EUTERPE: EFFICIENT MULTI-LEVEL PRESERVATION OF PRETRAINED PERFORMANCE DURING PREFERENCE ALIGNMENT FOR PROTEIN INVERSE FOLDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Protein inverse folding (IF) aims to design amino acid sequences for a given protein backbone, making it a fundamental tool for protein design. Recent Direct Preference Optimization (DPO)-based IF post-training improves developability[1] properties such as solubility and thermostability, but can drive the aligned model away from the pretrained IF model. Existing multi-objective methods address this trade-off by incorporating costly folding-derived structural signals as additional designability[2] alignment objectives. Our analysis shows, however, that the resulting structural gains largely recover the performance lost during developability alignment, rather than consistently improving beyond the pretrained IF model. This raises a natural question: **can we regulate developability alignment using the pretrained IF model itself, without costly folding-derived feedback?** We introduce **Euterpe**, an efficient semi-online preference optimization framework that uses the pretrained IF model as a reference prior to regulate alignment. Pair-level Reweighting uses reference-model energies to downweight preference pairs weakly supported by the pretrained distribution, while Token-Level Thinking-Pattern-Guided Reweighting modulates residue-wise updates according to the *thinking pattern* induced by the pretrained IF model. Built on ProteinMPNN, **Euterpe** preserves designability and AAR at levels comparable to methods using explicit folding-derived structural feedback, while eliminating such feedback during alignment, requiring fewer GFLOPs, and achieving up to a 6.9× speedup.

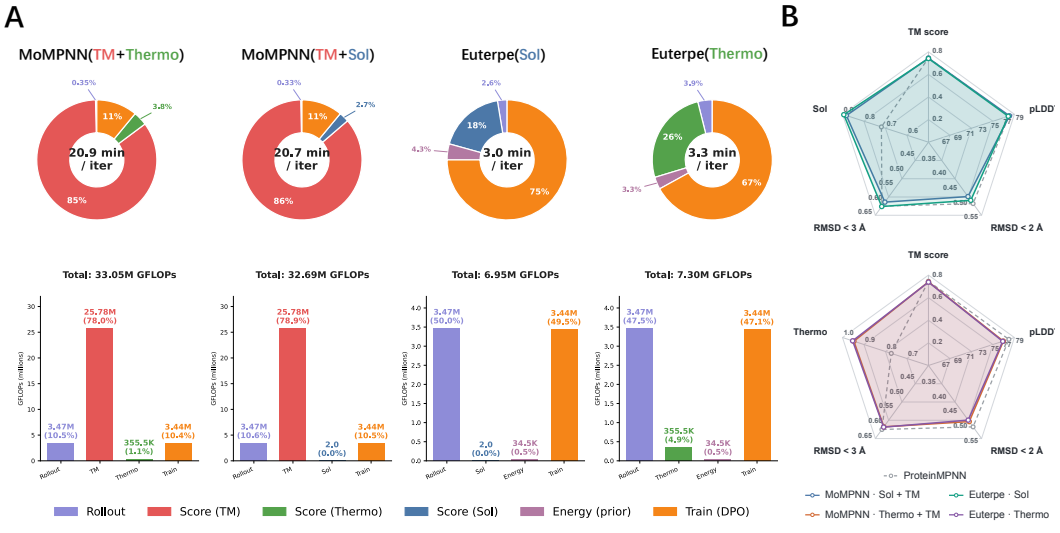


Figure 1: Efficiency and performance comparison. **(A)** Per-epoch runtime and computational cost (GFLOPs) of different post-training stages on 8 NVIDIA A100 40GB GPUs. **Euterpe** requires fewer GFLOPs while achieving up to a 6.9× speedup. **(B) Euterpe** achieves comparable designability without relying on folding-based scalar structural feedback.

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.

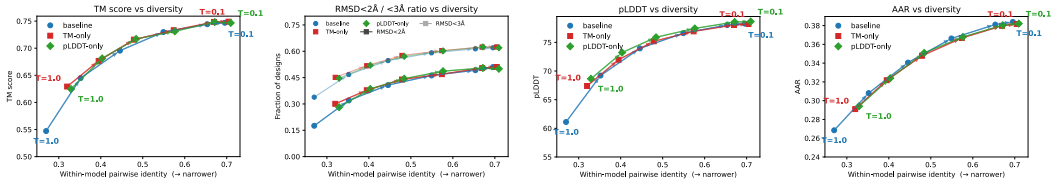


Figure 2: Comparison of the pretrained model and designability-only DPO models optimized with TM-score or pLDDT across sampling temperatures ($1.0 \rightarrow 0.8 \rightarrow 0.6 \rightarrow 0.4 \rightarrow 0.2 \rightarrow 0.1$). The gains are more pronounced at high temperatures but substantially diminish as the temperature decreases, while the aligned models follow trends similar to the pretrained model.

1 INTRODUCTION

Protein inverse folding (IF) aims to design amino acid sequences that are compatible with a prescribed protein backbone, serving as a fundamental task in computational protein design. Recent advances in deep learning have substantially improved IF by learning structure-to-sequence relationships from large-scale structural data (Jing et al., 2020; Hsu et al., 2022; Gao et al., 2022; 2024; Zhu et al., 2026). These methods have achieved strong sequence recovery and generalization across diverse protein backbones, providing a powerful foundation for downstream protein design. However, standard IF training primarily optimizes sequence likelihood under structural conditioning, rather than properties important for practical protein design (Widatalla et al., 2024; Sumida et al., 2024; Goverde et al., 2024). Although some biophysical preferences may be implicitly captured from the training distribution, they are not necessarily aligned with specific downstream fitness objectives, motivating post-training with property-specific supervision or preference signals.

Preference-based post-training has recently emerged as a promising approach for improving pretrained IF models beyond their original training objectives (Widatalla et al., 2024; Xu et al., 2026; Xue et al., 2025; Wang et al., 2025; Zeng et al., 2026). A particularly important setting is to improve developability[1] while retaining the performance of the pretrained IF model, as developability objectives such as solubility and thermostability do not directly enforce compatibility with the conditioning backbone. MoMPNN (Liu et al., 2026) considers this setting through multi-objective preference alignment, jointly optimizing developability preferences such as solubility and thermostability with designability[2] preferences constructed from folding-derived scalar scores.

We first observe that developability-only alignment can substantially degrade both designability and amino-acid recovery (AAR) (Fig. 5(B)), whereas incorporating folding-derived designability feedback largely restores these metrics toward their pretrained levels (see MoMPNN vs. ProteinMPNN in Tab. 1). As a complementary motivating analysis, we further perform designability-only DPO using representative folding-derived signals, including TM-score and pLDDT, to examine how much such feedback can improve ProteinMPNN in isolation. Although the gains are more pronounced at high sampling temperatures, they substantially diminish as the temperature decreases, while the aligned models exhibit sampling-dependent trends in both designability and AAR that remain close to those of the pretrained model (Fig. 2).

These observations motivate a natural question: **can we use the pretrained IF model itself to regulate developability alignment without costly folding-derived structural feedback?** To address this question, we introduce **Euterpe** (**E**fficient M**u**lti-Level Preserva**t**ion of Pretrained P**e**rformance during P**r**eference Alignment for **P**rotein Inv**e**rse Folding), an efficient semi-online preference optimization framework that uses the pretrained IF model as a reference prior to regulate preference updates. Rather than introducing explicit folding-derived designability feedback, it exploits information already encoded in the pretrained IF model at both the sequence and token levels. At the sequence level, **Euterpe** derives group-standardized reference energies from pretrained-model likelihoods to reweight preference pairs according to their support under the pretrained IF distribution. At the token level, it reweights residue-wise preference components according to the local *thinking*

[1] **Developability** refers to sequence properties relevant to practical protein quality beyond structural recovery, including solubility, thermostability, and evolutionary plausibility in this work.

[2] **Designability** refers to the ability of a designed sequence to fold into the intended target structure.

pattern induced by the pretrained IF model, assigning different relative contributions based on local reference support. Together, these mechanisms regulate preference-induced departure from the pretrained IF model while retaining effective alignment toward target developability properties.

The main contributions of our work are as follows:

- **We investigate the role of folding-derived structural feedback in preference alignment of ProteinMPNN.** Our experiments show that developability-only alignment can degrade pretrained IF performance, while folding-derived feedback largely restores it. Designability-only DPO further shows that gains beyond pretraining are strongly sampling-dependent and diminish at low sampling temperatures. Together, our results demonstrate that such feedback primarily serves to preserve pretrained IF performance during developability alignment, rather than to consistently improve beyond the pretrained model.
- **We propose Euterpe, an efficient semi-online preference optimization framework without explicit folding-derived structural feedback.** Euterpe uses the pretrained IF model as a structure-conditioned reference prior and regulates property alignment through pair-level reference-energy reweighting and token-level thinking-pattern-guided reweighting. It further supports multiple developability objectives through adaptive cross-objective preference optimization.
- **We demonstrate comparable preservation at substantially lower post-training cost.** Built on ProteinMPNN, **Euterpe** closely matches the preservation of designability and AAR achieved by methods using explicit folding-derived structural feedback, while requiring fewer GFLOPs and achieving up to a $6.9\times$ speedup.

2 Related Works

Protein Inverse Folding Pre-training. Protein inverse folding aims to design amino acid sequences that are compatible with a prescribed protein backbone. Traditional approaches rely on structural templates and physics-inspired priors, whereas modern learning-based methods directly learn transferable structure-to-sequence relationships from structural data. Early neural approaches employ geometric graph representations (Ingraham et al., 2019; Jing et al., 2020), followed by increasingly expressive models (Dauparas et al., 2022; Hsu et al., 2022; Gao et al., 2022; 2024). More recent studies further explore alternative generative paradigms (Nisonoff et al., 2025), the integration of protein language models (Zheng et al., 2023; Ren et al., 2024; Qiu et al., 2024), and the incorporation of complementary structural priors through retrieval and adaptive fusion (Zhu et al., 2026). These advances have substantially improved native sequence recovery and generalization across diverse protein backbones. Among them, ProteinMPNN remains one of the most widely adopted and experimentally validated IF models, serving as a common backbone for downstream protein design and post-training studies. Consistent with closely related post-training work (Liu et al., 2026), we adopt ProteinMPNN as our base model to ensure direct and controlled comparison under the same pretrained backbone.

Protein Inverse Folding Post-training. Post-training (Schulman et al., 2017; Christiano et al., 2017; Ziegler et al., 2019; Ouyang et al., 2022; Shao et al., 2024) has been increasingly explored in protein design to adapt pretrained models toward desired properties through supervised fine-tuning, preference optimization, reinforcement learning, among others (Mistani & Mysore, 2024; Stocco et al., 2024; Zhang et al., 2026; Blalock et al.). Within protein inverse folding, it has been introduced to incorporate property-specific or structural feedback into backbone-conditioned sequence generation. Property-specific retraining methods (Goverde et al., 2024; Ertelt et al., 2024) adapt ProteinMPNN toward solubility and thermostability, respectively. Preference-based approaches further introduce experimental or folding-derived feedback: ProteinDPO (Widatalla et al., 2024) leverages experimentally derived stability preferences, InstructPLM-DPO (Xu et al., 2026) constructs pair-level preferences using TM-score, and ResiDPO (Xue et al., 2025) introduces residue-level structural supervision for designability alignment. ProteinZero (Wang et al., 2025) instead adopts online GRPO (Shao et al., 2024) with composite structural rewards. SSP (Zeng et al., 2026) decouples structural objectives into separate policies that interact through a shared sampling pool. More recently, MoMPNN (Liu et al., 2026) considers multi-objective preference alignment for pretrained IF models, jointly optimizing developability and designability-related objectives within a semi-online DPO framework. Our work focuses on understanding the role of the folding-derived designability

signals used in this setting and whether their preservation function can instead be provided by the pretrained IF model itself.

3 PRELIMINARIES

ProteinMPNN. ProteinMPNN (Dauparas et al., 2022) is an order-agnostic autoregressive model for protein inverse folding. Given a protein backbone x and a sequence $y = (y_1, \ldots, y_L)$ of length L, it generates residues according to a randomly sampled decoding order o. Conditioned on a specific decoding order, the sequence likelihood is factorized as

$$\pi_\theta(y \mid x, o) = \prod_{i=1}^{L} \pi_\theta\left(y_{\sigma(i)} \mid x, y_{<o^{-1}(i)}\right), \tag{1}$$

where $o^{-1}(i)$ denotes the decoding step of residue i, and $y_{<o^{-1}(i)} = \{y_{o(j)} \mid j < o^{-1}(i)\}$ denotes the residues generated before y_i under the decoding order o.

Following the energy-based interpretation of probabilistic models (LeCun et al., 2006; Song & Kingma, 2021), we define the length-normalized sequence energy as

$$E_\theta(y \mid x, o) = -\frac{1}{L} \sum_{i=1}^{L} \log \pi_\theta\left(y_i \mid x, y_{<o^{-1}(i)}\right), \tag{2}$$

where a lower energy indicates a higher sequence likelihood under the given backbone.

Direct Preference Optimization. Direct Preference Optimization (DPO) (Rafailov et al., 2023) learns from pairwise preferences without explicitly training a reward model. Given a conditioning input x, after constructing winner-loser pairs (y_w, y_l), DPO optimizes a policy π_θ as follow:

$$\mathcal{L}_{\mathrm{DPO}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\left[\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\mathrm{ref}}(y_w \mid x)} - \log\frac{\pi_\theta(y_l \mid x)}{\pi_{\mathrm{ref}}(y_l \mid x)}\right]\right)\right], \tag{3}$$

where $\sigma(\cdot)$ denotes the sigmoid function and β controls the strength of preference optimization.

Multi-Objective Direct Preference Optimization. Multi-Objective DPO (MoDPO) (Liu et al., 2026) extends DPO to optimize multiple potentially conflicting objectives. For k-th objective, a preference dataset $\mathcal{D}_k$ is constructed according to its corresponding reward r_k. The overall objective is formulated as

$$\begin{aligned}\mathcal{L}_{\mathrm{MoDPO}}(\theta) = -\sum_{k=1}^{K} w_k \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_k}\Bigg[\log\sigma\Bigg(\frac{1}{w_k}\Big[\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\mathrm{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\mathrm{ref}}(y_l \mid x)} \\ - \sum_{k'\neq k} w_{k'}\left(r_{k'}(x, y_w) - r_{k'}(x, y_l)\right)\Big]\Bigg)\Bigg].\end{aligned} \tag{4}$$

Here, w_k controls the importance of the k-th objective. Compared with standard DPO, MoDPO incorporates the reward differences from auxiliary objectives into the preference margin. Therefore, the optimization strength of each preference pair is adjusted according to its performance on other objectives, mitigating conflicts among multiple preferences.

4 METHODS

4.1 PROBLEM FORMULATION AND SEMI-ONLINE PREFERENCE CONSTRUCTION

Given a protein backbone x, protein inverse folding aims to model the conditional distribution over amino-acid sequences $y = (y_1, \ldots, y_L)$ that are compatible with x. We denote the current policy as $\pi_\theta(y \mid x)$ and the frozen pretrained IF model as $\pi_{\mathrm{ref}}(y \mid x)$. For each backbone x, candidate sequences are sampled from the current policy, $y \sim \pi_\theta(\cdot \mid x)$. For each sampled sequence, we record the decoding order used during generation and reuse it throughout optimization. Specifically,

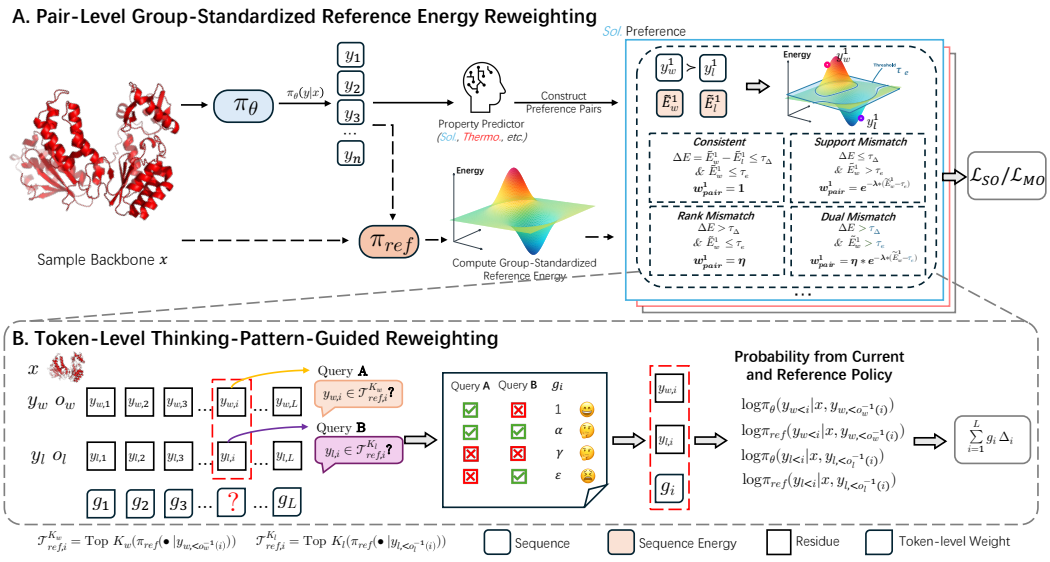


Figure 3: Overview of **Euterpe**. (A) Pair-level group-standardized reference energy reweighting adjusts each property preference according to its consistency with the pretrained IF reference distribution. (B) Token-Level Thinking-Pattern-Guided Reweighting further modulates residue-wise preference updates using the reference model's Top-K candidates.

the same stored order is used to compute the current- and reference-policy likelihoods, the reference energy, and the token-level thinking-pattern Top-K support. Therefore, no decoding order is resampled when constructing or optimizing a preference pair.

For property alignment, we consider K target properties, each associated with an in silico predictor M_k and a corresponding preference dataset $\mathcal{D}_k$. Following the semi-online training paradigm of (Liu et al., 2026), training alternates between rollout and optimization stages. At each round, we sample backbones x from the training set and generate N candidate sequences $\{y_j\}_{j=1}^{N}$ from the current policy, $y_j \sim \pi_\theta(\cdot \mid x), j = 1, \ldots, N$. For each property k, the generated sequences are scored by the corresponding predictor M_k and converted into preference pairs following (Liu et al., 2026; Xu et al., 2026), yielding the property-specific dataset $\mathcal{D}_k$. Pairs with insufficient property-score differences are discarded using the corresponding threshold δ_k.

4.2 Pair-Level Group-Standardized Reference-Support Reweighting

Property-based preference pairs specify the desired alignment direction, but they do not indicate whether the preferred sequence remains compatible with the pretrained IF model. We therefore use the frozen reference model π_{ref} to characterize each preference pair from two complementary perspectives: the relative ordering of the preferred and rejected sequences, and the reference support of the preferred sequence.

For the N candidate sequences generated from the same backbone, we compute the reference energy using Eq. 2 and standardize it within the rollout group:

$$\widetilde{E}_j = \frac{E_j - \mu_E}{\sigma_E + \epsilon}, \tag{5}$$

where $\epsilon = 0.1$, μ_E and σ_E denote the mean and standard deviation of the reference energies within the group. Since the reference-energy scale can vary across different backbones, group-wise standardization measures the support of each candidate relative to other sequences generated for the same backbone. A lower $\widetilde{E}_j$ therefore indicates stronger support from the pretrained IF model within the corresponding structure-conditioned rollout group.

For a preference pair $(y_w, y_l) \in \mathcal{D}_k$, let $\widetilde{E}_w$ and $\widetilde{E}_l$ denote the standardized reference energies of the preferred and rejected sequences, respectively, and define

$$\Delta E = \widetilde{E}_w - \widetilde{E}_l. \tag{6}$$

We use ΔE to characterize pairwise reference conflict and $\widetilde{E}_w$ to characterize the reference support of the preferred sequence. Specifically, $\Delta E > \tau_\Delta$ indicates that the reference ordering conflicts with the property preference beyond a tolerated margin, whereas $\widetilde{E}_w > \tau_e$ indicates that the preferred sequence lies in a relatively weakly supported region of the rollout group. Together, these two criteria define four reference states: *Consistent*, *Rank Mismatch*, *Support Mismatch*, and *Dual Mismatch*, as illustrated in Fig. 3(A). We assign the pair-level weight as

$$w_{\text{pair}} = \eta^{\mathbb{I}(\Delta E > \tau_\Delta)} \exp\left[-\lambda(\widetilde{E}_w - \tau_e)_+\right], \tag{7}$$

where $(a)_+ = \max(a, 0)$, τ_Δ controls the tolerance for pairwise reference conflict, τ_e specifies the reference-support threshold for the preferred sequence, $\eta \in (0, 1]$ controls the attenuation associated with rank mismatch, and $\lambda \geq 0$ controls the attenuation of weakly supported preferred sequences. The two factors capture complementary reference information, with ΔE reflecting pairwise reference conflict and $\widetilde{E}_w$ reflecting the support of the preferred sequence. Their attenuation effects are combined when both forms of mismatch occur.

4.3 Token-Level Thinking-Pattern-Guided Reweighting

Pair-level reweighting captures sequence-level compatibility with the pretrained IF model, while residue-wise preference signals within the same pair may still exhibit different local consistency. Following the notion of *thinking patterns* in (Li et al., 2026), we instantiate the pretrained IF model's local thinking pattern as its conditional residue preferences under each decoding context, represented by the corresponding Top-K residue candidates. We then use this local reference-support pattern to construct the token-level thinking-pattern-guided weight g_i. For each generated sequence, we retain the decoding order sampled during rollout and use the same order when querying the frozen reference model, ensuring that the token-level reference support is evaluated under the sequence's original decoding context.

For a preference pair (y_w, y_l), let o_w and o_l denote the decoding orders associated with the preferred and rejected sequences, respectively, where $o^{-1}(i)$ gives the decoding step corresponding to residue position i. At each position, we query the frozen reference model under the corresponding decoding context and retain its top-ranked residue candidates:

$$\mathcal{T}^w_{\text{ref},i} = \text{TopK}_{K_w}\left[\pi_{\text{ref}}\left(\cdot \mid x, y_{w,<o_w^{-1}(i)}\right)\right], \qquad \mathcal{T}^l_{\text{ref},i} = \text{TopK}_{K_l}\left[\pi_{\text{ref}}\left(\cdot \mid x, y_{l,<o_l^{-1}(i)}\right)\right], \tag{8}$$

where K_w and K_l denote the numbers of top-ranked residue candidates retained for the preferred and rejected sequences, respectively. These sets characterize the local reference support of the pretrained IF model under the corresponding decoding contexts.

For notational convenience, we define the corresponding support indicators as

$$m^w_i = \mathbb{I}\left[y_{w,i} \in \mathcal{T}^w_{\text{ref},i}\right], \qquad m^l_i = \mathbb{I}\left[y_{l,i} \in \mathcal{T}^l_{\text{ref},i}\right]. \tag{9}$$

We then assign a token-level thinking-pattern-guided weight according to the resulting local reference-support pattern:

$$g_i(m^w_i, m^l_i) = \begin{cases} 1, & m^w_i = 1, \quad m^l_i = 0, \\ \alpha, & m^w_i = 1, \quad m^l_i = 1, \\ \gamma, & m^w_i = 0, \quad m^l_i = 0, \\ \epsilon, & m^w_i = 0, \quad m^l_i = 1, \end{cases} \tag{10}$$

where we set $(1 > \alpha, \gamma, \epsilon > 0)$, such that the token-level weighting only attenuates preference signals relative to the fully supported case rather than amplifying them. The weighting reflects the consistency between the property preference and the local thinking pattern of the pretrained IF

model. Preferences supported only on the preferred side receive larger weights. When both residues are supported, the reference provides weaker discrimination, while support for neither residue indicates limited local evidence. Preferences supported only on the rejected side receive the smallest weight. Thus, g_i softly modulates residue-wise preference signals according to local reference support without imposing a hard constraint.

4.4 Multi-Level Reweighted Preference Optimization

Based on the residue-wise decomposition of the standard DPO objective in Eq. 18 from App. A.1, Euterpe modulates preference optimization at both the pair and residue levels. The pair-level weight w_{pair} controls the contribution of an entire preference pair, while the token-level weight g_i controls the contribution of each residue-wise preference signal. For a single target property, the resulting objective is denoted as:

$$\mathcal{L}_{\text{SO}} = -\mathbb{E}_{(x,y_w,y_l)}\left[w_{\text{pair}}(y_w,y_l)\log\sigma\left(\beta\sum_{i=1}^{L} g_i(m_i^w,m_i^l)\Delta_i\right)\right]. \tag{11}$$

A detailed optimization interpretation of the reweighting mechanisms is provided in App. A.2. For multi-property alignment, we apply **Euterpe** within the flexible-margin multi-objective formulation of (Liu et al., 2026):

$$\mathcal{L}_{\text{MO}} = -\sum_{k=1}^{K} w_k \mathbb{E}_{\mathcal{D}_k}\left[w_{\text{pair}}\log\sigma\left(\frac{\beta}{w_k}\sum_{i=1}^{L} g_i\Delta_i - \mathcal{M}_k\right)\right], \tag{12}$$

where $\mathcal{M}_k$ is the flexible cross-property margin (see App. A.2 for details).

5 Experiments

5.1 Experimental Setting

Dataset. We conduct all experiments on CATH 4.3 (Orengo et al., 1997), a widely used benchmark for protein inverse folding that contains experimentally determined protein structures. We adopt the standard train/validation/test split used in prior inverse-folding work (Dauparas et al., 2022; Gao et al., 2022; Zhu et al., 2026; Liu et al., 2026). Preference data are constructed from structures in the training split, while all reported results are evaluated on the held-out CATH 4.3 test set.

Training and Testing. For a fair comparison, Euterpe follows the training and evaluation protocol of Liu et al. (2026), including the rollout procedure, optimization schedule, property predictors, sampling settings, and evaluation metrics. We compare against representative pretrained inverse-folding, property-oriented post-training, and guidance-based methods. Additional implementation details, hyperparameter settings, and baseline configurations are provided in App. B.2, App. B.3, App. B.4, and App. B.1.

5.2 Euterpe Achieves Comparable Performance at Substantially Lower Cost

As shown in Tab. 1, **Euterpe** achieves performance comparable to MoMPNN (Liu et al., 2026) variants that explicitly use folding-derived structural feedback. Across both solubility and thermostability alignment, Euterpe closely matches MoMPNN in designability and AAR while maintaining comparable target-property performance. The same trend also holds when evolutionary plausibility (EP) is jointly optimized. These results suggest that the preservation effect of folding-derived feedback can be largely recovered from information already encoded in the pretrained IF model, without introducing an external folding model into the post-training loop. We further verify this trend using an independent predictor (App. C.3).

More importantly, **Euterpe** achieves this balance at substantially lower post-training cost. As shown in Tab. 2, removing the folding-derived structural feedback leads to a marked reduction in both runtime and computational cost, with up to a $6.9\times$ speedup and substantially fewer GFLOPs. The ad-

Table 1: Comparison of protein sequence design methods on the CATH 4.3 test set. **Fold. FB** denotes whether folding-derived structural feedback (TM score) is used during post-training.

Method	Targets	Fold. FB	Designability Metrics				Developability Metrics			AAR↑
			TM score↑	pLDDT↑	RMSD < 2 Å ↑	RMSD < 3 Å ↑	Sol↑	Thermo↑	EP↓	
Test Dataset	–	–	0.767	79.8	0.519	0.637	0.621	0.248	5.63	1.000
ProteinMPNN	–	–	0.746	78.1	0.510	0.619	0.719	0.773	6.63	0.384
ESM-IF	–	–	0.736	78.4	0.468	0.608	0.732	0.715	6.03	0.460
ProteinDPO	Thermo	No	0.719	75.3	0.439	0.579	0.521	0.958	7.94	0.356
Guidance	Sol	No	0.748	78.5	0.501	0.627	0.762	0.807	6.29	0.386
Guidance	Thermo	No	0.742	76.7	0.507	0.619	0.734	0.906	6.77	0.381
Solubility										
SolubleMPNN	Sol	No	0.738	77.6	0.491	0.611	0.794	0.815	6.48	0.377
MoMPNN	Sol + TM	**Yes**	0.744	78.2	0.486	0.605	0.882	0.746	6.30	0.380
Euterpe	Sol	**No**	0.741	78.1	0.499	0.620	**0.894**	0.787	6.35	0.377
MoMPNN	Sol + TM + EP	**Yes**	0.741	78.3	0.501	0.619	0.850	0.790	6.09	0.381
Euterpe	Sol + EP	**No**	0.740	77.9	0.497	0.617	**0.860**	0.808	**6.09**	0.378
Thermostability										
HyperMPNN	Thermo	No	0.707	73.2	0.477	0.576	0.721	0.930	7.72	0.355
MoMPNN	Thermo + TM	**Yes**	0.740	77.2	0.492	0.610	0.678	0.947	6.19	0.374
Euterpe	Thermo	**No**	0.738	77.1	0.487	0.611	0.693	**0.953**	5.97	0.372
MoMPNN	Thermo + TM + EP	**Yes**	0.742	77.9	0.500	0.616	0.718	0.897	6.06	0.382
Euterpe	Thermo + EP	**No**	0.741	77.7	0.494	0.619	0.776	**0.902**	**5.91**	0.378

Table 2: Mean computational cost per training epoch under different methods. Total time additionally includes checkpointing and synchronization overhead.

Target	Method	Cost	Rollout	Sol	Thermo	TM	Energy	EP	Train	Total
Sol	MoMPNN [Sol+TM]	Time (s)	4.1	33.3	–	1068.2	–	–	137.6	1243.3
		GFLOPs	3.47×10^6	2.00	–	2.58×10^7	–	–	3.44×10^6	3.27×10^7
	Euterpe [Sol]	Time (s)	4.7	32.7	–	–	7.9	–	135.1	**180.5**
		GFLOPs	3.47×10^6	2.00	–	–	3.45×10^4	–	3.44×10^6	$\mathbf{6.95\times10^6}$
Thermo	MoMPNN [Thermo+TM]	Time (s)	5.1	–	48.3	1066.9	–	–	135.7	1256.1
		GFLOPs	3.47×10^6	–	3.56×10^5	2.58×10^7	–	–	3.44×10^6	3.30×10^7
	Euterpe [Thermo]	Time (s)	7.5	–	50.6	–	6.0	–	130.1	**195.2**
		GFLOPs	3.47×10^6	–	3.56×10^5	–	3.45×10^4	–	3.44×10^6	$\mathbf{7.30\times10^6}$
Sol+EP	MoMPNN [Sol+TM+EP]	Time (s)	4.5	34.1	–	1099.3	–	1309.7	138.8	2586.6
		GFLOPs	3.47×10^6	2.00	–	2.58×10^7	–	1.84×10^8	3.44×10^6	2.17×10^8
	Euterpe [Sol+EP]	Time (s)	5.0	30.7	–	–	14.2	1315.9	138.7	**1504.7**
		GFLOPs	3.47×10^6	2.00	–	–	3.45×10^4	1.84×10^8	3.44×10^6	$\mathbf{1.91\times10^8}$
Thermo+EP	MoMPNN [Thermo+TM+EP]	Time (s)	4.1	–	38.3	1099.0	–	1311.3	138.1	2591.0
		GFLOPs	3.47×10^6	–	3.56×10^5	2.58×10^7	–	1.84×10^8	3.44×10^6	2.17×10^8
	Euterpe [Thermo+EP]	Time (s)	5.0	–	38.9	–	10.0	1315.2	131.9	**1501.1**
		GFLOPs	3.47×10^6	–	3.56×10^5	–	3.45×10^4	1.84×10^8	3.44×10^6	$\mathbf{1.91\times10^8}$

ditional reference-energy computation introduces only minor overhead. Thus, **Euterpe** replaces expensive external folding-derived structural feedback with reference information from the pretrained IF model while maintaining a similar trade-off between property alignment and preservation of pretrained IF performance. We further conduct cross-benchmark evaluation on CASP15 TS45, without additional training or tuning (App. C.2).

5.3 COMPARISON WITH GLOBAL KL-BASED PRESERVATION

We compare **Euterpe** with two baselines: 1) the standard DPO objective with the global coefficient β tuned, and 2) an explicit KL-regularized variant with varying λ_{KL} (App. B.1). As shown in Fig. 4, stronger global reference regularization generally shifts the model from property alignment toward preservation of the pretrained policy. At comparable solubility, **Euterpe** achieves higher TM score and RMSD-based success rates while maintaining similar pLDDT. Explicit KL yields higher AAR under strong regularization, which is expected because it directly constrains the aligned policy toward the pretrained ProteinMPNN distribution, thereby favoring native-sequence recovery. In contrast, **Euterpe** applies reference-based regulation selectively at both the pair and token levels rather than uniformly restricting distributional deviation, allowing greater property-oriented adaptation while better preserving structural compatibility. Overall, these results highlight the advantage of **Euterpe** in achieving a more favorable trade-off.

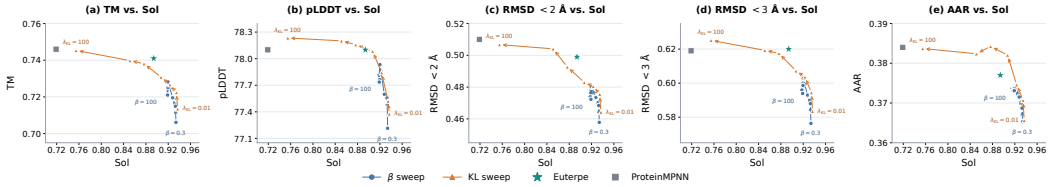
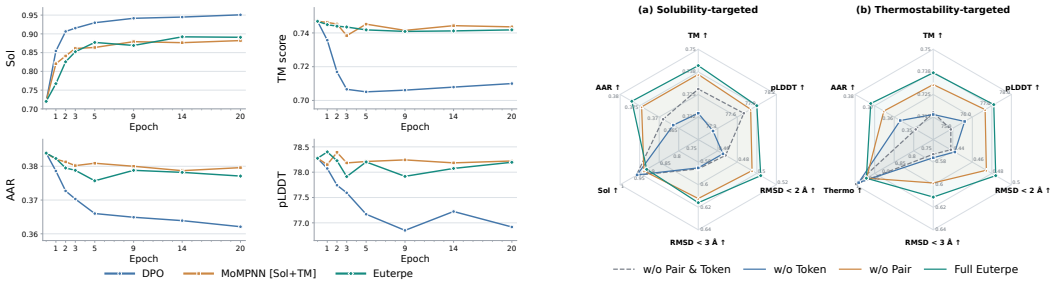


Figure 4: **Global KL-based preservation baseline.** We compare standard DPO with different β values $(0.3, 0.5, 0.7, 2, 3, 20, 100)$ and standard DPO augmented with explicit token-level KL regularization ($\lambda_{\mathrm{KL}} = 0.01, 0.1, 1.0, 10, 100$). The arrows indicate these settings in the order listed above.



(A) (B)

Figure 5: Analysis of **Euterpe**. (A) Optimization dynamics during solubility alignment. (B) Ablation of pair-level and token-level thinking-pattern-guided reweighting under solubility and thermostability alignment. Removing both reweighting reduces it to standard property-only DPO.

5.4 Ablation and Optimization Dynamics

We further analyze the contribution of the two reweighting mechanisms in Fig. 5(B). Removing both components reduces **Euterpe** to property-only preference optimization and substantially degrades pretrained IF performance. The token-level reweighting accounts for most of the preservation effect: removing it leads to consistent drops in designability and AAR under both solubility- and thermostability-targeted alignment, whereas removing the pair-level component retains most of these gains. Pair-level reweighting nevertheless provides complementary improvements when combined with the token-level mechanism.

Fig. 5(A) further shows how this effect evolves during optimization. Property-only DPO rapidly improves solubility, but progressively degrades TM score, pLDDT, and AAR. By epoch 2, it already surpasses the final solubility score of **Euterpe**, yet still yields worse preservation metrics, indicating that **Euterpe**'s advantage cannot be explained merely by early checkpoint selection. Throughout optimization, **Euterpe** maintains more stable pretrained IF performance and follows a trajectory close to MoMPNN [Sol+TM], without using folding-derived structural feedback. Hyperparameter sensitivity analyses are provided in App. C.

6 Conclusion

We introduce **Euterpe**, an efficient semi-online preference optimization framework for protein inverse folding. Our results show that folding-derived structural feedback during developability alignment mainly serves to preserve pretrained IF performance rather than consistently improve it beyond pretraining. Based on this observation, Euterpe uses the pretrained IF model itself as a reference prior to regulate preference optimization at both the sequence and residue levels. **Euterpe** achieves comparable property alignment, designability, and AAR to folding-feedback-based methods, while requiring fewer GFLOPs and achieving up to a $6.9\times$ speedup. These results suggest that pretrained IF models can provide effective preservation signals without costly folding-based feedback.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the implementation of research code. We did not use generative AI tools for developing the research methodology, designing experiments, formulating mathematical claims, or interpreting experimental results, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for code editing and debugging, as well as manuscript writing and language polishing to improve grammatical correctness, clarity, and readability. We have reviewed all AI-assisted work. In particular, all AI-assisted code was manually inspected, tested, and validated by the authors for correctness before being used in the experiments, and all AI-assisted text was carefully reviewed to ensure that it accurately reflects the intended scientific content. All scientific ideas, methodological designs, experiments, analyses, results, and conclusions were developed and verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the information necessary to reproduce our experiments throughout the main paper and appendix. The datasets, data splits, evaluation protocols, baseline configurations, training settings, hyperparameters, sampling configurations, and implementation details are provided in Appendix B. We report nearly all hyperparameters and experimental settings used in our experiments, providing sufficient detail to enable faithful reproduction of the reported results. We also describe the protocol for repeated runs and evaluation to ensure consistency across experiments.

REFERENCES

Nathaniel Blalock, Srinath Seshadri, Agrim Babber, Sarah Fahlberg, and Philip Romero. Functional alignment of protein language models via reinforcement learning with experimental feedback.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

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

Moritz Ertelt, Phillip Schlegel, Max Beining, Leonard Kaysser, Jens Meiler, and Clara T Schoeder. Hypermpnn–a general strategy to design thermostable proteins learned from hyperthermophiles. *bioRxiv*, 2024.

Zhangyang Gao, Cheng Tan, Pablo Chacón, and Stan Z Li. Pifold: Toward effective and efficient protein inverse folding. *arXiv preprint arXiv:2209.12643*, 2022.

Zhangyang Gao, Cheng Tan, Yijie Zhang, Xingran Chen, Lirong Wu, and Stan Z Li. Proteininvbench: Benchmarking protein inverse folding on diverse tasks, models, and metrics. *Advances in Neural Information Processing Systems*, 36:68207–68220, 2023.

Zhangyang Gao, Cheng Tan, Xingran Chen, Yijie Zhang, Jun Xia, Siyuan Li, and Stan Z Li. Kwdesign: Pushing the limit of protein design via knowledge refinement. In *International Conference on Learning Representations*, volume 2024, pp. 34556–34574, 2024.

Casper A Goverde, Martin Pacesa, Nicolas Goldbach, Lars J Dornfeld, Petra EM Balbi, Sandrine Georgeon, Stéphane Rosset, Srajan Kapoor, Jagrity Choudhury, Justas Dauparas, et al. Computational design of soluble and functional membrane protein analogues. *Nature*, 631(8020):449–458, 2024.

Max Hebditch, M Alejandro Carballo-Amador, Spyros Charonis, Robin Curtis, and Jim Warwicker. Protein–sol: a web tool for predicting protein solubility from sequence. *Bioinformatics*, 33(19): 3098–3100, 2017.

Jiri Hon, Martin Marusiak, Tomas Martinek, Antonin Kunka, Jaroslav Zendulka, David Bednar, and Jiri Damborsky. Soluprot: prediction of soluble protein expression in escherichia coli. *Bioinformatics*, 37(1):23–28, 2021.

Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *International conference on machine learning*, pp. 8946–8970. PMLR, 2022.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

Michael Jendrusch and Jan O Korbel. Efficient protein structure generation with sparse denoising models. *Nature machine intelligence*, 7(9):1429–1445, 2025.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

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

Junqi Liu, Xiaoyang Hou, Chence Shi, Xin Liu, Zhi Yang, and Jian Tang. Property-driven protein inverse folding with multi-objective preference alignment. In *International Conference on Learning Representations*, volume 2026, pp. 151064–151087, 2026.

Pouria Mistani and Venkatesh Mysore. Preference optimization of protein language models as a multi-objective binder design paradigm. *arXiv preprint arXiv:2403.04187*, 2024.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. In *International Conference on Learning Representations*, volume 2025, pp. 36052–36106, 2025.

Christine A Orengo, Alex D Michie, Susan Jones, David T Jones, Mark B Swindells, and Janet M Thornton. Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

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

Jiezhong Qiu, Junde Xu, Jie Hu, Hanqun Cao, Liya Hou, Zijun Gao, Xinyi Zhou, Anni Li, Xiujuan Li, Bin Cui, et al. Instructplm: Aligning protein language models to follow protein structure instructions. *bioRxiv*, pp. 2024–04, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Milong Ren, Chungong Yu, Dongbo Bu, and Haicang Zhang. Accurate and robust protein sequence design with carbondesign. *Nature Machine Intelligence*, 6(5):536–547, 2024.

Chiara Rodella, Symela Lazaridi, and Thomas Lemmin. Temberture: advancing protein thermostability prediction with deep learning and attention mechanisms. *Bioinformatics Advances*, 4(1): vbae103, 2024.

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

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

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Filippo Stocco, Maria Artigues-Lleixa, Andrea Hunklinger, Talal Widatalla, Marc Guell, and Noelia Ferruz. Guiding generative protein language models with reinforcement learning. *arXiv preprint arXiv:2412.12979*, 2024.

Kiera H Sumida, Reyes Núñez-Franco, Indrek Kalvet, Samuel J Pellock, Basile IM Wicky, Lukas F Milles, Justas Dauparas, Jue Wang, Yakov Kipnis, Noel Jameson, et al. Improving protein expression, stability, and function with proteinmpnn. *Journal of the American Chemical Society*, 146(3):2054, 2024.

Ziwen Wang, Jiajun Fan, Ruihan Guo, Thao Nguyen, Heng Ji, and Ge Liu. Proteinzero: Self-improving protein generation via online reinforcement learning. *arXiv preprint arXiv:2506.07459*, 2025.

Talal Widatalla, Rafael Rafailov, and Brian Hie. Aligning protein generative models with experimental fitness via direct preference optimization. *bioRxiv*, pp. 2024–05, 2024.

Junde Xu, Zijun Gao, Xinyi Zhou, Xingyi Cheng, Le Song, Guangyong Chen, Pheng-Ann Heng, and Jiezhong Qiu. Protein inverse folding from structure feedback. *Advances in Neural Information Processing Systems*, 38:24274–24299, 2026.

Fanglei Xue, Andrew Kubaney, Zhichun Guo, Joseph K Min, Ge Liu, Yi Yang, and David Baker. Improving protein sequence design through designability preference optimization. *arXiv preprint arXiv:2506.00297*, 2025.

Wenwu Zeng, Xiaoyu Li, Haitao Zou, Yutao Dou, Xiongjun Zhao, and Shaoliang Peng. Symmetric self-play online preference optimization for protein inverse folding. *bioRxiv*, pp. 2026–03, 2026.

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

Yulin Zhang, He Cao, Zihao Jiang, Chenyi Zi, Zhipeng Zhou, Zijing Liu, Yu Li, Jia Li, and Ziqi Gao. Proteinopd: Towards effective and efficient preference alignment for protein design. *arXiv preprint arXiv:2605.10189*, 2026.

Zaixiang Zheng, Yifan Deng, Dongyu Xue, Yi Zhou, Fei Ye, and Quanquan Gu. Structure-informed language models are protein designers. In *International conference on machine learning*, pp. 42317–42338. PMLR, 2023.

Yiran Zhu, Changxi Chi, Hongxin Xiang, Wenjie Du, Xiaoqi Wang, and Jun Xia. Refold: Refining protein inverse folding with efficient structural matching and fusion. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 13147–13156, 2026.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Appendix

A.1 Token-Level Decomposition of DPO

Given a protein backbone x, ProteinMPNN factorizes the likelihood of a sequence $y = (y_1, \ldots, y_L)$ according to a decoding order o as

$$\pi_\theta(y \mid x, o) = \prod_{i=1}^{L} \pi_\theta\left(y_i \mid x, y_{<o^{-1}(i)}, o\right), \tag{13}$$

where i denotes the residue index in the original sequence order and $o^{-1}(i)$ denotes the decoding step of residue i. For simplicity, we omit o from the conditioning variables of each token probability below.

For a preference pair (y_w, y_l) with decoding orders o_w and o_l, respectively, the standard DPO objective is

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y_w,y_l)} \log \sigma\left[\beta\left(\log \frac{\pi_\theta(y_w \mid x, o_w)}{\pi_{\text{ref}}(y_w \mid x, o_w)} - \log \frac{\pi_\theta(y_l \mid x, o_l)}{\pi_{\text{ref}}(y_l \mid x, o_l)}\right)\right]. \tag{14}$$

Using Eq. 13, we have

$$\begin{aligned}\log \frac{\pi_\theta(y_w \mid x, o_w)}{\pi_{\text{ref}}(y_w \mid x, o_w)} = \sum_{i=1}^{L} \Big[&\log \pi_\theta\left(y_{w,i} \mid x, y_{w,<o_w^{-1}(i)}\right) \\ &- \log \pi_{\text{ref}}\left(y_{w,i} \mid x, y_{w,<o_w^{-1}(i)}\right)\Big],\end{aligned} \tag{15}$$

and similarly for y_l. Substituting both decompositions into Eq. 14 gives the token-level form of DPO:

$$\begin{aligned}\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y_w,y_l)} \log \sigma\Bigg[\beta \sum_{i=1}^{L} \Bigg(&\log \pi_\theta\left(y_{w,i} \mid x, y_{w,<o_w^{-1}(i)}\right) - \log \pi_{\text{ref}}\left(y_{w,i} \mid x, y_{w,<o_w^{-1}(i)}\right) \\ &- \log \pi_\theta\left(y_{l,i} \mid x, y_{l,<o_l^{-1}(i)}\right) + \log \pi_{\text{ref}}\left(y_{l,i} \mid x, y_{l,<o_l^{-1}(i)}\right)\Bigg)\Bigg].\end{aligned} \tag{16}$$

For notational convenience, we define

$$\begin{aligned}\Delta_i = \; &\log \pi_\theta(y_{w,i} \mid x, y_{w,<o_w^{-1}(i)}) - \log \pi_{\text{ref}}(y_{w,i} \mid x, y_{w,<o_w^{-1}(i)}) \\ &- \log \pi_\theta(y_{l,i} \mid x, y_{l,<o_l^{-1}(i)}) + \log \pi_{\text{ref}}(y_{l,i} \mid x, y_{l,<o_l^{-1}(i)}),\end{aligned} \tag{17}$$

such that

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y_w,y_l)} \log \sigma\left(\beta \sum_{i=1}^{L} \Delta_i\right). \tag{18}$$

Therefore, the sequence-level DPO objective can be equivalently expressed as the sum of residue-wise policy log-ratio differences. Moreover, the winner and loser sequences retain their respective rollout orders, o_w and o_l, and each policy–reference log-ratio is computed under the same stored order for both models.

A.2 Optimization Interpretation

Distributional interpretation of pair-level reweighting. The effect of w_{pair} can also be understood from the perspective of the empirical preference distribution. Let $\mathcal{D}$ denote the preference distribution within a fixed optimization stage, and let $\ell_{\text{DPO}}(p)$ denote the per-pair DPO loss in Eq. 18, where $p = (x, y_w, y_l)$ represents a preference pair. The pair-level weighted objective can be written as

$$\mathcal{L}_{\text{pair}} = \mathbb{E}_{p\sim\mathcal{D}}\left[w_{\text{pair}}(p)\, \ell_{\text{DPO}}(p)\right]. \tag{19}$$

Since $w_{\text{pair}}(p) > 0$ is computed from the frozen reference model and treated as fixed with respect to θ, define

$$C_w = \mathbb{E}_{p\sim\mathcal{D}}\left[w_{\text{pair}}(p)\right], \tag{20}$$

and

$$\mathcal{D}_w(p) = \frac{w_{\text{pair}}(p)}{C_w}\mathcal{D}(p). \tag{21}$$

By construction, $\mathcal{D}_w$ is a normalized distribution. Therefore,

$$\mathcal{L}_{\text{pair}} = C_w \mathbb{E}_{p\sim\mathcal{D}_w}\left[\ell_{\text{DPO}}(p)\right]. \tag{22}$$

Because $C_w > 0$ is independent of θ within the optimization stage, pair-level reweighting is equivalent, up to a positive constant factor, to optimizing the standard DPO loss under the reweighted preference distribution $\mathcal{D}_w$.

This view clarifies that w_{pair} does not modify the DPO objective within an individual preference pair. Instead, it changes the relative contribution of different preference pairs to optimization. In the semi-online setting, this interpretation applies within each optimization stage, since the preference distribution is refreshed after subsequent rollouts.

Residue-wise decomposition of DPO. For an order-agnostic autoregressive inverse-folding model such as ProteinMPNN, we condition on a sampled decoding order o. The conditional sequence probability can be factorized as

$$\pi_\theta(y \mid x, o) = \prod_{i=1}^{L} \pi_\theta\left(y_i \mid x, y_{<o^{-1}(i)}\right), \tag{23}$$

where $y_{<o^{-1}(i)}$ denotes the residues decoded before position i under order o. Therefore,

$$\log \frac{\pi_\theta(y \mid x, o)}{\pi_{\text{ref}}(y \mid x, o)} = \sum_{i=1}^{L}\left[\log \pi_\theta\left(y_i \mid x, y_{<o^{-1}(i)}\right) - \log \pi_{\text{ref}}\left(y_i \mid x, y_{<o^{-1}(i)}\right)\right]. \tag{24}$$

For a preference pair (y_w, y_l), we define the residue-wise policy–reference log-ratio difference as

$$\begin{aligned}\Delta_i = &\log \pi_\theta\left(y_{w,i} \mid x, y_{w,<o_w^{-1}(i)}\right) - \log \pi_{\text{ref}}\left(y_{w,i} \mid x, y_{w,<o_w^{-1}(i)}\right) \\ &- \log \pi_\theta\left(y_{l,i} \mid x, y_{l,<o_l^{-1}(i)}\right) + \log \pi_{\text{ref}}\left(y_{l,i} \mid x, y_{l,<o_l^{-1}(i)}\right).\end{aligned} \tag{25}$$

Using Eq. 24, the sequence-level DPO margin can then be written as

$$\log \frac{\pi_\theta(y_w \mid x, o_w)}{\pi_{\text{ref}}(y_w \mid x, o_w)} - \log \frac{\pi_\theta(y_l \mid x, o_l)}{\pi_{\text{ref}}(y_l \mid x, o_l)} = \sum_{i=1}^{L} \Delta_i. \tag{26}$$

Hence, conditioned on the sampled decoding orders, the standard DPO objective (Rafailov et al., 2023) can be equivalently expressed as

$$\ell_{\text{DPO}} = -\log \sigma\left(\beta \sum_{i=1}^{L} \Delta_i\right). \tag{27}$$

This decomposition does not introduce independent residue-level preference labels; it simply rewrites the original sequence-level DPO margin as the sum of its autoregressive residue-wise components.

Optimization effect of multi-level reweighting. Based on the above decomposition, **Euterpe** introduces pair-level and residue-level reweighting:

$$\ell_{\text{Euterpe}} = -w_{\text{pair}} \log \sigma(z), \qquad z = \beta \sum_{i=1}^{L} g_i \Delta_i. \tag{28}$$

Both w_{pair} and g_i are treated as fixed reweighting coefficients during each optimization step and do not receive gradients.

Using

$$\frac{\partial[-\log\sigma(z)]}{\partial z} = -\sigma(-z), \tag{29}$$

and

$$\nabla_\theta z = \beta \sum_{i=1}^{L} g_i \nabla_\theta \Delta_i, \tag{30}$$

we obtain

$$\nabla_\theta \ell_{\text{Euterpe}} = -w_{\text{pair}} \beta \sigma(-z) \sum_{i=1}^{L} g_i \nabla_\theta \Delta_i. \tag{31}$$

For comparison, standard DPO gives

$$\nabla_\theta \ell_{\text{DPO}} = -\beta\sigma(-z_{\text{DPO}}) \sum_{i=1}^{L} \nabla_\theta \Delta_i, \qquad z_{\text{DPO}} = \beta \sum_{i=1}^{L} \Delta_i. \tag{32}$$

Equations 31 and 32 clarify the distinct roles of the two reweighting levels. Since w_{pair} multiplies the entire pairwise objective, it directly scales the overall gradient contribution of the corresponding preference pair without changing its DPO margin z. Preference pairs that are less consistent with the pretrained reference can therefore be attenuated as a whole.

In contrast, g_i operates inside the residue-wise aggregation. It changes the relative contribution of $\nabla_\theta \Delta_i$ from different residue positions before they are combined into the final parameter update. Consequently, reference-supported residue-wise preference signals can retain larger contributions, whereas less-supported or conflicting signals can be selectively attenuated.

Importantly, token-level reweighting is not equivalent to independently multiplying a separately computed gradient for each residue. Since g_i also enters $z = \beta \sum_i g_i \Delta_i$, it additionally changes the common sigmoid factor $\sigma(-z)$ shared by all residue-wise components. Thus, g_i affects both the relative composition of the residue-wise update and the overall pairwise gradient through the aggregated DPO margin. The token-level mechanism is therefore more precisely interpreted as reweighting the residue-wise composition of the preference signal before the sequence-level DPO objective is evaluated.

Relation to the global DPO coefficient. DPO can be derived from the KL-regularized policy optimization objective (Rafailov et al., 2023),

$$\max_{\pi} \; \mathbb{E}_{x,y\sim\pi(\cdot|x)} \left[r(x,y)\right] - \beta D_{\text{KL}} \left(\pi(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)\right). \tag{33}$$

Its optimal policy has the form

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{r(x,y)}{\beta}\right), \tag{34}$$

where $Z(x)$ is the normalization constant. Therefore, β acts as a global coefficient controlling the trade-off between reward optimization and deviation from the reference policy, and is shared uniformly across preference pairs and residue positions.

In the practical DPO objective, β also uniformly scales the aggregated preference margin,

$$z_{\text{DPO}} = \beta \sum_{i=1}^{L} \Delta_i, \tag{35}$$

and hence affects both the overall gradient scale and the common sigmoid factor in Eq. equation 32. It does not, however, introduce pair- or residue-specific relative weighting.

By contrast, w_{pair} and g_i are determined separately for individual preference pairs and residue positions. The former controls the overall contribution of a preference pair, while the latter changes the relative composition of residue-wise preference signals within the pair.

To make the distinction between g_i and β explicit, consider the special case

$$g_i = c, \qquad \forall i. \tag{36}$$

Then,

$$\beta \sum_{i=1}^{L} g_i \Delta_i = (\beta c) \sum_{i=1}^{L} \Delta_i. \tag{37}$$

For this preference pair, token-level reweighting is therefore equivalent to replacing β with an effective coefficient $\beta_{\text{eff}} = \beta c$. If the same constant c were shared across all residue positions and all preference pairs, it would reduce to a global rescaling of the standard DPO coefficient.

Euterpe does not operate in this regime. In general, g_i varies across residue positions according to the local reference-support pattern. Hence,

$$\sum_i g_i \nabla_\theta \Delta_i \not\equiv c \sum_i \nabla_\theta \Delta_i \tag{38}$$

for any single scalar c. Heterogeneous token-level reweighting can therefore change the relative contribution, and in general the direction, of the aggregated residue-wise update rather than merely rescaling its global magnitude. It cannot, in general, be represented by a single scalar adjustment of the global DPO coefficient.

Flexible margin for multi-property alignment. For multi-property alignment, we follow the flexible-margin formulation of Liu et al. (2026). Let $s_k(x, y)$ denote the raw score of property k. We convert each property score into a reward with a consistent "higher-is-better" direction:

$$r_k(x, y) = \begin{cases} c_k \, s_k(x, y), & \text{if higher values are preferred,} \\ -c_k \, s_k(x, y), & \text{if lower values are preferred,} \end{cases} \tag{39}$$

where $c_k > 0$ is a property-specific scaling coefficient. Among the properties considered in this work, **only evolutionary plausibility (EP), measured by perplexity, is lower-is-better and therefore uses the negative form**. Solubility and thermostability use the positive form.

For the preference dataset $\mathcal{D}_k$ constructed with respect to property k, we then define the cross-property margin as

$$\mathcal{M}_k(x, y_w, y_l) = \frac{1}{w_k} \sum_{k' \neq k} w_{k'} \left[r_{k'}(x, y_w) - r_{k'}(x, y_l) \right], \tag{40}$$

where w_k denotes the weight assigned to property k.

B Implementation

B.1 Comparison Methods

We compare **Euterpe** with representative methods from three categories: (1) pretrained inverse-folding models, including ProteinMPNN (Dauparas et al., 2022), ESM-IF (Hsu et al., 2022); (2) property-oriented post-training methods, including ProteinDPO (Widatalla et al., 2024), SolubleMPNN (Goverde et al., 2024), HyperMPNN (Ertelt et al., 2024), and MoMPNN (Liu et al., 2026); (3) guidance-based methods, Guidance [Sol] and Guidance [Thermo] (Nisonoff et al., 2025). During each evaluation, we sample one sequence for each test backbone. To account for sampling randomness, we repeat the entire evaluation five times with independent resampling and report the average performance across the five evaluations. For a controlled comparison, we also retrained MoMPNN under the same training and evaluation settings used in our experiments.

We additionally construct a DPO+KL baseline to examine whether directly regularizing the policy toward the pretrained ProteinMPNN can recover the preservation effect of Euterpe. The training objective is

$$\mathcal{L}_{\text{DPO+KL}} = \mathcal{L}_{\text{DPO}} + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}}. \tag{41}$$

For every batch, we apply the KL regularizer uniformly to all preferred and rejected sequences. Specifically, the winner and loser sequences are treated identically for the KL term, without

Table 3: Pretrained IF backbones underlying the evaluated methods.

Base Model \ Method	ProteinDPO	SolubleMPNN	HyperMPNN	Guidance	MoMPNN	**Euterpe**
ESM-IF1	✓					
ProteinMPNN		✓	✓	✓	✓	✓

preference-role-specific weighting. For each valid designable residue, we compute the token-level KL divergence between the current policy and the frozen reference model under the same backbone, teacher-forced sequence context, and decoding order:

$$\mathrm{KL}_i = \sum_{a \in \mathcal{A}} \pi_\theta(a \mid x, y^\sigma_{<i}, \sigma) \log \frac{\pi_\theta(a \mid x, y^\sigma_{<i}, \sigma)}{\pi_{\mathrm{ref}}(a \mid x, y^\sigma_{<i}, \sigma)}, \tag{42}$$

where $\mathcal{A}$ denotes the 21-token amino-acid vocabulary. The KL loss is averaged over all valid designable residues from both preferred and rejected sequences in the batch, without preference-role-specific weighting. The reference model remains frozen, while gradients are propagated only through the policy.

B.2 Property Calculation

Structural Designability. We use ESMFold (Lin et al., 2023) to predict the three-dimensional structure of each designed sequence. Structural similarity to the target backbone is evaluated using TM-score computed by TM-align (Zhang & Skolnick, 2005). We additionally extract pLDDT from the ESMFold prediction as a measure of structural confidence.

Evolutionary Plausibility. We evaluate evolutionary plausibility using the `esm2_t33_650M_UR50D` model from ESM-2 (Lin et al., 2023). Following prior work (Liu et al., 2026), we compute sequence pseudo-perplexity (**EP**) from the model likelihoods. Lower EP indicates greater compatibility with the sequence distribution captured by the pretrained protein language model.

Solubility. We use Protein-Sol (Hebditch et al., 2017) as the predictor for protein solubility. Protein-Sol assigns each designed sequence a solubility score (**Sol**), with larger values indicating higher predicted solubility. We additionally evaluate the generated sequences using SoluProt (Hon et al., 2021) as an independent solubility predictor (App. C.3).

Thermostability. We use TemBERTure (Rodella et al., 2024) to evaluate thermostability. We adopt its classification output and use the predicted thermophilic probability as the thermostability score (**Thermo**), where larger values indicate higher predicted thermostability.

Scope of property evaluation. Following prior work (Liu et al., 2026), we use the same property predictors for preference construction and target-property evaluation, ensuring a consistent and directly comparable evaluation protocol. Accordingly, Sol and Thermo are used as computational measures of property alignment throughout our experiments. Our focus is on whether pretrained inverse-folding performance can be effectively preserved under such property alignment without relying on expensive folding-derived structural feedback. To this end, preservation is evaluated separately using structural and sequence-level metrics.

B.3 CATH 4.3 Benchmark

Data. We use the CATH 4.3 benchmark (Orengo et al., 1997) with the standard train/validation/test split adopted in prior inverse-folding work (Hsu et al., 2022; Gao et al., 2023). All preference pairs used for post-training are constructed exclusively from structures in the training split. No structure or native sequence from the validation or test split is used for preference construction or post-training. All reported results are evaluated on the held-out test split. Each example consists of an experimentally determined protein backbone and its corresponding native amino-acid sequence.

Table 4: Standard deviations induced by five independent inference-time sequence-resampling evaluations

Method	Targets	Designability Metrics				Developability Metrics			AAR
		TM score	pLDDT	RMSD < 2 Å	RMSD < 3 Å	Sol	Thermo	EP	
Solubility									
MoMPNN	Sol + TM	0.0018	0.0825	0.0049	0.0052	0.0013	0.0050	0.0118	0.0003
Euterpe	Sol	0.0017	0.0941	0.0022	0.0033	0.0017	0.0042	0.0143	0.0002
MoMPNN	Sol + TM + EP	0.0019	0.0542	0.0064	0.0089	0.0015	0.0027	0.0141	0.0004
Euterpe	Sol + EP	0.0031	0.0604	0.0052	0.0068	0.0014	0.0026	0.0079	0.0002
Thermostability									
MoMPNN	Thermo + TM	0.0027	0.0642	0.0077	0.0067	0.0017	0.0042	0.0142	0.0004
Euterpe	Thermo	0.0015	0.0823	0.0089	0.0065	0.0014	0.0021	0.0038	0.0003
MoMPNN	Thermo + TM + EP	0.0020	0.0954	0.0058	0.0094	0.0008	0.0057	0.0211	0.0006
Euterpe	Thermo + EP	0.0028	0.1120	0.0052	0.0055	0.0020	0.0043	0.0050	0.0002

Evaluation Metrics. For each designed sequence, we predict its structure using ESMFold (Lin et al., 2023) and compare the predicted backbone with the target structure. We evaluate both structural designability and sequence-level developability using the following metrics.

- **TM-score** measures the global structural similarity between the ESMFold-predicted structure and the target backbone. Higher values indicate better recovery of the target fold.
- **RMSD success rate** measures the fraction of designed sequences whose predicted structures recover the target backbone within a specified Cα RMSD threshold after structural alignment. Threshold-based success rates are commonly used to quantify protein designability, with RMSD cutoffs around 2–3 Å adopted in prior protein design benchmarks (Jendrusch & Korbel, 2025). We therefore report the fractions of designs with RMSD below 2 Å and 3 Å, denoted RMSD < 2 Å and RMSD < 3 Å, respectively.
- **pLDDT** is the mean confidence score of the ESMFold-predicted structure. Higher values indicate greater confidence in the predicted fold.
- **EP** denotes the ESM-2 pseudo-perplexity. Lower values indicate greater evolutionary plausibility.
- **Sol** and **Thermo** denote the solubility and thermostability scores predicted by Protein-Sol and TemBERTure, respectively.
- **AAR** measures amino-acid recovery with respect to the native sequence:

$$\mathrm{AAR} = \frac{1}{L}\sum_{i=1}^{L}\mathbb{I}(y_i = \hat{y}_i), \tag{43}$$

  where L is the sequence length, y_i is the native residue at position i, and $\hat{y}_i$ is the corresponding designed residue.

All metrics are first computed for individual generated sequences and then aggregated over the CATH 4.3 test set.

B.4 Hyperparameter Settings

For all semi-online DPO-based methods, we use the same rollout and optimization protocol. At each epoch, 256 backbones are randomly sampled from the training set, and eight sequences are generated for each backbone at a sampling temperature of 1.0. The sampled sequences are ranked according to the target property, with rank 1 denoting the most preferred sequence, and paired by rank as $(1, 5)$, $(2, 6)$, $(3, 7)$, and $(4, 8)$, yielding four candidate preference pairs per backbone.

Before optimization, the candidate pairs are filtered using a property-specific threshold δ_k. Let $s_k(y)$ denote the score of property k. For higher-is-better properties, a pair is retained only if

$$s_k(y_w) - s_k(y_l) > \delta_k, \tag{44}$$

Table 5: Hyperparameter settings of **Euterpe**.

Hyperparameter	Solubility	Thermostability
β	0.5	0.5
Pair-level reweighting		
η	0.95	0.95
λ	1.0	1.0
τ_Δ	0.1	-0.2
τ_e	-0.2	-0.2
δ	0.05	0.02
Token-level reweighting		
K_w	3	3
K_l	3	3
α	0.7	0.7
ϵ	0.1	0.1
γ	0.2	0.3

whereas for lower-is-better properties, it is retained only if

$$s_k(y_l) - s_k(y_w) > \delta_k. \tag{45}$$

This filtering removes pairs with insufficient separation in property scores before DPO training.

The model is then optimized for 400 steps with a batch size of 64. A new rollout and preference dataset are constructed at the beginning of each epoch, and this semi-online procedure is repeated for 20 epochs. We use Adam with a learning rate of 5×10^{-6}, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. The final checkpoint after 20 epochs is used for evaluation, where sequences are sampled at temperature 0.1.

The hyperparameter configurations used for the different properties experiments are summarized in Tab. 5 and Tab. 6.The property-specific threshold δ is used to filter preference pairs, retaining a winner–loser pair only when their property-score difference exceeds δ. Since solubility and thermostability scores lie in $[0, 1]$, we set $\delta = 0.05$ and $\delta = 0.02$ for solubility and thermostability, respectively. For EP, which is measured by perplexity and has a different numerical scale, we use $\delta = 0.5$.

C Robustness Experiments

C.1 Hyperparameter Sensitivity Analysis

Sensitivity to token-level Top-K candidate sizes. We examine the sensitivity of **Euterpe** to the preferred- and rejected-side candidate-set sizes, K_w and K_l, used in token-level thinking-pattern-guided reweighting. As shown in Fig. 6, different configurations induce a clear trade-off between solubility alignment and preservation of pretrained IF performance. Smaller candidate sets impose stronger preservation, at the cost of relatively lower solubility, whereas the less selective $5/5$ configuration achieves higher solubility accompanied by larger decreases in TM score, pLDDT, and AAR. $K_w = K_l = 3$ lies in the intermediate regime and achieves a balanced trade-off across these metrics. Overall, the observed trends are smooth across the tested configurations, indicating that the behavior of **Euterpe** is not tied to a particular choice of K_w and K_l.

Sensitivity to pair-level reweighting thresholds. We next examine the sensitivity of **Euterpe** to the pair-level thresholds τ_Δ and τ_e. As shown in Fig. 7, performance varies only moderately across the tested configurations. Different threshold choices mainly shift the trade-off between property alignment and preservation of pretrained IF performance, and no single configuration consistently dominates across all metrics. $(\tau_\Delta, \tau_e) = (0.1, -0.2)$ achieves strong structural preservation while maintaining effective solubility alignment. The relatively small variation across neighboring configurations further indicates that **Euterpe** is not highly sensitive to these thresholds.

Table 6: Hyperparameter settings of **Euterpe** for multi-property alignment.

Hyperparameter	**Sol+EP**	**Thermo+EP**
β	0.5	0.5
Objective configuration		
Target-property weight w_{tar}	2.0	3.0
EP weight w_{EP}	1.0	1.0
Margin coefficient λ_{margin}	1.0	1.0
Target reward scale c_{tar}	1.0	2.0
EP reward scale c_{EP}	0.2	0.3
Pair-level reweighting		
Target η	0.95	0.95
EP η	0.95	0.95
Target λ	1.0	1.0
EP λ	2.0	2.0
τ_Δ	0.0	0.0
τ_e	0.0	0.0
Target δ	0.05	0.02
EP δ	0.5	0.5
Token-level reweighting		
K_w	3	3
K_l	3	3
α	0.7	0.7
ϵ	0.1	0.1
γ	0.2	0.2

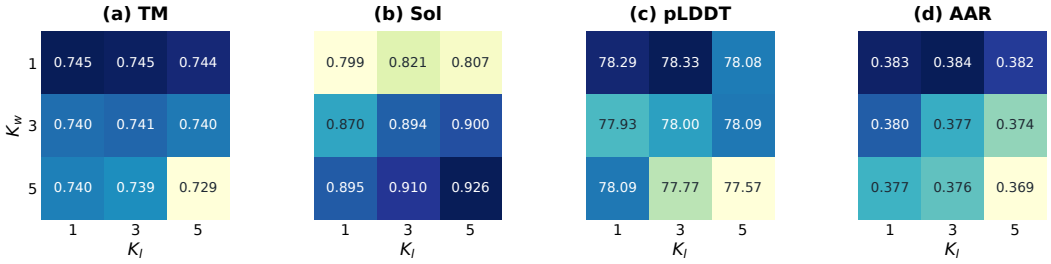


Figure 6: Sensitivity to the token-level Top-K configuration.

We additionally examine the sensitivity to η, which controls the attenuation of preference pairs that conflict with the pretrained reference. As shown in Fig. 8, decreasing η gradually shifts the trade-off from property alignment toward preservation of pretrained IF performance. Performance remains stable across the tested range. $\eta = 0.95$ lies between the two more extreme settings and provides a balanced trade-off between inverse-folding performance and solubility alignment.

C.2 Cross-Benchmark Evaluation on CASP15 TS45

To assess cross-benchmark generalization beyond CATH4.3, we further evaluate on CASP15 TS45 (Gao et al., 2023), which contains 45 target structures derived from CASP15 rather than the CATH benchmark. We directly apply the checkpoints and hyperparameter configurations used in the CATH4.3 experiments, without additional training or tuning on TS45.

As shown in Tab. 7, **Euterpe** achieves an overall property–designability trade-off comparable to MoMPNN [Sol+TM] on TS45, without using folding-derived structural feedback during post-training. The two methods achieve nearly identical average solubility scores (0.8537 vs. 0.8555). Notably, **Euterpe** obtains higher solubility on 29 of the 45 targets. The per-target distributions in Fig. 9 further show that the two methods are closely matched across most targets; the slightly

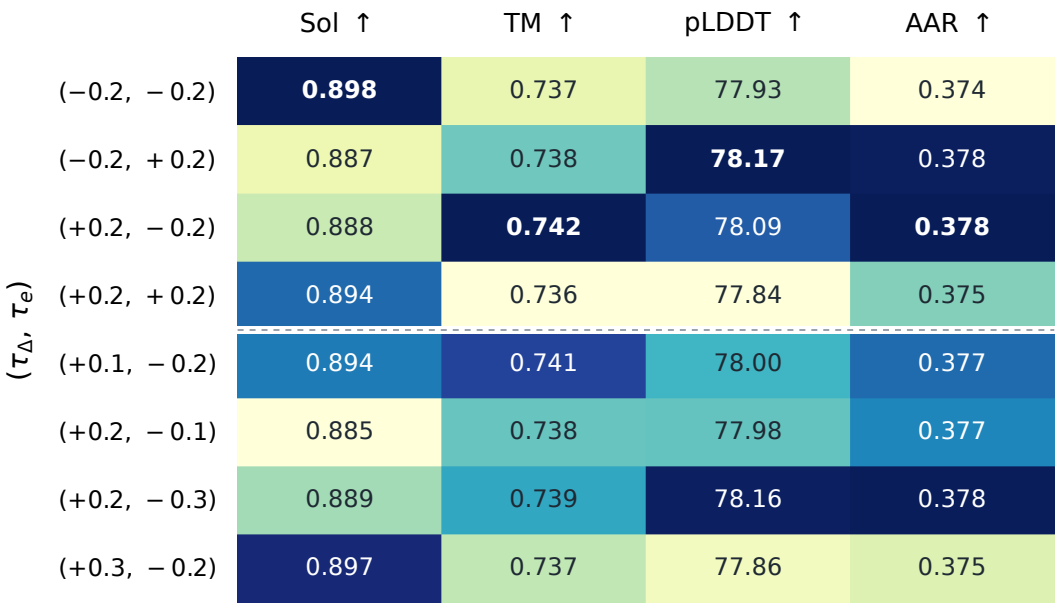


Figure 7: Sensitivity to the pair-level thresholds τ_Δ and τ_e.

Metric	$\eta = 1.00$	$\eta = 0.95$	$\eta = 0.90$
TM ↑	0.7386 ± 0.0020	0.7414 ± 0.0022	0.7416 ± 0.0015
pLDDT ↑	77.8890 ± 0.1233	78.0430 ± 0.1001	78.0439 ± 0.0912
Sol ↑	0.9035 ± 0.0013	0.8940 ± 0.0009	0.8864 ± 0.0009
AAR ↑	0.3760 ± 0.0004	0.3766 ± 0.0007	0.3772 ± 0.0006

Figure 8: Sensitivity to the pair-level attenuation coefficient η.

lower overall mean of **Euterpe** is mainly due to larger negative gaps on a smaller subset of targets rather than systematic underperformance. Together with the CATH4.3 results, these results provide additional evidence for the cross-benchmark generalization of **Euterpe**.

C.3 Additional Evaluation with an Independent Predictor

Following prior work(Liu et al., 2026), our primary evaluation adopts the same scoring function used to provide the optimization signal during training. This protocol enables direct comparison with existing methods under a consistent evaluation setting. However, it also introduces a potential concern: the predictor itself is fully frozen and is not updated during training, but its output is used as the optimization signal, and the same fixed predictor is subsequently used for evaluation. As a result, improvements under this metric may partly reflect adaptation to the specific scoring function of the frozen predictor rather than improvements that transfer to an independent evaluator.

To address this concern, we conducted an additional evaluation using SoluProt Hon et al. (2021), an independently developed predictor that was not used during training, model selection, or checkpoint optimization. We did not retrain or fine-tune any model for this experiment. Instead, we used the previously trained checkpoints to generate sequences and evaluated these sequences directly with SoluProt.

As shown in Table 8, the improvement observed under the original evaluation protocol remains evident when the generated sequences are assessed by an independent predictor. The consistency of these trends under an evaluator that was not involved in training or model selection reduces the likelihood that the observed gains are solely attributable to optimization against the original scorer.

Table 7: **Cross-benchmark evaluation on CASP15 TS45.** Results are averaged over the 45 CASP15 targets. "Sol wins" reports the number of targets on which a method achieves a higher solubility score than the other aligned method.

Method	TM ↑	pLDDT ↑	RMSD < 2 Å ↑	RMSD < 3 Å ↑	Sol ↑	Sol wins ↑	EP ↓	AAR ↑
ProteinMPNN	0.5870	71.3983	0.2889	0.3778	0.6372	–	7.3070	0.3700
MoMPNN [Sol+TM]	0.5875	**73.0787**	0.2815	**0.3926**	**0.8555**	16/45	7.2353	0.3578
Euterpe [Sol]	**0.5904**	71.5752	**0.3111**	0.3704	0.8537	**29/45**	**7.0611**	0.3591

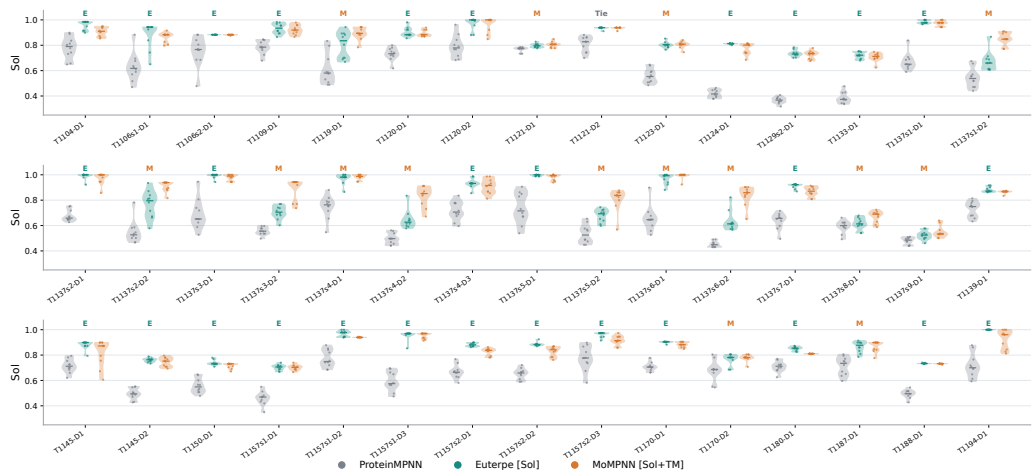


Figure 9: **Per-target solubility distributions on CASP15 TS45.** For each target structure, we independently sample 10 sequences from each method and visualize their predicted Sol scores. E and M indicate a higher mean score for Euterpe and MoMPNN, respectively.

Instead, the results provide additional evidence that the learned sequence distribution captures properties that transfer across independently developed predictive models.

Table 8: Additional evaluation using an independent predictor (SoluProt). Higher scores indicate higher predicted solubility. The column ≥ 0.5 reports the percentage of generated sequences with a predictor score of at least 0.5.

Model	**Mean** ↑	**Std.**	**Median** ↑	≥ 0.5 ↑
ProteinMPNN	0.6220	0.197	0.638	73.7%
MoMPNN [Sol+TM]	0.7704	0.134	0.787	95.5%
Euterpe [Sol]	**0.7814**	**0.123**	**0.792**	**97.0%**