix M.

Self-conditioning. We examine how self-conditioning affects the binding quality and diversity of generated candidates. Figure 2(d) shows that self-conditioning shifts small-molecule Vina scores and peptide interface energies toward more favorable values. Figure 2(e) provides a more detailed comparison for peptides, covering both binding quality and diversity. Self-conditioning improves both ΔG and IMP while reducing sequence and structural diversity, suggesting that it concentrates generation on candidates with more favorable binding energies. Note that disabling self-conditioning still yields strong binding performance with greater diversity, offering an alternative when diversity is a priority. Complete results are in Table 6. Appendix L.2 further examines model-rolled training.

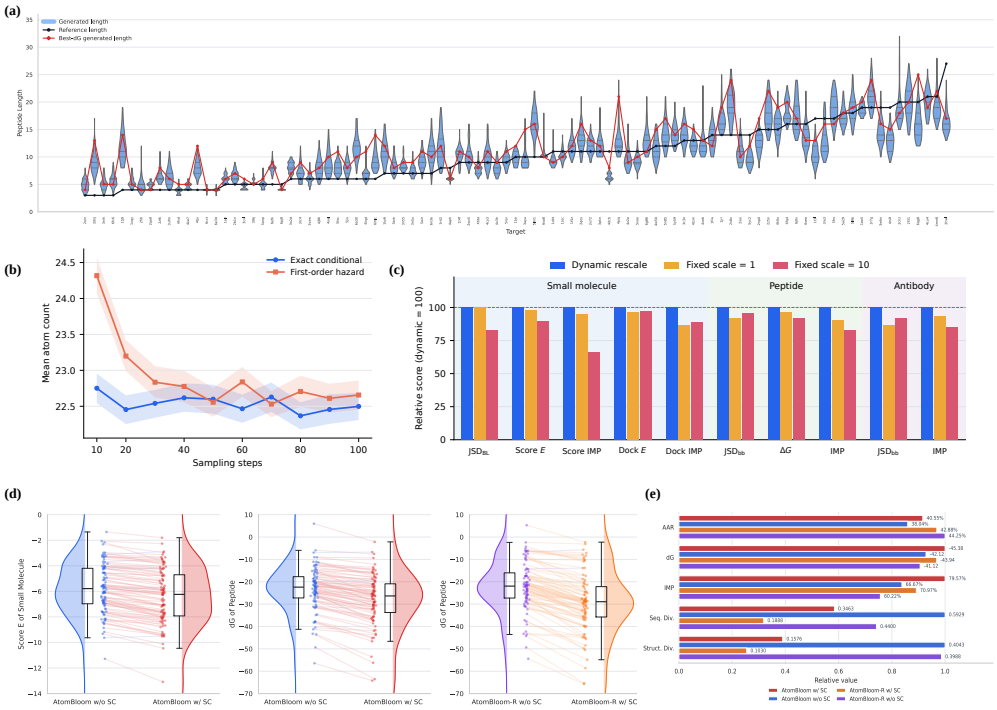


Figure 2: (a) Peptide length distributions. (b) Generated atom count with exact event probabilities or a first-order hazard approximation. (c) Dynamic versus fixed rescaling across modalities (dynamic = 100, higher is better). (d) Binding-score distributions with and without self-conditioning. (e) Peptide metrics with and without self-conditioning.

5.2 Case Studies

Figure 3 shows two peptide designs with non-reference lengths generated by AtomBloom. In 2B1N, the AtomBloom design achieves a lower predicted interface energy than both the reference and fixed-length designs. Compared with the reference and AtomBloom-R designs, the AtomBloom binder forms contacts with an additional receptor region and achieves a larger buried interface area, indicating that the length change is used to expand the binding footprint.

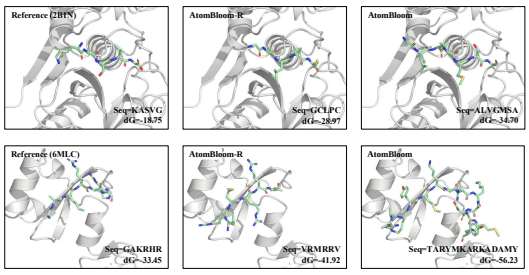


Figure 3: Case studies of adaptive binder length.

In 6MLC, adaptive length enables a different binding mode. The reference and AtomBloom-R peptides primarily occupy a local receptor groove. In contrast, the longer AtomBloom-generated binder extends along the receptor surface, with its C-terminus folding back onto another surface patch rather than into solvent. This surface-spanning pattern engages more receptor residues and substantially increases the buried interface area.

6 Conclusion

We introduced AtomBloom, an all-atom branching-flow framework that jointly generates binder size, atom types, and geometry. Exact interval event probabilities, dynamic rescaling, and lineage-aware self-conditioning make variable-cardinality generation effective under finite sampling. Across small molecules, peptides, and antibodies, AtomBloom achieves strong structural and binding performance. Its adaptive-length designs explore favorable binding configurations beyond reference-count generation, supporting binder size as an integral part of the molecular design space.

AI USE STATEMENT

AI tools were used to polish the writing and assist in creating the main overview figure. The authors reviewed and take responsibility for the final text and figure.

REFERENCES

Jared Adolf-Bryfogle, Oleks Kalyuzhniy, Michael Kubitz, Brian D Weitzner, Xiaozhen Hu, Yumiko Adachi, William R Schief, and Roland L Dunbrack Jr. Rosettaantibodydesign (rabd): A general framework for computational antibody design. *PLoS computational biology*, 14(4):e1006112, 2018.

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.

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Lukas Billera, Hedwig Nora Nordlinder, Jack Collier Ryder, Anton Oresten, Aron Stålmarck, Theodor Mosetti Björk, and Ben Murrell. Branching flows: Discrete, continuous, and manifold flow matching with splits and deletions. 2026. URL https://arxiv.org/abs/2511.09465.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. 2023. URL https://arxiv.org/abs/2208.04202.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1:1–11, 2009.

Beniamin Goldys and Bohdan Maslowski. The Ornstein–Uhlenbeck bridge and applications to Markov semigroups. *Stochastic Processes and their Applications*, 118(10):1738–1767, 2008. doi: 10.1016/j.spa.2007.10.010.

Jiaqi Guan, Xiangxin Zhou, Yuwei Yang, Yu Bao, Jian Peng, Jianzhu Ma, Qiang Liu, Liang Wang, and Quanquan Gu. Decompdiff: Diffusion models with decomposed priors for structure-based drug design. In *International Conference on Machine Learning*, pp. 11827–11846. PMLR, 2023.

Rui Jiao, Xiangzhe Kong, Yinjun Jia, Yijia Zhang, Ziyi Yang, Yang Liu, and Jianzhu Ma. Scalable peptide design via memory-efficient equivariant transformer. 2026. URL https://arxiv.org/abs/2606.25006.

Xiangzhe Kong, Wenbing Huang, and Yang Liu. Conditional antibody design as 3d equivariant graph translation. In *The Eleventh International Conference on Learning Representations*, 2023a.

Xiangzhe Kong, Wenbing Huang, and Yang Liu. End-to-end full-atom antibody design. In *International Conference on Machine Learning*, pp. 17409–17429. PMLR, 2023b.

Xiangzhe Kong, Yinjun Jia, Wenbing Huang, and Yang Liu. Full-atom peptide design with geometric latent diffusion. *arXiv preprint arXiv:2402.13555*, 2024.

Xiangzhe Kong, Zishen Zhang, Ziting Zhang, Rui Jiao, Jianzhu Ma, Wenbing Huang, Kai Liu, and Yang Liu. Unimomo: Unified generative modeling of 3d molecules for de novo binder design. 2025. URL https://arxiv.org/abs/2503.19300.

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 *Forty-first International Conference on Machine Learning*, 2024.

Haitao Lin, Yufei Huang, Odin Zhang, Yunfan Liu, Lirong Wu, Siyuan Li, Zhiyuan Chen, and Stan Z Li. Functional-group-based diffusion for pocket-specific molecule generation and elaboration. *Advances in Neural Information Processing Systems*, 36, 2024a.

Haitao Lin, Lirong Wu, Yufei Huang, Yunfan Liu, Odin Zhang, Yuanqing Zhou, Rui Sun, and Stan Z Li. Geoab: Towards realistic antibody design and reliable affinity maturation. In *Forty-first International Conference on Machine Learning*, 2024b.

Haitao Lin, Odin Zhang, Huifeng Zhao, Dejun Jiang, Lirong Wu, Zicheng Liu, Yufei Huang, and Stan Z Li. Ppflow: Target-aware peptide design with torsional flow matching. *bioRxiv*, pp. 2024–03, 2024c.

Haitao Lin, Guojiang Zhao, Odin Zhang, Yufei Huang, Lirong Wu, Zicheng Liu, Siyuan Li, Cheng Tan, Zhifeng Gao, and Stan Z Li. Cbgbench: Fill in the blank of protein-molecule complex binding graph. *arXiv preprint arXiv:2406.10840*, 2024d.

Haitao Lin, Yufei Huang, Odin Zhang, Siqi Ma, Meng Liu, Xuanjing Li, Lirong Wu, Jishui Wang, Tingjun Hou, and Stan Z Li. Diffbp: Generative diffusion of 3d molecules for target protein binding. *Chemical Science*, 2025.

Meng Liu, Youzhi Luo, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. Generating 3d molecules for target protein binding. In *International Conference on Machine Learning*, pp. 13912–13924. PMLR, 2022.

Shitong Luo, Jiaqi Guan, Jianzhu Ma, and Jian Peng. A 3d generative model for structure-based drug design. *Advances in Neural Information Processing Systems*, 34:6229–6239, 2021.

Shitong Luo, Yufeng Su, Xingang Peng, Sheng Wang, Jian Peng, and Jianzhu Ma. Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures. *Advances in Neural Information Processing Systems*, 35:9754–9767, 2022.

Karolis Martinkus, Jan Ludwiczak, Wei-Ching Liang, Julien Lafrance-Vanasse, Isidro Hotzel, Arvind Rajpal, Yan Wu, Kyunghyun Cho, Richard Bonneau, Vladimir Gligorijevic, et al. Abdiffuser: full-atom generation of in-vitro functioning antibodies. *Advances in Neural Information Processing Systems*, 36, 2024.

Xingang Peng, Shitong Luo, Jiaqi Guan, Qi Xie, Jian Peng, and Jianzhu Ma. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. In *International Conference on Machine Learning*, pp. 17644–17655. PMLR, 2022.

Pedro O Pinheiro, Arian Rokkum Jamasb, Omar Mahmood, Vishnu Sresht, and Saeed Saremi. Structure-based drug design by denoising voxel grids. In *Forty-first International Conference on Machine Learning*, 2024.

Yanru Qu, Keyue Qiu, Yuxuan Song, Jingjing Gong, Jiawei Han, Mingyue Zheng, Hao Zhou, and Wei-Ying Ma. Molcraft: Structure-based drug design in continuous parameter space. In *Forty-first International Conference on Machine Learning*, 2024.

Sebastian Salentin, Sven Schreiber, V Joachim Haupt, Melissa F Adasme, and Michael Schroeder. Plip: fully automated protein–ligand interaction profiler. *Nucleic acids research*, 43(W1):W443–W447, 2015.

Arne Schneuing, Charles Harris, Yuanqi Du, Kieran Didi, Arian Jamasb, Ilia Igashov, Weitao Du, Carla Gomes, Tom L Blundell, Pietro Lio, et al. Structure-based drug design with equivariant diffusion models. *Nature Computational Science*, 4(12):899–909, 2024.

Yuxuan Song, Jingjing Gong, Minkai Xu, Ziyao Cao, Yanyan Lan, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Equivariant flow matching with hybrid probability transport for 3d molecule generation. *Advances in Neural Information Processing Systems*, 36:549–568, 2023.

Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

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.

A Extended Related Work

Protein-conditioned binder design. Equivariant diffusion, flow, autoregressive, and latent models generate small-molecule coordinates and types in protein pockets (Luo et al., 2021; Liu et al., 2022; Schneuing et al., 2024; Guan et al., 2023; Peng et al., 2022; Qu et al., 2024). Peptide and antibody methods additionally couple sequence with atomistic structure through residue- or fragment-level latent variables, geometric flows, or structural refinement (Kong et al., 2024; 2025; Lin et al., 2024c; Li et al., 2024; Luo et al., 2022; Kong et al., 2023a; Lin et al., 2024b). UniMoMo (Kong et al., 2025) provides a unified block representation and the benchmarks used here. AtomBloom instead generates ligand atoms directly without a learned block representation or latent decoder.

Variable cardinality and self-conditioning. Autoregressive construction and fixed-capacity masked representations determine cardinality through generation order or slot occupancy. Branching Flows (Billera et al., 2026) instead couple continuous-state transport with a binary forest whose split and deletion events change cardinality. AtomBloom adapts this process to protein-conditioned atom-level generation and develops its finite-step realization. Self-conditioning reuses earlier clean-sample estimates (Chen et al., 2023), but standard formulations assume fixed state correspondence. AtomBloom combines model-rolled training with lineage-based propagation so that conditioning remains defined as atoms split or disappear.

B Finite-Step Analysis of Branching Events

Sampling from continuous event-time laws with finitely many network evaluations requires converting these laws into finite-interval transition probabilities. Under the latent-event construction below, the survival-ratio factor used by AtomBloom gives the exact finite-interval law for split and deletion events. For the split schedule used in our experiments, we further quantify the error introduced by a first-order hazard approximation.

B.1 Exact event realization over a finite interval

AtomBloom is an adaptive finite-step sampler. At each interval boundary, the predictor is evaluated on the current state, and its outputs parameterize the conditional transition over the next interval. Let $\mathcal{F}_s$ denote the generated state and target context available at the boundary time s. Conditioned on $\mathcal{F}_s$, the predicted event parameters are fixed quantities for the transition from s to t, but they may change after every state update along the trajectory.

Let F_b be the CDF of a split time and $S_b = 1 - F_b$ its survival function. Given that an event has not occurred before s, its probability of occurring by t is

$$\rho_{s,t} = \Pr(T_b \in [s,t] \mid T_b \geq s) = 1 - \frac{S_b(t)}{S_b(s)}. \tag{22}$$

To connect this event probability to offspring counts, consider an active node i with predicted Poisson mean

$$\lambda_i = \exp(\widehat{\ell}_{s,i}). \tag{23}$$

Suppose that its latent number of potential additional offspring is

$$U_i \mid \mathcal{F}_s \sim \text{Poisson}(\lambda_i). \tag{24}$$

Independently assign each potential offspring a split time from the branch-time law conditioned on survival to s. The number realized in $[s,t]$ is

$$K_i(s,t) = \sum_{j=1}^{U_i} \mathbf{1}\{T_{ij} \in [s,t]\}. \tag{25}$$

Lemma 1 (exact conditional thinning). Under this latent-event construction,

$$K_i(s,t) \mid \mathcal{F}_s \sim \text{Poisson}\left(\lambda_i \rho_{s,t}\right). \tag{26}$$

Proof. Conditional on U_i, each potential offspring is retained independently with probability $\rho_{s,t}$, so

$$K_i(s,t) \mid U_i, \mathcal{F}_s \sim \mathrm{Binomial}(U_i, \rho_{s,t}). \tag{27}$$

Writing $\rho = \rho_{s,t}$, its conditional probability generating function is

$$\mathbb{E}[z^{K_i} \mid \mathcal{F}_s] = \mathbb{E}[(1-\rho+\rho z)^{U_i} \mid \mathcal{F}_s] \tag{28}$$

$$= \exp(\lambda_i \rho(z-1)), \tag{29}$$

which is the probability generating function of $\mathrm{Poisson}(\lambda_i \rho)$. □

It follows immediately that

$$\mathbb{E}[K_i(s,t) \mid \mathcal{F}_s] = \mathrm{Var}(K_i(s,t) \mid \mathcal{F}_s) = \lambda_i \rho_{s,t}. \tag{30}$$

Thus, the survival-ratio factor thins the predicted offspring count by the exact probability that each latent split time falls in the interval.

Deletion has the analogous construction. Let $d_i = \sigma(\widehat{d}_{s,i})$ be the predicted deletion propensity, and let S_d be the survival function of the deletion time. Conditional on $\mathcal{F}_s$, let the latent deletion indicator be distributed as $\mathrm{Bernoulli}(d_i)$ and, when it equals one, draw its event time independently from the deletion-time law conditioned on survival to s. Then

$$D_i(s,t) \mid \mathcal{F}_s \sim \mathrm{Bernoulli}\left(d_i \eta_{s,t}\right), \qquad \eta_{s,t} = 1 - \frac{S_d(t)}{S_d(s)}. \tag{31}$$

Hence the split and deletion rules in Eq. (17) apply the same conditional-event principle to Poisson and Bernoulli latent events, respectively.

Remark 1 (consistency under interval subdivision). The survival-ratio construction remains consistent when an interval is subdivided. For a partition

$$s = t_0 < t_1 < \cdots < t_m = t, \tag{32}$$

define

$$\rho_k = 1 - \frac{S_b(t_{k+1})}{S_b(t_k)}. \tag{33}$$

The probability that a fixed potential event survives every subinterval is

$$\prod_{k=0}^{m-1} (1-\rho_k) = \prod_{k=0}^{m-1} \frac{S_b(t_{k+1})}{S_b(t_k)} = \frac{S_b(t)}{S_b(s)}. \tag{34}$$

Consequently, the probability that it occurs somewhere in $[s,t]$ is exactly $\rho_{s,t}$, independent of the chosen partition. The same telescoping argument applies to deletion. Thus, the survival-ratio factors preserve the prescribed event-time law under interval subdivision. This consistency follows the same latent event across the partition.

B.2 Bias of the first-order hazard approximation

If the split hazard h_b is twice continuously differentiable near s, the exact conditional split probability is

$$\rho_{s,s+\Delta} = 1 - \exp\left(-\int_s^{s+\Delta} h_b(u)\,\mathrm{d}u\right). \tag{35}$$

Expanding around s gives

$$\rho_{s,s+\Delta} = h_b(s)\Delta + \frac{h_b'(s) - h_b(s)^2}{2}\Delta^2 + O(\Delta^3). \tag{36}$$

The factor $h_b(s)\Delta$ is therefore only a first-order approximation to the interval probability.

AtomBloom uses the branch-time schedule $T_b \sim \mathrm{Beta}(1,2)$, for which

$$S_b(t) = (1-t)^2, \qquad h_b(t) = \frac{2}{1-t}. \tag{37}$$

The exact and first-order interval factors are

$$\rho_{s,t} = 1 - \left(\frac{1-t}{1-s}\right)^2, \qquad \widetilde{\rho}_{s,t} = \frac{2(t-s)}{1-s}. \tag{38}$$

Corollary 1 (bias of the first-order split factor). For $0 \le s < t < 1$, the first-order hazard factor exceeds the exact interval probability by

$$\widetilde{\rho}_{s,t} - \rho_{s,t} = \left(\frac{t-s}{1-s}\right)^2. \tag{39}$$

For a fixed predicted mean λ_i, the corresponding excess in the expected number of additional offspring is

$$\mathbb{E}[\widetilde{K}_i \mid \mathcal{F}_s] - \mathbb{E}[K_i \mid \mathcal{F}_s] = \lambda_i \left(\frac{t-s}{1-s}\right)^2. \tag{40}$$

The error grows quadratically with the interval length relative to the remaining time horizon and vanishes as the step size tends to zero. Therefore, coarse first-order updates systematically inflate the expected split count.

B.3 Joint split and deletion bias

The atom-count ablation in Figure 2(b) uses a linear split factor and replaces the exact deletion factor by a frozen-hazard exponential factor. AtomBloom uses a uniform deletion-time distribution, for which, writing $r = (t-s)/(1-s)$,

$$\eta_{s,t} = r, \qquad \widetilde{\eta}_{s,t} = 1 - e^{-r}. \tag{41}$$

Since $1 - e^{-r} < r$ for $r > 0$, this approximation underestimates the deletion probability. We compare the expected number of active descendants contributed by one current node under the exact and approximate kernels. Define

$$\mu_i = \lambda_i \rho_{s,t}, \quad \widetilde{\mu}_i = \lambda_i \widetilde{\rho}_{s,t}, \quad p_i = d_i \eta_{s,t}, \quad \widetilde{p}_i = d_i \widetilde{\eta}_{s,t}, \tag{42}$$

where $d_i = \sigma(\widehat{d}_{s,i})$. In the conditional interval kernel, split and deletion are independent given $\mathcal{F}_s$, and deletion takes precedence over offspring expansion. The number of active nodes contributed by node i at time t is therefore

$$M_i = (1 - D_i)(1 + K_i), \tag{43}$$

with conditional expectation

$$\mathbb{E}[M_i \mid \mathcal{F}_s] = (1 - p_i)(1 + \mu_i). \tag{44}$$

Proof of Proposition 1. Let $\widetilde{M}_i$ denote the corresponding node contribution under the linear split and exponential deletion factors. For $0 \le s < t < 1$,

$$\begin{aligned}
&\mathbb{E}[\widetilde{M}_i \mid \mathcal{F}_s] - \mathbb{E}[M_i \mid \mathcal{F}_s] \\
&\quad = (1 - \widetilde{p}_i)(\widetilde{\mu}_i - \mu_i) + (p_i - \widetilde{p}_i)(1 + \mu_i) \\
&\quad = (1 - d_i(1 - e^{-r}))\lambda_i r^2 + d_i\big(r - (1 - e^{-r})\big)(1 + \lambda_i \rho_{s,t}) \;>\; 0.
\end{aligned} \tag{45}$$

Since $\lambda_i > 0$, both terms are nonnegative and the split term is strictly positive for $r > 0$, proving the positive bias in Proposition 1. Furthermore, $r - (1 - e^{-r}) = r^2/2 + O(r^3)$, and therefore

$$\mathbb{E}[\widetilde{M}_i - M_i \mid \mathcal{F}_s] = \left(\lambda_i + \frac{d_i}{2}\right) r^2 + O(r^3), \tag{46}$$

which proves Eq. (18). □

The first term is the excess split count among nodes retained by the approximate deletion rule. The second is the parent and its expected offspring retained because deletion is underestimated. Thus, for the schedules used by AtomBloom, the two approximation errors act in the same direction and both increase the expected one-step cardinality. This joint effect matches the direction of the atom-count distortion observed in Figure 2(b).

C Complete Training and Generation Algorithms

Let $\mathcal{T}$ denote a sampled coalescent forest and $\pi_{s\to t}$ its ancestry map at two forest cuts. Each call to f_θ applies dynamic rescaling at its evaluation time and returns coordinate predictions in physical units (Section 3.5). Coordinate supervision uses the rescaled loss in Eq. (62). For a collection of predictions A, the notation $\pi_{s\to t}(A)$ assigns $A_{\pi_{s\to t}(j)}$ to descendant j. The auxiliary outputs $\widehat{\gamma}$ and $\widehat{\mathbf{q}}$ are residue-boundary and residue-type logits for peptides and antibodies (Appendix G).

Algorithm 1 Training Algorithm of AtomBloom

Input: dataset $\mathcal{S}$, self-conditioning probability p_{SC}
Output: predictor f_θ

Initialize θ and moving-average parameters $\bar{\theta}$
while not converged **do**
  $(c, \mathcal{X}_1) \sim \mathcal{S}, \quad \mathcal{T} \sim p(\mathcal{T} \mid \mathcal{X}_1)$
  $t \sim \mathrm{Uniform}(0,1), \quad (\mathcal{X}_t, Y_t) \sim q_{\mathcal{T}}(\cdot \mid \mathcal{X}_1, t)$
  $H_t \leftarrow \varnothing$
  if $u < p_{\mathrm{SC}}$, $u \sim \mathrm{Uniform}(0,1)$, **then**
    Sample paired times (s,t) as in Appendix H and draw $(\mathcal{X}_s, \mathcal{X}_t, Y_t, \pi_{s\to t})$ from the same tree $\mathcal{T}$
    $\widehat{\mathcal{X}}_{1|s} \leftarrow \mathrm{stopgrad}(f_\theta^{\mathrm{end}}(\mathcal{X}_s, c, s))$ Section 3.6
    $\mathcal{X}_t \leftarrow q_{s\to t}(\cdot \mid \mathcal{X}_s, \widehat{\mathcal{X}}_{1|s}), \quad H_t \leftarrow \pi_{s\to t}(\widehat{\mathcal{X}}_{1|s})$ Eq. (20)
  end if
  $\widehat{Y}_t \leftarrow f_\theta(\mathcal{X}_t, c, t, H_t)$ Eq. (7)
  $\mathcal{L}(\theta) \leftarrow \mathcal{L}(\widehat{Y}_t, Y_t)$ Eq. (14)
  $\theta \leftarrow \mathrm{AdamW}(\theta, \nabla_\theta \mathcal{L}), \quad \bar{\theta} \leftarrow \mathrm{EMA}(\bar{\theta}, \theta)$
end while
return $f_{\bar{\theta}}$

Algorithm 2 Sampling Algorithm of AtomBloom

Input: $f_{\bar{\theta}}$, pocket c, grid $0 = t_0 < \cdots < t_L = 1$, state $\mathcal{X}_0$
Output: binder $\widehat{\mathcal{G}}$

$H_0 \leftarrow \varnothing$
for $k = 0, \ldots, L-1$ **do**
  $\widehat{Y}_k = (\widehat{\mathcal{X}}_{1|t_k}, \widehat{\ell}_k, \widehat{d}_k, \widehat{\gamma}_k, \widehat{\mathbf{q}}_k) \leftarrow f_{\bar{\theta}}(\mathcal{X}_{t_k}, c, t_k, H_{t_k})$ Eq. (7)
  $\rho_k \leftarrow 1 - S_{\mathrm{sp}}(t_{k+1})/S_{\mathrm{sp}}(t_k), \quad \eta_k \leftarrow 1 - S_{\mathrm{del}}(t_{k+1})/S_{\mathrm{del}}(t_k)$ Eqs. (16)–(17)
  $K_i \sim \mathrm{Poisson}(\rho_k e^{\widehat{\ell}_{k,i}}), \quad D_i \sim \mathrm{Bernoulli}(\eta_k \sigma(\widehat{d}_{k,i}))$ Eq. (17)
  $(\mathcal{X}^+_{t_k}, \pi_k) \leftarrow \mathrm{DeleteThenExpand}(\mathcal{X}_{t_k}, D, K)$
  $\mathcal{X}_{t_{k+1}} \leftarrow q_{t_k \to t_{k+1}}(\cdot \mid \mathcal{X}^+_{t_k}, \widehat{\mathcal{X}}_{1|t_k})$ Eq. (10)
  $\mathcal{X}_{t_{k+1}} \leftarrow \mathrm{RefineSplit}(\mathcal{X}_{t_k}, \mathcal{X}_{t_{k+1}}, c)$ Appendix D
  $\mathcal{X}_{t_{k+1}} \leftarrow \mathrm{ModalityUpdate}(\mathcal{X}_{t_{k+1}}, c, t_{k+1})$ Appendix E
  $H_{t_{k+1}} \leftarrow \pi_k(\mathrm{stopgrad}(\widehat{\mathcal{X}}_{1|t_k}))$ Section 3.6
end for
$\widehat{\mathcal{G}} \leftarrow \mathrm{Finalize}(\mathcal{X}_1, c)$ Appendix E
return $\widehat{\mathcal{G}}$

D POST-SPLIT ENDPOINT REFINEMENT

This section describes the endpoint-refinement step used in Algorithm 2. As the endpoint predictor is applied to $\mathcal{X}_s$ before splitting, each new child inherits its parent's endpoint $\overline{\mathbf{z}}_{1|s,j} = \widehat{\mathbf{z}}_{1|s,\pi_{s\to t}(j)}$. Siblings thus share the same endpoint and OU-bridge conditional mean under direct transport.

Transporting the children with these inherited endpoints yields a provisional time-t state.

$$\widetilde{\mathbf{z}}_{t,j} \sim \mathbb{P}_{\mathrm{OU}}\big(\mathbf{Z}_t \in \cdot \;\big|\; \mathbf{Z}_s = \mathbf{z}_{s,\pi_{s\to t}(j)}, \mathbf{Z}_1 = \overline{\mathbf{z}}_{1|s,j}\big)\,. \tag{47}$$

We then predict an endpoint for each child, $\widehat{\mathbf{z}}_{1|t,j} = f_\theta^{\mathrm{end}}(\widetilde{\mathcal{X}}_t, c, t)_j$, and resample the time-$t$ states from the same time-s parents.

$$\mathbf{z}_{t,j} \sim \mathbb{P}_{\mathrm{OU}}\big(\mathbf{Z}_t \in \cdot \;\big|\; \mathbf{Z}_s = \mathbf{z}_{s,\pi_{s\to t}(j)}, \mathbf{Z}_1 = \widehat{\mathbf{z}}_{1|t,j}\big)\,. \tag{48}$$

The second predictor call uses the coordinate scale $b(t)$ associated with the provisional state's time, and the recovered endpoints are used for transport in physical coordinates. This step retains the split and deletion decisions already sampled for the interval.

E MODALITY-SPECIFIC GENERATION AND DECODING

We first describe modality-specific updates during sampling, then the final conversion of generated atoms into a binder. At the terminal time, AtomBloom produces atom coordinates $\widehat{\boldsymbol{r}}_i \in \mathbb{R}^3$, atom-type probabilities $\widehat{p}_i \in \Delta^{C-1}$, and modality-specific auxiliary predictions. These quantities define a binder through the following decoding maps.

E.1 SMALL MOLECULES

Late geometric correction. Let $\widehat{\boldsymbol{r}}_i$ and $\boldsymbol{r}_j^{\mathrm{p}}$ denote a predicted ligand-atom position and a pocket-atom position, with van der Waals radii R_i and R_j^{p}. For tolerance τ, define their overlap by

$$o_{ij} = \Big[R_i + R_j^{\mathrm{p}} - \tau - \big\|\widehat{\boldsymbol{r}}_i - \boldsymbol{r}_j^{\mathrm{p}}\big\|_2\Big]_+ . \tag{49}$$

The correction direction for ligand atom i is

$$\mathbf{d}_i = \sum_j o_{ij} \frac{\widehat{\boldsymbol{r}}_i - \boldsymbol{r}_j^{\mathrm{p}}}{\big\|\widehat{\boldsymbol{r}}_i - \boldsymbol{r}_j^{\mathrm{p}}\big\|_2 + \epsilon}, \qquad \widehat{\boldsymbol{r}}_i \leftarrow \widehat{\boldsymbol{r}}_i + \alpha\ \mathrm{clip}_{d_{\max}}(\mathbf{d}_i), \tag{50}$$

where α is the correction strength and $\mathrm{clip}_d(\mathbf{v}) = \mathbf{v}\min\{1, d/(\|\mathbf{v}\|_2 + \epsilon)\}$ bounds the displacement norm. Applying Eq. (50) during the late sampling intervals reduces steric overlap without specifying molecular connectivity.

Molecular-graph decoding. The terminal element label is $\widehat{a}_i = \arg\max_a \widehat{p}_i(a)$. MolCraft (Qu et al., 2024) defines a graph-decoding map from the generated atoms to a molecular graph,

$$\widehat{G} = \mathcal{R}_{\mathrm{MC}}\left(\{(\widehat{\boldsymbol{r}}_i, \widehat{a}_i)\}_{i=1}^{N_1}\right), \tag{51}$$

where $\mathcal{R}_{\mathrm{MC}}$ determines the bond set and bond orders. The generated small-molecule binder is the resulting attributed geometric graph $\widehat{G}$.

E.2 PEPTIDES AND ANTIBODIES

Residue partition and type. For ordered atoms $1, \ldots, N_t$, let $\widehat{\gamma}_{t,i}$ be the predicted boundary logit between atoms i and $i+1$. At the commitment time t_c, the cut indicators

$$b_i = \mathbf{1}\{\widehat{\gamma}_{t_c,i} > 0\}, \qquad i = 1, \ldots, N_{t_c} - 1, \tag{52}$$

induce a partition $\mathcal{P} = \{g_1, \ldots, g_M\}$ into maximal contiguous groups. Let $\mathcal{P}_{\geq 2} = \{g \in \mathcal{P} : |g| \geq 2\}$ denote the non-singleton groups. For $g \in \mathcal{P}_{\geq 2}$, we pool the atom hidden representations $\mathbf{u}_{t_c,i}$ and predict its residue class as

$$\mathbf{u}_g = \frac{1}{|g|}\sum_{i\in g} \mathbf{u}_{t_c,i}, \qquad \widehat{z}_g = \arg\max_z \big[f_{\mathrm{grp}}(\mathbf{u}_g)\big]_z . \tag{53}$$

Residue-cardinality constraint. Let $m(z)$ be the canonical number of heavy atoms in residue class z, and let $n_g(s)$ be the cardinality of group g at time s. Over a constrained interval $[s,t]$, define the desired net cardinality change

$$\Delta_g(s,t) = \begin{cases} \mathrm{round}(\rho_{s,t}\,[m(\widehat{z}_g) - n_g(s)])\,, & n_g(s) < m(\widehat{z}_g), \\ -\,\mathrm{round}(\eta_{s,t}\,[n_g(s) - m(\widehat{z}_g)])\,, & n_g(s) > m(\widehat{z}_g), \\ 0, & n_g(s) = m(\widehat{z}_g), \end{cases} \tag{54}$$

where $\rho_{s,t}$ and $\eta_{s,t}$ are the exact split and deletion factors from Eqs. (16)–(17). The numbers of added and removed atoms, A_g and R_g, are chosen subject to

$$A_g - R_g = \Delta_g(s,t), \qquad A_g \geq 0, \qquad 0 \leq R_g < n_g(s). \tag{55}$$

Removal sites are selected according to the predicted deletion scores, while the A_g additions are sampled from the categorical distribution $w_{s,i} = \exp(\widehat{\ell}_{s,i}) / \sum_{j \in g} \exp(\widehat{\ell}_{s,j})$. Thus, the constraint preserves the committed partition and uses the learned event scores while moving each group toward the cardinality implied by its predicted residue class.

Canonical residue alignment. For a predicted residue class $\widehat{z}_g$, let $T_g = (V_g, E_g, \{a_u\}_{u \in V_g})$ be its canonical heavy-atom template. For a terminal group satisfying $|g| = |V_g|$, the correspondence between the template and generated atoms can be represented by a bijection $\pi_g : V_g \to g$. Define

$$\mathcal{L}_{\mathrm{align}}(\pi_g) = -\sum_{u \in V_g} \log \widehat{p}_{\pi_g(u)}(a_u) + \lambda_{\mathrm{geo}} \sum_{(u,v) \in E_g} \left\| \widehat{\boldsymbol{r}}_{\pi_g(u)} - \widehat{\boldsymbol{r}}_{\pi_g(v)} \right\|_2 . \tag{56}$$

The alignment solver returns $\widehat{\pi}_g \in \mathrm{Bij}(V_g, g)$ as an approximate minimizer of Eq. (56). The collection of aligned residue graphs defines the identities, coordinates, and covalent topology of the generated peptide or antibody binder. The AtomBloom-R variant uses the reference partition in place of Eq. (52), while retaining the same atom-level generation and decoding maps.

F Details of dynamic rescaling

This section gives the bridge coefficients and implementation details for dynamic rescaling in Section 3.5. All coordinates below are centered at the pocket center used for conditioning. The scaling schedule follows the coefficients of the position OU bridge in Eq. (10). For a bridge from time zero to time one, its conditional coordinate can be written as

$$\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \beta_t \mathbf{x}_1 + \gamma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I_3), \tag{57}$$

where

$$\alpha_t = \frac{\sinh(\kappa_{\mathrm{OU}}(1-t))}{\sinh(\kappa_{\mathrm{OU}})}, \qquad \beta_t = \frac{\sinh(\kappa_{\mathrm{OU}} t)}{\sinh(\kappa_{\mathrm{OU}})},$$
$$\gamma_t^2 = \frac{\sigma_{\mathrm{OU}}^2 \sinh(\kappa_{\mathrm{OU}} t)\sinh(\kappa_{\mathrm{OU}}(1-t))}{\kappa_{\mathrm{OU}} \sinh(\kappa_{\mathrm{OU}})}. \tag{58}$$

Given an empirical initial scale $b_0 > 0$, we define

$$b(t) = \sqrt{b_0^2 \alpha_t^2 + \beta_t^2 + \gamma_t^2}. \tag{59}$$

This gives $b(0) = b_0$ and $b(1) = 1$. We use $b_0 = 10$, $\kappa_{\mathrm{OU}} = 0.5$, and $\sigma_{\mathrm{OU}} = 1$.

Predictor coordinates. At each predictor call at time t, every input coordinate is transformed by

$$\mathcal{S}_t(\mathbf{x}) = \frac{\mathbf{x}}{b(t)}. \tag{60}$$

The same transformation is applied to current ligand atoms, pocket atoms, and self-conditioning virtual nodes. In particular, a virtual node carrying the earlier coordinate prediction $\widehat{\mathbf{x}}_{1|s,i}$ is placed at $\widehat{\mathbf{x}}_{1|s,i}/b(t)$ for a call at time t. Atom-type features and graph connectivity are retained.

Let $\widehat{\mathbf{y}}_{1|t,i}$ denote the terminal coordinate returned by the network in this scaled coordinate system. Its prediction in the original coordinate system is

$$\widehat{\mathbf{x}}_{1|t,i} = b(t)\widehat{\mathbf{y}}_{1|t,i}. \tag{61}$$

The recovered coordinates are used for OU-bridge transport and stored self-conditioning predictions. The predictor and endpoint-refinement calls each use the scale associated with their own evaluation time.

Coordinate supervision. For active forest node $v_i(t)$, the coordinate target supplied to the network is $\mathbf{x}(v_i(t))/b(t)$. In training, we replace the coordinate loss in Eq. (12) with its rescaled counterpart, computed in this coordinate system,

$$\mathcal{L}_{\text{pos}}^{\text{scaled}} = \mathbb{E}\left[\frac{1}{3}\left\|\widehat{\mathbf{y}}_{1|t,i} - \frac{\mathbf{x}(v_i(t))}{b(t)}\right\|_2^2\right] \tag{62}$$

The expectation averages over training examples, sampled times, and valid active atoms. Auxiliary geometric losses in Appendix G use the recovered coordinates from Eq. (61).

G Auxiliary Losses

The auxiliary objectives impose residue-level and geometric structure on the atom-level terminal prediction. Let $\mathbf{u}_{t,i}$ denote the hidden representation of active atom i, and let $\widehat{\boldsymbol{r}}_{1|t,i}$ be its predicted terminal coordinate. Every loss below is averaged over its valid supervised atoms, groups, or pairs. Unavailable targets and padding are excluded.

G.1 Residue supervision

Boundary prediction. For each valid adjacent pair $(i, i+1)$, let $b_i \in \{0,1\}$ indicate whether the two atoms belong to different residues, and let $\widehat{\gamma}_{t,i}$ be the predicted boundary logit. With $\mathcal{E}_{\text{adj}}$ denoting the set of supervised adjacent pairs, the boundary objective is

$$\mathcal{L}_{\text{boundary}} = \frac{1}{|\mathcal{E}_{\text{adj}}|}\sum_{i\in\mathcal{E}_{\text{adj}}} \text{BCEWithLogits}\left(\widehat{\gamma}_{t,i}, b_i\right). \tag{63}$$

Residue type. Let $\mathcal{G}$ be the set of supervised residue groups and z_g the residue class of group g. We first pool its atom representations,

$$\mathbf{u}_{t,g} = \frac{1}{|g|}\sum_{i\in g}\mathbf{u}_{t,i}, \qquad \widehat{\mathbf{q}}_{t,g} = f_{\text{grp}}(\mathbf{u}_{t,g}), \tag{64}$$

where f_{grp} produces residue-type logits. For C_g residue classes and label-smoothing parameter ε, define $\widetilde{y}_{g,z} = (1-\varepsilon)\mathbf{1}\{z = z_g\} + \varepsilon/C_g$. The corresponding classification loss is

$$\mathcal{L}_{\text{group}} = -\frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}}\sum_{z=1}^{C_g}\widetilde{y}_{g,z}\log\text{softmax}\left(\widehat{\mathbf{q}}_{t,g}\right)_z. \tag{65}$$

These two objectives are used for peptide and antibody generation. The boundary predictions determine the terminal partition in Eq. (52).

G.2 GEOMETRIC AND TOPOLOGICAL SUPERVISION

Let $\boldsymbol{r}_i$ denote the terminal target coordinate and define the smooth ℓ_1 penalty

$$H(a) = \begin{cases} \frac{1}{2}a^2, & |a| < 1, \\ |a| - \frac{1}{2}, & |a| \geq 1. \end{cases} \tag{66}$$

For a distance cutoff r_{c}, the supervised local-pair set is $\mathcal{E}_{\mathrm{loc}} = \{(i,j) : i < j,\ \|\boldsymbol{r}_i - \boldsymbol{r}_j\|_2 < r_{\mathrm{c}}\}$. The local-geometry loss preserves short-range distances within the binder.

$$\mathcal{L}_{\mathrm{local}} = \frac{1}{|\mathcal{E}_{\mathrm{loc}}|} \sum_{(i,j)\in\mathcal{E}_{\mathrm{loc}}} H\big(\|\widehat{\boldsymbol{r}}_{1|t,i} - \widehat{\boldsymbol{r}}_{1|t,j}\|_2 - \|\boldsymbol{r}_i - \boldsymbol{r}_j\|_2\big)\,. \tag{67}$$

Let $\boldsymbol{r}_k^{\mathrm{p}}$ be the fixed coordinate of pocket atom k and $\mathcal{E}_{\mathrm{pocket}} = \{(k,i) : \|\boldsymbol{r}_k^{\mathrm{p}} - \boldsymbol{r}_i\|_2 < r_{\mathrm{c}}\}$. Pocket-conditioned geometry is supervised by

$$\mathcal{L}_{\mathrm{pocket}} = \frac{1}{|\mathcal{E}_{\mathrm{pocket}}|} \sum_{(k,i)\in\mathcal{E}_{\mathrm{pocket}}} H\big(\|\boldsymbol{r}_k^{\mathrm{p}} - \widehat{\boldsymbol{r}}_{1|t,i}\|_2 - \|\boldsymbol{r}_k^{\mathrm{p}} - \boldsymbol{r}_i\|_2\big)\,. \tag{68}$$

Finally, let $y_{ij} \in \{0, 1, \ldots, B\}$ be the bond class of a supervised ligand-touching pair, where class 0 denotes no bond, and let $\widehat{\mathbf{e}}_{t,ij}$ be its predicted bond logits. For the supervised pair set $\mathcal{E}_{\mathrm{bond}}$, we use

$$\mathcal{L}_{\mathrm{bond}} = -\frac{1}{|\mathcal{E}_{\mathrm{bond}}|} \sum_{(i,j)\in\mathcal{E}_{\mathrm{bond}}} \log \mathrm{softmax}\left(\widehat{\mathbf{e}}_{t,ij}\right)_{y_{ij}}\,. \tag{69}$$

The geometric and bond objectives are activated only after the topology supervision horizon. Bond supervision is further restricted to samples for which the active leaves admit a valid correspondence with terminal ligand atoms.

Collecting the active terms in Eq. (14),

$$\begin{aligned} \sum_m w_m \mathcal{L}_{\mathrm{aux},m} = {} & w_{\mathrm{boundary}}\mathcal{L}_{\mathrm{boundary}} + w_{\mathrm{group}}\mathcal{L}_{\mathrm{group}} \\ & + w_{\mathrm{local}}\mathcal{L}_{\mathrm{local}} + w_{\mathrm{pocket}}\mathcal{L}_{\mathrm{pocket}} + w_{\mathrm{bond}}\mathcal{L}_{\mathrm{bond}}, \end{aligned} \tag{70}$$

with modality-inapplicable terms omitted.

H Implementation Details

We train one AtomBloom predictor for each modality and use its exponential-moving-average checkpoint for generation. The hyperparameters used in all reported experiments are provided in Table 5. For self-conditioning, we resample training times until $t \geq \delta_{\min}$, draw $\Delta \sim \mathrm{Uniform}(\delta_{\min}, \min(\delta_{\max}, t))$, and set $s = t - \Delta$. The bounds are listed in Table 5.

Table 5: Hyperparameters of AtomBloom.

Name	Value	Description
Architecture		
`n_layers`	6	Number of transformer layers.
`n_heads`	8	Number of attention heads.
`hidden_dim`	512	Dimension of hidden atom features.
`edge_dim`	32	Dimension of edge features.
`time_conditioning`	AdaLN-Zero	Conditioning mechanism for continuous time.
Optimization		
`optimizer`	AdamW	Optimizer used for all parameters.
`lr`	5×10^{-4}	Initial learning rate.
`lr_scheduler`	ReduceLROnPlateau	Learning-rate schedule.
`epochs`	400	Number of training epochs.
`precision`	bfloat16	Arithmetic precision.
`grad_clip`	$10.0 \to 1.5$	Gradient-norm threshold annealed during warm-up.
`ema_decay`	0.999	Exponential moving-average decay.
`ema_start`	50	Epoch at which moving-average updates begin.
Coordinate Scaling, Branching, and Self-Conditioning		
$b(t)$	Eq. (19)	Dynamic coordinate scale shared by ligand, pocket, and conditioning nodes.
b_0	10	Initial coordinate scale. The terminal scale is 1.
`split_time_dist`	$\mathrm{Beta}(1, 2)$	Continuous split-time scheduler.
`deletion_time_dist`	$\mathrm{Uniform}(0, 1)$	Continuous deletion-time scheduler.
`event_end_time`	0.9	Time after which split and deletion are disabled.
`sc_start`	50	Epoch at which model-rolled self-conditioning begins.
`sc_probability`	0.5	Probability of model-rolled training.
`sc_time_gap`	$[0.01, 0.1]$	Bounds $[\delta_{\min}, \delta_{\max}]$ on $t - s$.
`topology_teacher_forcing`	0.5	Probability of supplying reference auxiliary topology.
Generation		
`num_steps`	50	Number of finite sampling intervals.
`boundary_commit_time`	0.8	Boundary commitment time for peptides and antibodies.
`residue_constraint`	true	Enable residue-size constraints for peptides and antibodies.
`late_avoid_clash`	true	Enable late geometric guidance for small molecules.
`avoid_clash_after_t`	0.9	Guidance activation time for small molecules.
`avoid_clash_tolerance`	0.3 Å	Van der Waals overlap tolerance for small molecules.
`avoid_clash_max_step`	0.5 Å	Maximum displacement per guidance update.
`avoid_clash_strength`	1.5	Scale of the repulsive displacement.

Topology teacher forcing supplies reference bonds and group connectivity only to the auxiliary topology heads. It does not replace the generated atom state.

I Evaluation Protocols

I.1 Data and splits

We follow prior training data and benchmark splits for all three modalities (Kong et al., 2025). For peptides, we train on PepBench and ProtFrag (Kong et al., 2024), consisting of $4{,}157$ protein–peptide complexes and $70{,}498$ synthetic samples, and use 114 complexes for validation. We evaluate on the LNR benchmark of 93 protein–peptide complexes. For antibodies, we follow the established filtered SAbDab split (Kong et al., 2025), with $9{,}473$ entries for training and 400 for validation, and evaluate on 60 complexes from RAbD (Adolf-Bryfogle et al., 2018). For small molecules, we use the established CrossDocked2020 split (Peng et al., 2022), with $99{,}900$ complexes for training and 100 for testing, and sample 100 training complexes for validation.

I.2 De novo peptide design

Setup. We evaluate de novo peptide binder design on the LNR benchmark of 93 protein–peptide complexes, following UniMoMo (Kong et al., 2025). AAR measures amino-acid recovery. C-RMSD and L-RMSD are Cα RMSDs in Å after target-complex and ligand self-alignment, respectively. ΔG is the Rosetta interface energy (Alford et al., 2017), and IMP is the percentage of targets for which a generated candidate improves over the native binder. Clash and dihedral-distribution metrics follow the UniMoMo protocol. We evaluate at Sample@100.

Baselines. We compare with RFdiffusion (Watson et al., 2023), PepFlow (Li et al., 2024), PepGLAD (Kong et al., 2024), and single-/multi-domain variants of UniMoMo (Kong et al., 2025).

I.3 Antibody design

Setup. We evaluate CDR-H3 design on 60 antigen–antibody complexes from RAbD (Adolf-Bryfogle et al., 2018), following UniMoMo (Kong et al., 2025). We report AAR, RMSD, and IMP as recovery metrics under Sample@1, Sample@10, and Sample@100. AAR measures recovery of the native sequence, RMSD measures recovery of the native CDR structure, and IMP reports the percentage of complexes for which a generated CDR achieves better binding energy than the native CDR. We further evaluate structural plausibility using internal CDR and antigen–CDR clash rates together with backbone and side-chain dihedral-distribution JSDs.

Baselines. We compare with MEAN (Kong et al., 2023a), dyMEAN (Kong et al., 2023b), GeoAB-R and GeoAB-D (Lin et al., 2024b), DiffAb (Luo et al., 2022), and the single-domain and multi-domain variants of UniMoMo (Kong et al., 2025).

I.4 Small-molecule design

Setup. We evaluate de novo small-molecule design following the CBGBench protocol (Lin et al., 2024d). The benchmark evaluates substructure distributions, chemical properties, molecular geometry, and protein–ligand interactions. Following UniMoMo (Kong et al., 2025), the overall score is computed from the four category ranks with interaction receiving twice the weight of the other categories. QED and SA follow Bickerton et al. (2012) and Ertl & Schuffenhauer (2009), respectively.

Baselines. We compare with 14 previous methods spanning autoregressive (Peng et al., 2022), diffusion-based (Liu et al., 2022; Schneuing et al., 2024; Lin et al., 2025; Guan et al., 2023), fragment- or motif-based (Lin et al., 2024a), voxel-based (Pinheiro et al., 2024), Bayesian-flow (Qu et al., 2024), and latent generative approaches (Kong et al., 2025). Baseline results are taken from UniMoMo (Kong et al., 2025).

J RMSD FOR VARIABLE-LENGTH BINDERS

We measure structural similarity over the reference residue domain. After aligning the target proteins, let $R = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$ and $G = (\mathbf{g}_1, \ldots, \mathbf{g}_M)$ denote the reference and generated Cα coordinate traces in residue order. Residues with available Cα coordinates form each trace. For antibody design, the traces contain the designed CDR-H3 region.

Residue correspondence. Each reference residue is paired with a generated residue through a map $\phi : \{1, \ldots, N\} \to \{1, \ldots, M\}$. When $M \geq N$, the map is strictly increasing and selects an ordered subset of the generated trace. When $M < N$, it is nondecreasing and surjective, allowing generated residues to be reused while including every generated residue. The admissible maps are

$$\Phi_{N,M} = \begin{cases} \{\phi : \phi(j+1) > \phi(j),\ 1 \leq j < N\}, & M \geq N, \\ \{\phi : \phi(j+1) \geq \phi(j),\ 1 \leq j < N,\ \mathrm{Im}(\phi) = \{1, \ldots, M\}\}, & M < N. \end{cases} \tag{71}$$

We choose the correspondence in the target-aligned coordinate frame by minimizing the total squared distance,

$$\widehat{\phi} = \operatorname*{arg\,min}_{\phi \in \Phi_{N,M}} \sum_{j=1}^{N} \|\mathbf{g}_{\phi(j)} - \mathbf{r}_j\|_2^2. \tag{72}$$

This monotone matching is computed by dynamic programming in $O(NM)$ time. Every comparison contains exactly N residue pairs.

Complex and ligand RMSD. C-RMSD measures agreement in the target-aligned frame,

$$\text{C-RMSD}(G, R) = \sqrt{\frac{1}{N} \sum_{j=1}^{N} \|\mathbf{g}_{\widehat{\phi}(j)} - \mathbf{r}_j\|_2^2}. \tag{73}$$

L-RMSD uses the same residue pairs and additionally aligns the generated ligand to the reference,

$$\text{L-RMSD}(G, R) = \min_{Q \in \mathrm{SO}(3),\, \mathbf{a} \in \mathbb{R}^3} \sqrt{\frac{1}{N} \sum_{j=1}^{N} \|Q\mathbf{g}_{\widehat{\phi}(j)} + \mathbf{a} - \mathbf{r}_j\|_2^2}. \tag{74}$$

The rigid alignment is computed by centering the paired traces and applying the singular value decomposition with a proper-rotation constraint. The antibody RMSD reported in Table 2 uses the C-RMSD definition.

Equivalence at equal length. When $M = N$, the only strictly increasing map from $\{1, \ldots, N\}$ to itself is $\widehat{\phi}(j) = j$. The correspondence therefore reduces to the original residue-wise pairing, and Eqs. (73) and (74) become

$$\begin{aligned} \text{C-RMSD}(G, R) &= \sqrt{\frac{1}{N} \sum_{j=1}^{N} \|\mathbf{g}_j - \mathbf{r}_j\|_2^2}, \\ \text{L-RMSD}(G, R) &= \min_{Q \in \mathrm{SO}(3),\, \mathbf{a} \in \mathbb{R}^3} \sqrt{\frac{1}{N} \sum_{j=1}^{N} \|Q\mathbf{g}_j + \mathbf{a} - \mathbf{r}_j\|_2^2}. \end{aligned} \tag{75}$$

These are the conventional Cα RMSDs under target alignment and ligand self-alignment, respectively.

Geometric cost of shorter traces. When $M < N$, the surjective correspondence covers every reference position by reusing some generated positions. For a fixed correspondence, let $S_i = \{j : \widehat{\phi}(j) = i\}$ and let $\bar{\mathbf{r}}_i = |S_i|^{-1} \sum_{j \in S_i} \mathbf{r}_j$. The squared error decomposes as

$$\sum_{j \in S_i} \|\mathbf{g}_i - \mathbf{r}_j\|_2^2 = |S_i| \|\mathbf{g}_i - \bar{\mathbf{r}}_i\|_2^2 + \sum_{j \in S_i} \|\mathbf{r}_j - \bar{\mathbf{r}}_i\|_2^2. \tag{76}$$

Consequently,

$$\text{C-RMSD}(G,R)^2 \geq \frac{1}{N}\sum_{i=1}^{M}\sum_{j\in S_i}\|\mathbf{r}_j-\bar{\mathbf{r}}_i\|_2^2. \tag{77}$$

This bound quantifies the geometric cost of representing distinct reference positions with a shared generated position. Shortening cannot simply remove reference positions from the score. Appendix K separately examines their receptor contacts.

K Additional Analysis of Adaptive Peptide Length

We analyze the 100 candidates generated by AtomBloom for each of the 93 LNR targets in Table 1. These analyses examine how candidate length relates to energy-based selection and whether additional residues participate in receptor contacts. Length is measured by residue count, whereas structural correspondence uses the available Cα coordinates as defined in Appendix J.

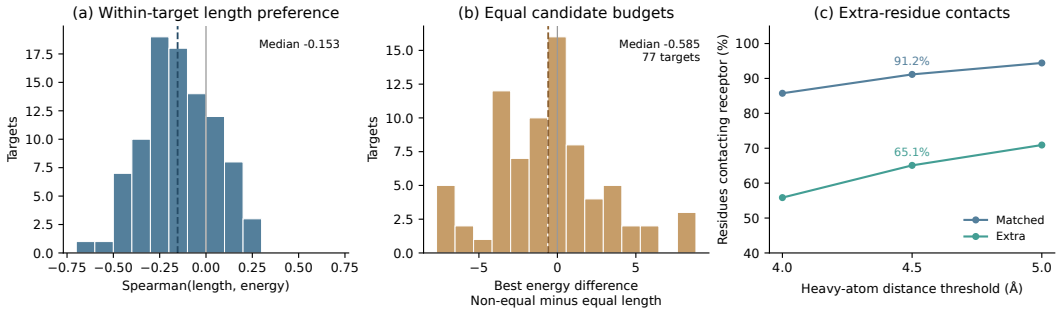


Figure 4: Adaptive-length diagnostics on the existing peptide samples. (a) Length–energy Spearman correlations computed within each target. (b) Differences in the minimum energy under equal candidate budgets for non-reference-length and reference-length subsets, averaged over 500 draws per target. Negative values favor non-reference lengths. (c) Median target-level receptor-contact fractions for residues included in the RMSD correspondence and for extra residues in longer candidates. Dashed lines in (a) and (b) mark the medians.

Energy-based selection and candidate length. Does selecting a low-energy peptide amount to selecting the longest candidate? Across 93 targets, the candidate with the lowest ΔG is also the longest for only six targets (6.45%). This count is unchanged when any tied lowest-energy candidate is considered. The median within-target Spearman correlation between residue count and ΔG is -0.153 (Figure 4a), and 52 of 93 targets have $|\rho| < 0.2$. The selected candidate has a median length percentile of 72%, using midranks for equal lengths. Thus, energy-based selection exhibits a mild preference for longer candidates but is not equivalent to selecting the longest peptide. Favorable candidates emerge from combinations of length, sequence, and geometry within the expanded design space.

Equal-budget comparison within AtomBloom. To separate the value of non-reference lengths from differences in candidate counts, we partition each target's candidates into reference-length and non-reference-length subsets. We draw $k = \min(n_{\text{equal}}, n_{\text{non-equal}}, 10)$ candidates without replacement from each subset and compare their minimum energies. We retain the 77 targets with $k \geq 3$ and repeat the draw 500 times per target. After averaging the non-reference-minus-reference energy difference within each target, the median across targets is -0.585 (Figure 4b). Under matched sampling budgets, non-reference-length candidates achieve lower minimum energies on average for 43 of the 77 targets. This comparison supports the additional search value of variable-length candidates under matched candidate budgets. It compares subsets of the same model's outputs, with the non-reference-length subset containing both shorter and longer peptides.

Receptor contacts of extra residues. For candidates longer than the reference, we identify the generated residues used by the order-preserving RMSD correspondence and call the remaining

residues *extra*. A residue contacts the receptor if any ligand–receptor heavy-atom pair lies within the specified distance threshold. We compute contact fractions separately for matched and extra residues, pooling residues within each target and then taking the median across targets. At 4.5 Å, the fractions are 91.2% for matched residues and 65.1% for extra residues. The extra-residue fraction ranges from 55.9% to 70.9% as the threshold varies from 4 to 5 Å (Figure 4c). These results show that added residues frequently participate in receptor contacts, providing a route to extending the binding footprint. The contact analysis uses generated structures before Rosetta relaxation.

Examples across peptide lengths. Figure 5 illustrates that AtomBloom can generate peptides both longer and shorter than the reference binder, as well as designs with the same residue count. These twelve examples demonstrate the flexibility to explore peptide lengths beyond the reference count.

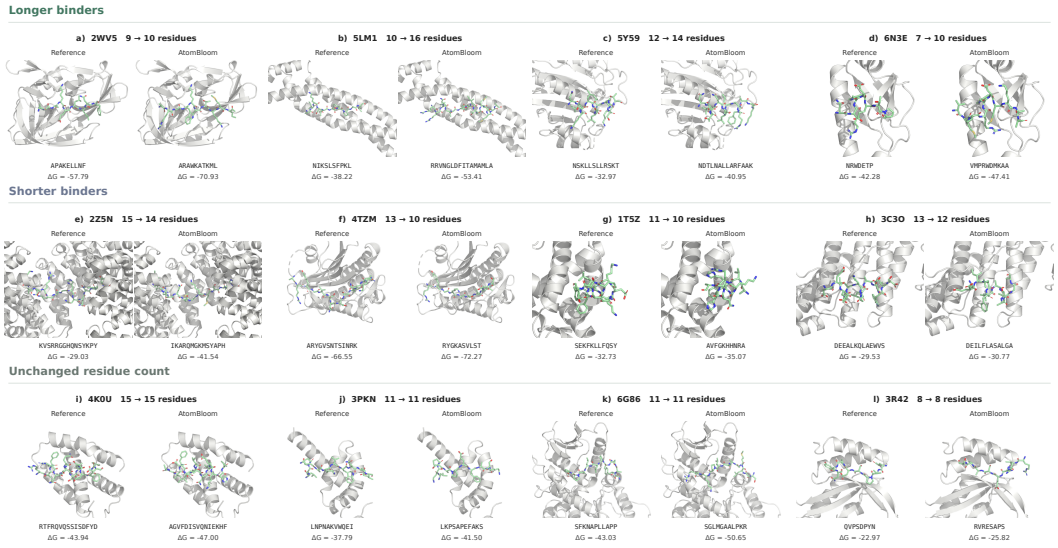


Figure 5: Flexibility of AtomBloom to generate peptides longer or shorter than the reference binder. The three rows show increased (top), decreased (middle), and unchanged (bottom) residue counts, with four targets per row. Each pair shows the reference and an AtomBloom candidate in the same target coordinate frame and at the same scale, with light-gray receptor cartoons and green peptide carbon atoms. Sequences, residue counts, and predicted interface energies are shown. Cases were selected from existing samples for sequence composition, structural quality, visual clarity, and favorable predicted interface energies relative to the reference.

L Additional Self-Conditioning Results

L.1 Inference-time self-conditioning

Table 6 reports the complete peptide results underlying Figure 2(e). For both AtomBloom-R and AtomBloom, self-conditioning is disabled only at inference time while retaining the same trained checkpoint and evaluation protocol.

Table 6: Inference-time self-conditioning analysis for peptide binder design on 93 LNR targets at Sample@100. The metrics follow Table 1.

	AtomBloom-R		AtomBloom	
Metric	w/ SC	w/o SC	w/ SC	w/o SC
AAR (%) ↑	42.88	44.25	40.55	38.04
C-RMSD (Å) ↓	1.92	1.41	1.18	1.41
L-RMSD (Å) ↓	0.76	0.62	0.69	0.79
ΔG ↓	-43.94	-41.12	-45.38	-42.12
IMP (%) ↑	70.97	60.22	79.57	66.67
Clash_{in} ↓	0.08%	0.38%	0.13%	0.63%
$\text{Clash}_{\text{out}}$ ↓	0.44%	0.93%	0.38%	0.95%
JSD_{bb} ↓	0.109	0.116	0.102	0.109
JSD_{sc} ↓	0.156	0.138	0.155	0.129
Seq. Div. ↑	0.189	0.440	0.346	0.593
Struct. Div. ↑	0.103	0.399	0.158	0.404

L.2 Training with model-rolled states

Model-rolled training exposes the predictor to continuous states influenced by its own earlier predictions. To examine this choice, we compare it with *oracle-state training*: the predictor receives the original state $\mathcal{X}_t$ sampled from the training bridge, together with the same detached ancestor predictions as self-conditioning context. Both variants retain the sampled genealogy, lineage-based context assignment, and original supervision targets. The comparison therefore changes how the current training state is constructed while retaining self-conditioning in both training and inference.

Setup. We retrain the small-molecule model and both the fixed-residue-count (AtomBloom-R) and adaptive-residue-count (AtomBloom) peptide and antibody models with model rolling disabled. Each training configuration retains its original seed, data, losses, optimizer, batch size, and 400-epoch schedule. We evaluate the final EMA checkpoints with 50 sampling steps and 100 candidates per target. Both variants use self-conditioning and endpoint refinement during inference.

Small molecules. Table 7 shows that model-rolled training improves Vina energies and improvement rates, QED, synthetic accessibility, and substructure distributions. Thus, the clearest gains from model rolling are in docking scores, chemical properties, and substructure quality.

Peptides and antibodies. Table 8 compares binding scores, local geometry, sequence recovery, and sequence diversity. Model-rolled training improves peptide interface energies and backbone geometry and reduces clashes in both variants. For antibodies, it improves IMP and local geometry in both variants; the mean best interface energy favors oracle-state training for AtomBloom-R and model-rolled training for AtomBloom. Oracle-state training increases sequence diversity in all four models.

Table 7: Model-rolled versus oracle-state training for small-molecule design. Both variants generate 100 candidates for each of the 100 test targets.

Metric	Model-rolled	Oracle-state
Atom-type JSD ↓	0.0379	0.0450
Ring-type JSD ↓	0.1047	0.1665
Functional-group JSD ↓	0.1275	0.1567
QED ↑	0.530	0.496
SA ↑	0.727	0.672
Vina Score E ↓	-6.350	-5.566
Vina Score IMP (%) ↑	43.66	26.94
Vina Dock E ↓	-7.479	-7.158
Vina Dock IMP (%) ↑	58.44	47.49
$\mathrm{JSD}_{\mathrm{BL}}$ ↓	0.2572	0.2572
$\mathrm{JSD}_{\mathrm{BA}}$ ↓	0.3740	0.3715

Table 8: Training-state comparison on 93 peptide and 60 antibody targets at Sample@100. Both variants retain inference-time self-conditioning. Clash rates are reported in percent.

	AtomBloom-R		AtomBloom	
Metric	Model-rolled	Oracle-state	Model-rolled	Oracle-state
Peptides				
AAR (%) ↑	42.88	44.24	40.55	38.46
ΔG ↓	-43.94	-41.98	-45.38	-45.16
$\mathrm{Clash}_{\mathrm{in}}$ (%) ↓	0.079	0.182	0.127	0.369
$\mathrm{Clash}_{\mathrm{out}}$ (%) ↓	0.435	0.735	0.384	0.910
$\mathrm{JSD}_{\mathrm{bb}}$ ↓	0.1087	0.1251	0.1017	0.1219
$\mathrm{JSD}_{\mathrm{sc}}$ ↓	0.1556	0.1343	0.1547	0.1470
Sequence diversity ↑	0.189	0.332	0.346	0.683
Antibodies				
AAR (%) ↑	54.29	53.29	48.15	47.55
IMP (%) ↑	71.67	68.33	78.33	66.67
$\mathrm{Clash}_{\mathrm{in}}$ (%) ↓	0.103	0.125	0.130	0.210
$\mathrm{Clash}_{\mathrm{out}}$ (%) ↓	0.026	0.043	0.039	0.090
$\mathrm{JSD}_{\mathrm{bb}}$ ↓	0.1824	0.1954	0.1799	0.2017
$\mathrm{JSD}_{\mathrm{sc}}$ ↓	0.2311	0.2847	0.2352	0.3045

M DYNAMIC RESCALING RESULTS

Table 9 gives the raw values plotted in Figure 2(c). We compare the dynamic schedule in Eq. (19) with $b(t) \equiv 1$ and $b(t) \equiv 10$, training each variant separately and evaluating the final epoch-400 EMA checkpoint. Each uses 50 sampling steps and 100 candidates per target, generated with 20 seeds and five candidates per seed. Exact event probabilities, endpoint refinement, and self-conditioning are retained across variants.

The metrics follow the modality-specific protocols in Appendix I. $\mathrm{JSD}_{\mathrm{BL}}$ measures small-molecule bond-length distributions, whereas $\mathrm{JSD}_{\mathrm{bb}}$ measures backbone dihedral distributions.

Table 9: Raw results for the dynamic rescaling ablation in Figure 2(c). IMP is reported in percent. Bold marks the best value in each row.

Modality	Metric	Dynamic	Fixed scale 1	Fixed scale 10
Small molecule	$\mathrm{JSD}_{\mathrm{BL}} \downarrow$	0.2572	**0.2563**	0.3016
	Vina Score $E \downarrow$	**-6.350**	-6.205	-5.704
	Vina Score IMP $\uparrow$	**43.66**	41.37	29.04
	Vina Dock $E \downarrow$	**-7.479**	-7.236	-7.257
	Vina Dock IMP $\uparrow$	**58.44**	50.39	51.91
Peptide	$\mathrm{JSD}_{\mathrm{bb}} \downarrow$	**0.1017**	0.1100	0.1063
	$\Delta G \downarrow$	**-45.38**	-43.66	-41.51
	IMP $\uparrow$	**79.57**	72.04	65.59
Antibody	$\mathrm{JSD}_{\mathrm{bb}} \downarrow$	**0.1799**	0.2038	0.1945
	IMP $\uparrow$	**78.33**	73.33	66.67

For the common display scale in Figure 2(c), a metric value m is shown as $100[1 + s(m - m_{\mathrm{dyn}})/|m_{\mathrm{dyn}}|]$, where $s = 1$ for higher-is-better metrics and $s = -1$ for lower-is-better metrics. This sets the dynamic baseline to 100 and gives all plotted scores the same direction.

N COMPLETE SMALL-MOLECULE RESULTS

Table 10 reports all evaluated metrics for the same methods shown in Table 4.

Table 10: Small-molecule quality across substructure, chemical properties, molecular geometry and interaction. Vina energies are in kcal/mol (Trott & Olson, 2010), and PLIP metrics follow Salentin et al. (2015). Bold and underline denote the best and second-best results.

Model	Substructure						Chemical Properties				Molecular Geometry			
	Atom Type		Ring Type		Functional Group		QED ↑	LogP	SA ↑	LPSK ↑	JSD_{BL} ↓	JSD_{BA} ↓	$Ratio_{cca}$ ↓	$Ratio_{cm}$ ↓
	JSD ↓	MAE ↓	JSD ↓	MAE ↓	JSD ↓	MAE ↓								
LIGAN	0.1167	0.8680	0.3163	0.2701	0.2468	0.0378	0.46	0.56	0.66	4.39	0.4645	0.5673	0.0096	0.0718
3DSBDD	0.0860	0.8444	0.3188	0.2457	0.2682	0.0494	0.48	0.47	0.63	4.72	0.5024	0.3904	0.2482	0.8683
GraphBP	0.1642	1.2266	0.5061	0.4382	0.6259	0.0705	0.44	3.29	0.64	4.73	0.5182	0.5645	0.8634	0.9974
Pocket2Mol	0.0916	1.0497	0.3550	0.3545	0.2961	0.0622	0.39	2.39	0.65	4.58	0.5433	0.4922	0.0576	0.4499
TargetDiff	0.0533	0.2399	0.2345	0.1559	0.2876	0.0441	0.49	1.13	0.60	4.57	0.2659	0.3769	0.0483	0.4920
DiffSBDD	0.0529	0.6316	0.3853	0.3437	0.5520	0.0710	0.49	-0.15	0.34	4.89	0.3501	0.4588	0.1083	0.6578
DiffBP	0.2591	1.5491	0.4531	0.4068	0.5346	0.0670	0.47	5.27	0.59	4.47	0.3453	0.4621	0.0449	0.4077
FLAG	0.1032	1.7665	0.2432	0.3370	0.3634	0.0666	0.41	0.29	0.58	**4.93**	0.4215	0.4304	0.6777	0.9769
D3FG	0.0644	0.8154	0.1869	0.2204	0.2511	0.0516	0.49	1.56	0.66	4.84	0.3727	0.4700	0.2115	0.8571
DecompDiff	0.0431	0.3197	0.2431	0.2006	0.1916	0.0318	0.49	1.22	0.66	4.40	0.2576	0.3473	0.0462	0.5248
MolCRAFT	0.0490	0.3208	0.2469	**0.0264**	0.1196	0.0477	0.48	0.87	0.66	4.39	**0.2250**	**0.2683**	0.0264	0.2691
VoxBind	0.0942	0.3564	0.2401	0.0301	**0.1053**	0.0761	0.54	2.22	0.65	4.70	0.2701	0.3771	0.0103	0.1890
UniMoMo (single)	0.0352	0.2796	0.1252	0.0976	0.1562	0.0173	0.53	1.36	0.68	4.69	0.3362	0.4166	0.0042	0.0715
UniMoMo (all)	**0.0280**	**0.2108**	0.1073	0.0726	0.1356	0.0161	**0.55**	1.55	0.70	4.68	0.3223	0.3848	**0.0040**	**0.0708**
AtomBloom	0.0379	0.2633	**0.1047**	0.0413	0.1275	**0.0114**	0.53	1.38	**0.73**	4.46	0.2572	0.3740	0.0045	0.0717

Model	Vina Score		Vina Min		Vina Dock				PLIP OA		PLIP PP	
	E ↓	IMP (%) ↑	E ↓	IMP (%) ↑	E ↓	IMP (%) ↑	MPBG (%) ↑	LBE ↑	JSD ↓	MAE ↓	JSD ↓	MAE ↓
LIGAN	**-6.47**	**62.13**	**-7.14**	**70.18**	-7.70	**72.71**	4.22	0.3897	0.0346	0.0905	0.1451	**0.3416**
3DSBDD	–	3.99	-3.75	17.98	-6.45	31.46	9.18	0.3839	0.0392	0.0934	0.1733	0.4231
GraphBP	–	0.00	–	1.67	-4.57	10.86	-30.03	0.3200	0.0462	0.1625	0.2101	0.4835
Pocket2Mol	-5.23	31.06	-6.03	38.04	-7.05	48.07	-0.17	**0.4115**	0.0319	0.2455	0.1535	0.4152
TargetDiff	-5.71	38.21	-6.43	47.09	-7.41	51.99	5.38	0.3537	0.0198	0.0600	0.1757	0.4687
DiffSBDD	–	12.67	-2.15	22.24	-5.53	29.76	-23.51	0.2920	0.0333	0.1461	0.1777	0.5265
DiffBP	–	8.60	–	19.68	-7.34	49.24	6.23	0.3481	0.0249	0.1430	**0.1256**	0.5639
FLAG	–	0.04	–	3.44	-3.65	11.78	-47.64	0.3319	0.0170	0.0277	0.2762	0.3976
D3FG	–	3.70	-2.59	11.13	-6.78	28.9	-8.85	0.4009	0.0638	**0.0135**	0.1850	0.4641
DecompDiff	-5.18	19.66	-6.04	34.84	-7.10	48.31	-1.59	0.3460	0.0215	0.0769	0.1848	0.4369
MolCRAFT	-6.15	54.25	-6.99	56.43	**-7.79**	56.22	8.38	0.3638	0.0214	0.0780	0.1868	0.4574
VoxBind	-6.16	41.80	-6.82	50.02	-7.68	52.91	**9.89**	0.3588	0.0257	0.0533	0.1850	0.4606
UniMoMo (single)	-5.50	26.37	-5.96	38.21	-7.19	49.60	2.15	0.3521	0.0175	0.0447	0.2372	0.4706
UniMoMo (all)	-5.72	30.40	-6.08	39.23	-7.25	51.59	7.50	0.3473	**0.0135**	0.0173	0.2012	0.4277
AtomBloom	-6.35	43.66	-6.64	49.97	-7.48	58.44	5.49	0.3658	0.0216	0.0330	0.1860	0.4254

O LIMITATIONS AND FUTURE WORK

AtomBloom learns pocket-conditioned generation from structural data. Reward-based fine-tuning is outside the scope of this work. Extending reinforcement learning to Branching Flows is a promising direction for optimizing task-specific molecular properties. Such an extension calls for algorithms that coordinate updates to continuous atom states with discrete splitting and deletion decisions as molecular cardinality changes.