EVALUATOR-IN-THE-LOOP MONTE CARLO TREE SEARCH VIA LLM AGENTS FOR MOTIF SCAFFOLDING IN PROTEIN DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Motif-scaffolding systems commonly follow a generate-then-filter paradigm, in which candidate proteins are generated independently and structural evaluation is used primarily for terminal screening or ranking. This paradigm underuses evaluation: failed predictions contain state-specific evidence about whether a design requires repair of motif geometry, global foldability, or other structural constraints. We introduce **ELMS** (Evidence-based LLM-guided Monte Carlo Search), an evaluator-in-the-loop search framework for motif scaffolding that turns such evaluator feedback into targeted design actions. Effective reuse of structural feedback is nontrivial because different scaffold states exhibit different failure modes, and repeatedly refining a single trajectory can prematurely commit computation to an unproductive region of sequence space. ELMS therefore retains evaluated scaffolds as persistent search states: a Critic Agent diagnoses state-local structural failures, a Policy Agent selects targeted operators with execution parameters, motif-locked operators realize legal sequence modifications, and MCTS determines which historical states should receive further design effort. Under the standard GeomMotif protocol (100 candidates per task), ELMS achieves Successful rates of 86.41% on single-motif tasks and 84.57% on paired-motif tasks, exceeding the strongest prior baseline by 19.3 and 21.9 percentage points, respectively. On MotifBench, under a matched 100-candidate search budget, it solves 26.7 of 30 tasks on average (88.89% Task Success), compared with 16.0 tasks (53.33%) for the strongest baseline. These results establish ELMS as an effective approach for converting structural evaluation from a terminal filter into actionable guidance for iterative motif scaffolding.

1 INTRODUCTION

Motif scaffolding connects a local hypothesis of molecular function to a physically realizable protein. A small set of residues may specify catalysis, binding, or coordination through a precise three-dimensional arrangement, as demonstrated in computational enzyme and epitope scaffolding (Correia et al., 2014). The surrounding protein, however, must stabilize this local geometry while forming a globally foldable and structurally coherent scaffold, motivating structure-conditioned scaffold design approaches such as constrained hallucination and inpainting (Wang et al., 2022). More recently, diffusion-based protein design has enabled direct generation of diverse scaffolds around prescribed structural motifs (Watson et al., 2023; Butcher et al., 2025), with subsequent methods extending this paradigm toward more flexible and atomically specified motif constraints (Ahern et al., 2026). Despite these advances, motif scaffolding still couples several objectives that need not improve together: preserving the prescribed motif geometry, maintaining global foldability, and producing structures sufficiently distinct from known or reference proteins. Reliably balancing these local and global requirements therefore remains a central challenge in functional protein design.

Most modern methods address motif scaffolding through a *generate-then-filter* paradigm. A motif-conditioned generator first produces a population of candidate scaffolds, after which structure prediction and benchmark evaluation are used to accept, reject, or rank the resulting designs. Diffusion-based approaches such as RFdiffusion have made motif-conditioned backbone generation substantially more effective (Watson et al., 2023), while flow-matching methods have provided an alterna-

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.

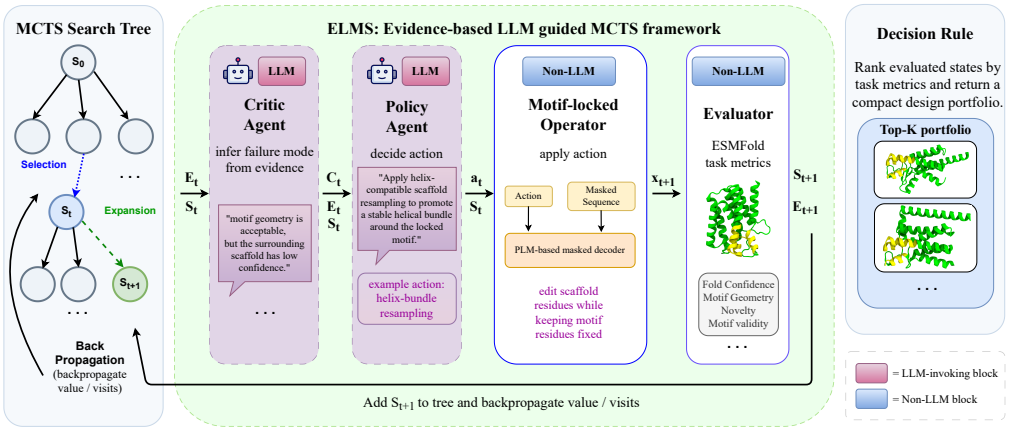


Figure 1: Overview of ELMS. The search tree selects an evaluated state for expansion; the Critic Agent analyzes the state-local evidence to identify which structural or task-specific constraints are limiting the current candidate, the Policy Agent translates this critic into a concrete edit plan over mutable sequence regions, and a motif-locked operator applies the edit while preserving the required motif positions to generate a new candidate sequence. Structural evaluation returns the resulting child to the search tree, where value and visit information is backpropagated to guide subsequent expansions. After search, evaluated states are ranked deterministically for benchmark reporting.

tive framework for sampling conditional protein structures (Yim et al., 2024). Subsequent methods such as Genie 2 Lin et al. (2024) and FADiff Liu et al. (2024) have expanded the range and flexibility of geometric constraints that can be imposed during generation. More recent approaches extend motif-conditioned design toward finer-grained and atomistic specifications (Ahern et al., 2026; Geffner et al., 2026). Despite these advances in generation, structural evaluation is still typically consumed only after candidates have been produced. Low structural confidence and motif displacement indicate different failure modes and therefore suggest different corrective actions, yet under a generate-then-filter interface they may both be treated simply as unsuccessful samples. Consequently, evaluation is primarily used to measure and filter failures rather than to determine how the next design should be changed.

GeomMotif Strashnov et al. (2026) and MotifBench Zheng et al. (2025) evaluate motif-scaffolding methods under benchmark-specified candidate populations, measuring whether generated designs satisfy success criteria and, in some analyses, novelty or diversity. Our question is complementary: when structural evaluation, high-accuracy simulation, or expert review is costly, can evaluator feedback be used during inference to discover at least one feasible scaffold within a fixed candidate/evaluator budget more reliably than generate-then-filter or simpler feedback-guided search?

To address this question, we introduce **E**vidence-based **L**LM-guided **M**onte Carlo **S**earch. ELMS treats evaluated scaffolds as persistent search states: a Critic Agent summarizes state-local structural failures, a Policy Agent selects a legal operator with execution parameters, motif-locked operators realize the edit, and MCTS decides which states receive further design effort. This turns structural evaluation from a terminal filter into an inference-time signal for allocating a limited design budget.

Our main contributions are:

- We formulate motif scaffolding as ELMS, which conducts Monte Carlo Tree Search with an evaluator in the loop. We therefore explicitly separate internal candidate-generation/evaluation budget from the final benchmark reporting interface.
- We introduce a multi-agent MCTS controller in which both LLM Critic and Policy Agents operate over state-local structural evidence, while motif-locked operators enforce constraints.
- We evaluate the framework on two complementary motif-scaffolding benchmarks. Under the standard GeomMotif Success protocol, it achieves Successful rates of 86.41% on single-motif tasks and 84.57% on paired-motif tasks. In a controlled MotifBench setting

with 100 evaluated candidates per case, our framework solves 26.7 of 30 tasks on average, corresponding to 88.89% Task Success. We further use matched-budget ablations and budget-scaling experiments to isolate the contribution of evaluator-guided tree search and analyze its sensitivity to inference compute.

2 Related Work

Motif-conditioned protein generation. Computational motif scaffolding has progressed from physics-based scaffold search to learned generative models capable of constructing proteins directly around prescribed structural motifs. Classical Rosetta-based approaches placed functional residues into existing or newly constructed scaffolds using energy-based optimization and conformational sampling (Kuhlman et al., 2003; Röthlisberger et al., 2008; Jiang et al., 2008; Correia et al., 2014). Deep-learning approaches subsequently removed the need for predefined scaffold libraries: constrained hallucination and RoseTTAFold inpainting generated structures around functional sites (Wang et al., 2022), while RFdiffusion established diffusion-based conditional backbone generation as a strong motif-scaffolding paradigm (Watson et al., 2023). Later methods expanded the flexibility of motif conditioning through SE(3) flow matching, sequence diffusion, and multi-motif generation, including FrameFlowYim et al. (2024), DPLMWang et al. (2024), Genie 2Lin et al. (2024), and FADiff (Liu et al., 2024). More recently, RFdiffusion2 and La-Proteina have extended motif conditioning toward unindexed and atom-level specifications (Ahern et al., 2026; Geffner et al., 2026). Standardized benchmarks such as MotifBenchZheng et al. (2025) and GeomMotifStrashnov et al. (2026) expose complementary limitations of these models: MotifBench evaluates function-derived catalytic and binding motifs, whereas GeomMotif isolates arbitrary geometric preservation across diverse structural contexts.

Evaluation-guided and agentic protein design. Modern protein-design pipelines commonly combine inverse folding and structure prediction, using ProteinMPNN, AlphaFold, or ESMFold to test whether generated sequences recover the intended structures (Dauparas et al., 2022; Jumper et al., 2021; Lin et al., 2023; Hayes et al., 2025). Most motif-scaffolding pipelines use these evaluations primarily for filtering, ranking, or benchmark scoring. Related work has begun to use external feedback more actively: guidance-based methods such as ProteinGuide steer pretrained protein generators toward desired properties, while agentic systems such as ProtAgents coordinate language models, scientific knowledge, and computational tools during protein-design workflows (Xiong et al., 2026a; Ghafarollahi & Buehler, 2024). These approaches demonstrate the value of incorporating additional evidence into design, but generally treat feedback as a global conditioning signal or tool-level reasoning resource rather than as candidate-specific evidence for deciding how an evaluated scaffold should be edited next.

Search and planning with evaluator feedback. Tree search methods provide a general mechanism for allocating inference-time computation toward promising branches while preserving exploration (Kocsis & Szepesvári, 2006; Browne et al., 2012; Xiong et al., 2026b; Gungordu et al., 2026; Xiong et al., 2026c). In motif scaffolding, fold confidence, motif geometry, and hard validity expose different online failure modes, while novelty and redundancy are most naturally evaluated post hoc on the accepted candidate set. Our work uses ELMS as a decision interface around fixed structural evaluators. Unlike generate-then-filter pipelines, each evaluated scaffold becomes a state whose evidence can guide the next expansion; unlike unconstrained agentic design, LLM outputs are limited to critic summaries and operator selection with execution parameters, while non-LLM modules enforce motif locks, realize child sequences, evaluate structures, and back propagate values through the tree.

3 Preliminaries

3.1 Evaluation of Success and Novelty

We follow the evaluation protocol of GeomMotif (Strashnov et al., 2026) and MotifBench Zheng et al. (2025) to assess whether a generated scaffold both preserves the target motif geometry and departs from the reference fold.

Success. We use benchmark-specific success definitions. In GeomMotif, we use the benchmark's modality-specific success criteria. For our sequence-generated candidates, success requires motif-residue C_α RMSD ≤ 1.0 Å and mean pLDDT ≥ 70. Structure-generator baselines retain the benchmark's native structure-level evaluation protocol, which uses the motif C_α RMSD gate together with a structure-level self-consistency gate rather than pLDDT. In MotifBench, success is defined at the submitted backbone level: each submitted backbone is sequence-designed eight times with ProteinMPNN and refolded with ESMFold, and the backbone is successful if at least one refolded design satisfies both motif RMSD < 1.0 Å over motif-residue N, C_α, and C atoms and backbone self-consistency RMSD < 2.0 Å over backbone C_α atoms. Following MotifBench, no pLDDT or confidence filter is applied to the final backbone-success decision.

Novelty. Success alone does not distinguish genuinely new scaffolds from candidates that simply reconstruct the original reference fold. Therefore, novelty is evaluated among successful candidates by comparing each generated scaffold to the corresponding ground-truth structure using TM-score. We denote the ground-truth reference scaffold for the current task as S_{ref}. A successful candidate is treated as novel when its TM-score to the reference structure is below 0.8:

$$\mathrm{Novel}(S_t) = \mathbb{I}\left[\mathrm{Success}(S_t) = 1 \ \wedge \ \mathrm{TMscore}(S_t, S_{\mathrm{ref}}) \leq 0.8\right].$$

3.2 Action Library and Motif-Locked Operators

We use a fixed LLM-synthesized library of biologically motivated motif-locked operators, distilled from protein-design literature, frozen before evaluation, and reused unchanged across GeomMotif and MotifBench. During search, the Policy Agent selects an operator identifier and execution parameters. Actions are realized through fixed-length masked decoding with a protein language model. Decoding may be stochastic, but modifies only permitted scaffold positions while preserving prescribed motif residue identities and their assigned sequence positions. Folded motif geometry is assessed separately by the structure evaluator. The operator inventory, routing, and decoding details are provided in Appendices C.4, C, and E.

4 Methodology

4.1 Framework Overview

ELMS improves motif scaffolding by using LLM agents to transform state-local structural feedback into targeted intervention decisions, enforcing motif preservation through motif-locked sequence operators, and adaptively allocating search effort toward promising or repairable design trajectories.

We formulate the search as a bounded MCTS problem (Kocsis & Szepesvári, 2006; Browne et al., 2012). Each tree node stores an evaluated motif-constrained scaffold state

$$S_i = (x_i, \Pi_i, \boldsymbol{\ell}_i, p_i, a_i^{\mathrm{in}}, E_i, N_i, W_i, Q_i).$$

Here, i indexes a node in the MCTS tree. $x_i \in \mathcal{A}^{L_i}$ is the protein sequence over amino-acid alphabet $\mathcal{A}$, and L_i is its total sequence length. $\boldsymbol{\ell}_i$ records the scaffold lengths around the motif segments: if a task contains m ordered motif segments, then

$$\boldsymbol{\ell}_i = (\ell_{i,0}, \ell_{i,1}, \ldots, \ell_{i,m})$$

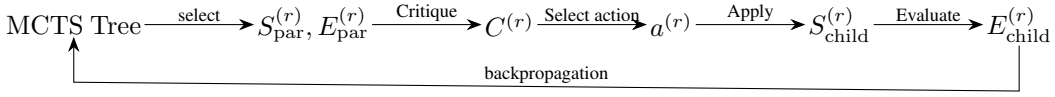
records the length before the first motif segment, the lengths between adjacent motif segments, and the length after the last motif segment. Together with the fixed motif segment lengths, these values determine L_i. Π_i is the motif-to-scaffold placement map, which maps each motif residue to a position in $\{1, \ldots, L_i\}$ while satisfying benchmark placement constraints and preserving motif order. The parent pointer p_i denotes the parent node of S_i, and a_i^{in} records the incoming motif-locked action specification that generated S_i from that parent. E_i is the cached structural evidence obtained when S_i is created, including motif RMSD, pLDDT, hard-validity signals, and online feasibility-proxy margins. (N_i, W_i, Q_i) are the visit count, accumulated value, and mean value used by MCTS.

As shown in Fig. 1, each iteration follows a select–critique–choose-action–apply–evaluate–backpropagate loop. Here, superscript (r) indexes the current expansion iteration and is separate from the persistent node index i. The selected parent state is $S_{\mathrm{par}}^{(r)}$, and the newly generated child

Table 1: Role map for ELMS. The table identifies which roles are dedicated LLM agents and which ones are non-LLM modules or fixed protein-model calls.

Role	Implementation	LLM?	Output
MCTS search loop	Deterministic UCB/back propagate functions	No	Selected tree node, visit counts, back propagate values
Critic Agent	gpt-5-nano with ontology-constrained JSON	Yes	Candidate-local critic labels and rationales
Policy Agent	gpt-5-nano with action-schema-constrained JSON	Yes	Legal operator choices plus execution parameters
Motif-locked operator	Motif-lock template plus sequence-model backend	No	Apply selected actions to legal scaffold regions while preserving motif residues
Structure evaluator	ESMFold plus benchmark-specific adapters	No	Fold confidence, motif RMSD, hard-validity signals, online proxy status
Benchmark reporter	Deterministic ranking procedure	No	Ranked evaluated candidate set

state is $S^{(r)}_{\text{child}}$. The Critic Agent reads the cached evidence $E^{(r)}_{\text{par}}$ associated with $S^{(r)}_{\text{par}}$ and produces a critique $C^{(r)}$. Conditioned on $(S^{(r)}_{\text{par}}, E^{(r)}_{\text{par}}, C^{(r)})$, the Policy Agent returns a legal action specification $a^{(r)}$ from the operator library in Appendix C.4. This specification includes an operator identifier and execution parameters such as search scale, mutable positions, repair fraction, and sampling temperature. The corresponding motif-locked operator applies the action to legal scaffold positions while preserving the prescribed motif residues, producing $S^{(r)}_{\text{child}}$. The child state is then folded and evaluated to obtain $E^{(r)}_{\text{child}}$, and the resulting search utility is backpropagated to update the MCTS statistics.



4.2 Search and Submission Budgets

Let B_{search} denote the internal search budget, defined as the number of candidate states evaluated by the structural prediction and evaluation pipeline. Let K_{sub} denote the number of final designs retained for benchmark reporting. These two quantities correspond to different practical stages: the search procedure may evaluate multiple intermediate states while exploring and repairing design trajectories, whereas a benchmark or downstream workflow may retain only a smaller reported set. We use this distinction to separate internal search effort from the final reporting interface when analyzing budget sensitivity. When the terminal ranking uses the benchmark success indicator, task solving at retained size K_{sub} is interpreted as whether the evaluated pool contains a successful candidate under B_{search}, not as an independent blind Top-K_{sub} selection problem.

4.3 LLM Agents and Non-LLM Modules

Table 1 separates the two LLM components, the Critic and Policy Agents, from non-LLM code and fixed pretrained protein-model inference. Both agents use gpt-5-nano with constrained JSON outputs; all numeric measurements, constraint checks, tree statistics, sequence edits, refolding calls, and benchmark metrics are produced outside the LLM.

This design prevents the LLM from inventing benchmark scores or bypassing motif constraints. The agents produce only constrained critic labels and legal action-schema outputs, including operator identifiers and execution parameters; motif-locked operators then apply the selected edits only to legal scaffold regions while preserving prescribed motif residues; motif-lock and validity checks are enforced deterministically. All numeric measurements and benchmark metrics come from the structure evaluator, not the LLM. The no-leakage interfaces, prompts, and schema validation rules are provided in Appendix F.

4.4 Task Representation and Blinded State Initialization

A benchmark instance is represented as

$$\mathcal{T} = (M, C, L, Ref),$$

where M specifies public motif residues and fixed identities, C contains placement or contig constraints, L is the allowed length range and Ref contains benchmark-provided reference information used only by the evaluator. The framework sees only the public motif mask, public motif identities, generated sequences, predicted structures, and candidate-local structural feedback. For both

GeomMotif and MotifBench, each task is initialized from benchmark-provided motif constraints rather than from a benchmark-origin scaffold sequence. The task adapter reads the required motif specification, including motif residues or segments, target length or allowed length range, motif-placement information, and reference motif coordinates. The initial state S_0 is a virtual motif-only root that stores these sanitized constraints but is not itself structurally evaluated. Root expansion uses legal restart or budgeted sequence-only operators to sample complete scaffold sequences by masking all non-motif scaffold positions under the chosen length and placement constraints. These operators preserve the prescribed motif residues, modify only legal scaffold regions, and provide broad sequence initializations before any evaluated parent exists.

After the first complete candidates are generated, they become the first evaluated tree nodes. All subsequent states are produced by MCTS expansion, motif-locked sequence modification, and fresh structural evaluation under the corresponding benchmark adapter.

4.5 MCTS Selection, Expansion, Evaluation, and Backpropagation

At iteration r, the search procedure descends the current tree using an upper-confidence criterion. For an already evaluated child node j of parent node i with $N_j > 0$, the default selection score is

$$\mathrm{UCB}(i, j) = Q_j + c\sqrt{\frac{\log(N_i + 1)}{N_j + 1}}.$$

Here, N_i is the number of backed-up child evaluations whose selected path contains node i, and N_j is the corresponding count for child node j. Thus, if a batch expansion evaluates multiple children with a shared ancestor, that ancestor receives one visit update per evaluated child. Q_j is defined only after $N_j > 0$ and equals the accumulated backpropagated value divided by N_j. Legal but unevaluated children are expanded before this score is computed; the $+1$ terms stabilize exploration among visited nodes and do not define Q_j for an unvisited child. The exploration constant c controls how strongly the search favors less-visited children. Among visited children, deterministic tie-breaking is performed by depth, hard-validity, and evaluator margin before repeatedly exploiting the current best-valued branch.

Expansion converts an evaluated leaf into one or more legal action specifications. The Critic Agent receives a compact node record containing the sequence length, public motif mask, parent action, current online metrics, evidence records, and visit statistics. It emits ontology labels such as `quality_failed`, `motif_geometry_failed`, `near_threshold`, or `state_successful`. The Policy Agent then maps the verified critic to legal operator choices and execution parameters from the operator library.

Each accepted action is realized by the motif-locked operator backend rather than by the LLM. The backend constructs an editable mask over non-motif residues, preserves motif identities and motif-to-sequence mapping, validates the amino-acid alphabet and length constraints, removes duplicates, and only then admits the child for evaluation. Newly generated sequences are folded with ESMFold (Lin et al., 2023); benchmark adapters compute fold confidence, motif RMSD, online feasibility-proxy margins, and hard-constraint flags. Final benchmark success, including MotifBench backbone-level success, is recomputed separately under the benchmark protocol.

After evaluation, the newly evaluated child node j receives a scalar search utility R_j, computed from its state-local evaluation record as detailed in Appendix D. This value is then backpropagated to every ancestor node i on the selected path:

$$N_i \leftarrow N_i + 1, \qquad W_i \leftarrow W_i + R_j, \qquad Q_i \leftarrow W_i / N_i.$$

The backpropagated value affects future tree selection only. The individual metrics remain available to the Critic Agent, Policy Agent, and terminal selector so that high-confidence but geometrically incorrect candidates are not collapsed into the same explanation as low-confidence but geometry-preserving candidates.

4.6 Termination and Benchmark-Level Reporting

Iteration in Fig. 1 stops when the tree reaches the evaluator budget or when no legal children remain. Benchmark reporting uses only structurally evaluated states. For each task t, let $\mathcal{V}_t$ denote the set

of evaluated states in the MCTS tree. We rank states in $\mathcal{V}_t$ by a fixed deterministic ordering: hard-valid candidates are preferred first, followed by benchmark success, lower motif RMSD and higher fold confidence. When a benchmark requires a retained set, $\mathcal{P}_t^{K_{\text{sub}}} \subseteq \mathcal{V}_t$ denotes the top-K_{sub} states under this ordering; if fewer than K_{sub} evaluated states exist, all evaluated states are retained. An empty evaluated pool or retained set contributes zero to the task-level indicator. The primary task-solving endpoint is

$$\text{TaskSolved}(B_{\text{search}}) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \mathbf{1}\left[\exists i \in \mathcal{V}_t : \; y_i = 1\right],$$

where $y_i \in \{0, 1\}$ is the benchmark-specific success indicator for evaluated state i. Because success-aware terminal ranking places successful candidates before failures, task solving at retained size K_{sub} is equivalent to this pool-level quantity for any $K_{\text{sub}} \geq 1$. Task Success denotes this task-level quantity, whereas y_i remains the candidate-level success indicator.

Benchmark-level comparisons follow the evaluation protocols of the corresponding benchmarks. For GeomMotif, we report the standard Successful rate using the original benchmark success gate. For MotifBench, we use Task Success as a fixed-search-budget endpoint under $B_{\text{search}} = 100$ evaluated candidates per task. Additional protocol details, auxiliary benchmark metrics, and clustering-based analyses are reported in Appendices A and B.

5 Experiments

5.1 Experimental Questions and Reporting Policy

The experiments address two related questions. First, under the standard GeomMotif protocol, can our framework improve benchmark Successful rates while using the original candidate protocol and success definition? Second, under a fixed candidate/evaluator budget, can ELMS discover successful scaffolds more reliably than non-adaptive generation or simpler iterative feedback search? Benchmark splits, candidate interfaces, and baseline-number sources are summarized in Appendix A.

Success is the primary main-text endpoint because it is the fixed benchmark-level feasibility question. During search, ELMS uses only candidate-local evaluator feedback and online proxy signals; final benchmark success is recomputed under each benchmark protocol after candidate generation. Novelty is reported post hoc: it is evaluated after the final candidate set is formed and is therefore treated as a characterization metric rather than primary online objectives.

5.2 GeomMotif Success and Novelty Comparison

Table 2 compares our framework with published GeomMotif baselines using the original GeomMotif Successful definition and candidate protocol (100 candidates per task), while reporting Novelty as a secondary post hoc statistic. Protocol details are provided in Appendix A.1. ELMS reaches 86.41% Successful on single-motif tasks and 84.57% on paired-motif tasks, while also reaching 83.3% Novel on single-motif tasks and 65.7% Novel on paired-motif tasks. These gains substantially exceed the strongest baseline methods on both single- and paired-motif tasks, indicating that evaluator-guided search improves feasibility across GeomMotif constraint settings. We also include a *Baseline LLM* that directly revises the current sequence from candidate-local evaluator feedback, without the action library, branch preservation, UCB selection, or value back propagation.

5.3 MotifBench De Novo Design Comparison

MotifBench contains 30 functional motif-scaffolding tasks. We compare methods under a matched 100-candidate search budget per task. Candidates are ranked deterministically for benchmark reporting; because this ranking uses the benchmark success indicator, Task Success is interpreted as whether the 100-candidate pool contains at least one successful candidate. Additional protocol details are provided in Appendix A.2.

Table 3 reports the fixed-search-budget MotifBench comparison. Across three runs, ELMS solves 26.7 of 30 tasks on average and achieves an 88.89% Success Rate, exceeding the strongest baseline by 10.7 solved tasks and 35.56 percentage points (16.0 tasks, 53.33%). This large gain shows

Table 2: GeomMotif success and novelty comparison. All rows use the standard GeomMotif protocol (100 candidates per task) and benchmark Successful definition. Values are reported as mean ± standard deviation. Post hoc Unique and additional diversity statistics are reported in Appendix B.

Model / system	Successful (%)		Novel (%)	
	Single	Paired	Single	Paired
Genie2	60.1 ± 1.0	32.9 ± 0.4	60.1 ± 1.0	26.6 ± 0.5
La-Proteina	67.1 ± 0.7	62.7 ± 0.5	67.1 ± 0.6	35.2 ± 0.9
RFdiffusion	65.1 ± 0.4	43.7 ± 1.0	65.1 ± 0.4	25.0 ± 0.5
Protpardelle-1C	56.2 ± 0.7	44.6 ± 0.5	56.2 ± 0.7	25.2 ± 0.4
FrameFlow	30.6 ± 0.4	25.1 ± 0.4	30.6 ± 0.4	19.7 ± 0.3
RFdiffusion2	24.9 ± 0.6	13.5 ± 0.5	24.9 ± 0.6	11.4 ± 0.3
ESM3 (seq)	17.4 ± 0.3	6.5 ± 0.5	11.3 ± 0.5	0.1 ± 0.0
DPLM-3B	19.3 ± 0.7	11.0 ± 0.5	10.2 ± 0.5	0.9 ± 0.2
DPLM-650M	15.9 ± 0.4	6.5 ± 0.5	8.7 ± 0.2	0.4 ± 0.1
ESM3 (seq & struct)	2.0 ± 0.1	0.7 ± 0.2	2.0 ± 0.1	0.0 ± 0.0
Baseline LLM (gpt-5-nano)	13.1 ± 0.1	7.2 ± 0.2	3.7 ± 0.0	0.3 ± 0.0
ELMS (gpt-5-nano)	$\mathbf{86.4} \pm 0.8$	$\mathbf{84.6} \pm 0.6$	$\mathbf{83.3} \pm 0.5$	$\mathbf{65.7} \pm 0.6$

Table 3: MotifBench fixed-search-budget comparison. Results are averaged over three independent runs; each run evaluates 100 candidates per task. Solutions denotes the mean number of tasks for which at least one evaluated candidate satisfies the MotifBench success criterion.

Method	Solutions	Success Rate(%)
RFdiffusion3	15.7	52.22 ± 4.84
La-Proteina	16.0	53.33 ± 1.92
Protpardelle-1c (cc91-large)	14.7	48.89 ± 1.11
Genie3	9.0	30.00 ± 1.92
Baseline LLM(GPT-5-nano)	8.3	27.78 ± 2.22
ELMS(GPT-5-nano)	**26.7**	$\mathbf{88.89 \pm 1.11}$

that evaluator-in-the-loop LLM-guided search is effective at using a fixed search budget to discover successful scaffolds.

5.4 Matched-Budget Search Ablation

This ablation asks whether the observed GeomMotif Success improvement comes from ELMS rather than from changing the underlying language model. All variants are evaluated on GeomMotif and use the same LLM model (gpt-5-nano), the same blinded motif inputs, the same legal action space, the same structural evaluator, the same Success gate, and the same total candidate/evaluator budget $B_{\text{search}} = 100$.

We instantiate four matched-budget search variants. *Random Operator Search* samples randomly from the shared legal action library without using evaluator feedback for routing. *Generate-then-filter* independently generates candidates without feeding evaluator results back into action selection. *Greedy Operator Search* uses candidate-local evaluator feedback to select operators, but follows only the current trajectory, with no tree selection or value backpropagation. *ELMS* uses the same feedback and action library while preserving multiple branches, applying UCB selection, and back propagating evaluator-derived values. Unlike Baseline LLM, Greedy Operator Search is therefore a matched-library search framework, differing from ELMS only in its tree-level search machinery rather than its action-selection interface. In Table 4, "Evaluator feedback" indicates whether later action choices can condition on structural evaluator outputs, and "Tree selection / value backpropagation" describes how evaluated feedback is retained and reused.

Table 4: Matched-budget GeomMotif search ablation with identical LLM, inputs, evaluator, success gate, and action space, using $B_{\text{search}} = 100$ evaluated candidates per task. All rows report Task Success: the percentage of tasks with at least one successful candidate.

Variant	Evaluator feedback	Action routing	Tree selection / value backpropagation	Single Success (%)	Paired Success (%)
Random Operator Search	No	Random legal routing	No	0.0	0.0
Generate-then-filter	No	Independent direct generation	No	6.1	10.4
Greedy Operator Search	Yes	Evidence-guided greedy routing	No; single trajectory	48.0	53.7
ELMS	Yes	Evidence-guided action selection	UCB selection + value backpropagation	**88.6**	**86.4**

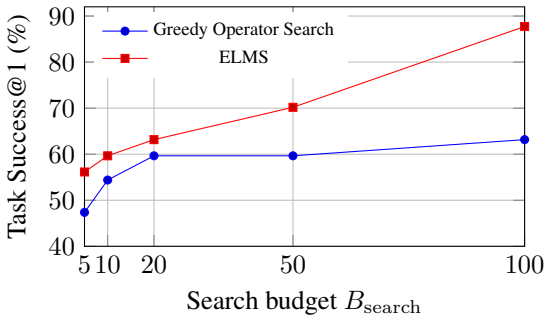


Figure 2: GeomMotif Task Success@1 as a function of internal search budget with $K_{\text{sub}} = 1$. Greedy feedback improves at small budgets but largely plateaus, whereas ELMS continues to benefit from additional search.

5.5 Search-Budget Scaling

We next examine how the two feedback-guided search methods use additional inference compute. We fix $K_{\text{sub}} = 1$ and vary the internal search budget over $B_{\text{search}} \in \{5, 10, 20, 50, 100\}$. Task Success@1 is the fraction of 57 GeomMotif tasks for which the returned design satisfies the Success gate. Greedy Operator Search and ELMS share the same blinded inputs, evaluator, and legal action space, differing only in whether evaluated alternatives are retained and exploited through tree statistics and value backpropagation.

Figure 2 shows different scaling behavior for the two controllers. Greedy Operator Search rises from 47.37% at $B = 5$ to 59.65% at $B = 20$, but reaches only 63.16% at $B = 100$. ELMS rises from 56.14% to 87.72% over the same budget range. In particular, from $B = 20$ to $B = 100$, greedy search improves by only 3.51 percentage points, while MCTS improves by 24.56 points. The widening gap indicates that the benefit of tree search is not simply due to receiving evaluator feedback: preserving evaluated alternatives allows additional inference compute to continue producing useful exploration rather than repeatedly refining a single trajectory.

6 Conclusion

We introduced ELMS for motif scaffolding, which turns structural evaluation into state-specific guidance for iterative protein design. ELMS retains evaluated scaffolds as persistent search states, uses Critic and Policy Agents to interpret structural evidence and select targeted modification strategies, executes these decisions through motif-locked operators, and uses MCTS to adaptively allocate further design effort across promising and repairable states.

Across GeomMotif and MotifBench, the framework consistently outperforms the strongest generative baselines under matched candidate/evaluator budgets. These results highlight ELMS as a promising approach for combining structural evaluation, agentic reasoning, and adaptive search in constrained protein design. Future work may incorporate richer structural and functional evidence, extend the action space beyond sequence-level refinement, and evaluate whether evaluator-guided search can generalize to broader protein-design objectives and experimentally validated design settings.

AI USE STATEMENT

Generative AI tools used for manuscript preparation were used solely to improve the clarity, grammar, and readability of the manuscript. They were not used for research ideation, methodology or experimental design, implementation, or result interpretation. Representative examples of wording-only edits are provided in Appendix G.

REFERENCES

Woody Ahern, Jason Yim, Doug Tischer, Saman Salike, Seth M. Woodbury, Donghyo Kim, Indrek Kalvet, Yakov Kipnis, Brian Coventry, Han Raut Altae-Tran, Magnus S. Bauer, Regina Barzilay, Tommi S. Jaakkola, Rohith Krishna, and David Baker. Atom-level enzyme active site scaffolding using RFdiffusion2. *Nature Methods*, 23:96–105, 2026. doi: 10.1038/s41592-025-02975-x.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

Jasper Butcher, Rohith Krishna, Raktim Mitra, Rafael I. Brent, Yanjing Li, Nathaniel Corley, Paul T. Kim, Jonathan Funk, Simon Mathis, Saman Salike, Aiko Muraishi, Helen Eisenach, Tuscan Rock Thompson, Jie Chen, Yuliya Politanska, Enisha Sehgal, Brian Coventry, Odin Zhang, Bo Qiang, Kieran Didi, Max Kazman, Frank DiMaio, and David Baker. De novo design of all-atom biomolecular interactions with RFdiffusion3. *bioRxiv*, 2025. doi: 10.1101/2025.09.18.676967. Preprint.

Bruno E. Correia, John T. Bates, Rebecca J. Loomis, et al. Proof of principle for epitope-focused vaccine design. *Nature*, 507:201–206, 2014. doi: 10.1038/nature12966.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J. Ragotte, Lukas F. Milles, Basile I. M. 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. doi: 10.1126/science.add2187.

Tomas Geffner, Kieran Didi, Zhonglin Cao, Danny Reidenbach, Zuobai Zhang, Christian Dallago, Emine Kucukbenli, Karsten Kreis, and Arash Vahdat. La-Proteina: Atomistic protein generation via partially latent flow matching. In *International Conference on Learning Representations*, 2026.

Alireza Ghafarollahi and Markus J. Buehler. ProtAgents: Protein discovery via large language model multi-agent collaborations combining physics and machine learning. *Digital Discovery*, 3 (7):1389–1409, 2024. doi: 10.1039/D4DD00013G.

Oguzhan Gungordu, Siheng Xiong, and Faramarz Fekri. Pathwise: Planning through world model for automated heuristic design via self-evolving llms. In *Proceedings of the Forty-third International Conference on Machine Learning*. PMLR, 2026. URL https://arxiv.org/abs/2601.20539.

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf A. Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, 2025. doi: 10.1126/science.ads0018.

Lin Jiang, Eric A. Althoff, Fernando R. Clemente, Lindsey Doyle, Daniela Röthlisberger, Alexandre Zanghellini, Jasmine L. Gallaher, Jamie L. Betker, Fujie Tanaka, III Barbas, Carlos F., Donald Hilvert, Kendall N. Houk, Barry L. Stoddard, and David Baker. De novo computational design of retro-aldol enzymes. *Science*, 319(5868):1387–1391, 2008. doi: 10.1126/science.1152692.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zidek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pp. 282–293. Springer, 2006. doi: 10.1007/11871842_29.

Brian Kuhlman, Gautam Dantas, Gregory C. Ireton, Gabriele Varani, Barry L. Stoddard, and David Baker. Design of a novel globular protein fold with atomic-level accuracy. *Science*, 302(5649): 1364–1368, 2003. doi: 10.1126/science.1089427.

Yeqing Lin, Minji Lee, Zhao Zhang, and Mohammed AlQuraishi. Out of many, one: Designing and scaffolding proteins at the scale of the structural universe with Genie 2. *arXiv preprint arXiv:2405.15489*, 2024.

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

Ke Liu, Weian Mao, Shuaike Shen, Xiaoran Jiao, Zheng Sun, Hao Chen, and Chunhua Shen. Floating anchor diffusion model for multi-motif scaffolding. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 31691–31708. PMLR, 2024.

Daniela Röthlisberger, Olga Khersonsky, Andrew M. Wollacott, Lin Jiang, Jason DeChancie, Jamie Betker, Jasmine L. Gallaher, Eric A. Althoff, Alexandre Zanghellini, Orly Dym, Shira Albeck, Kendall N. Houk, Dan S. Tawfik, and David Baker. Kemp elimination catalysts by computational enzyme design. *Nature*, 453(7192):190–195, 2008. doi: 10.1038/nature06879.

Pavel Strashnov, Andrey Shevtsov, Viacheslav Meshchaninov, Olga Kardymon, and Dmitry P. Vetrov. GeomMotif: A benchmark for arbitrary geometric preservation in protein generation. In *International Conference on Learning Representations*, 2026.

Jue Wang, Sidney Lisanza, David Juergens, Doug Tischer, Joseph L. Watson, Karla M. Castro, Robert Ragotte, Amijai Saragovi, Lukas F. Milles, Minkyung Baek, et al. Scaffolding protein functional sites using deep learning. *Science*, 377(6604):387–394, 2022. doi: 10.1126/science.abn2100.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 52309–52333. PMLR, 2024.

Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, et al. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, 2023. doi: 10.1038/s41586-023-06415-8.

Junhao Xiong, Ishan Gaur, Maria Lukarska, Hunter Nisonoff, Luke M. Oltrogge, David F. Savage, and Jennifer Listgarten. Property guidance for protein sequence generative models with ProteinGuide. *Nature Biotechnology*, 2026a. doi: 10.1038/s41587-026-03207-z.

Siheng Xiong, Oguzhan Gungordu, James C. Kerce, and Faramarz Fekri. Adaptive information control for search-augmented LLM reasoning. *arXiv preprint arXiv:2602.01672*, 2026b. URL https://arxiv.org/abs/2602.01672.

Siheng Xiong, Ali Payani, and Faramarz Fekri. Enhancing language model reasoning with structured multi-level modeling. In *The 14th International Conference on Learning Representations*, 2026c. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/3db7d123a316fc690f02818b21967af4-Abstract-Conference.html.

Jason Yim, Andrew Campbell, Emile Mathieu, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Frank Noé, Regina Barzilay, and Tommi S. Jaakkola. Improved motif-scaffolding with SE(3) flow matching. *Transactions on Machine Learning Research*, 2024.

Zhuoqi Zheng, Bo Zhang, Kieran Didi, Kevin K. Yang, Jason Yim, Joseph L. Watson, Hai-Feng Chen, and Brian L. Trippe. MotifBench: A standardized protein design benchmark for motif-scaffolding problems. *arXiv preprint arXiv:2502.12479*, 2025.

A Benchmark Protocols and Task Splits

This appendix summarizes the benchmark interfaces used in the experiments, including task splits, candidate budgets, input fields, and the source of baseline numbers. We distinguish benchmark evaluation from online MCTS search: the controller receives only public task information and generated-candidate feedback, whereas hidden references and official post hoc statistics are used only after candidate generation.

A.1 GeomMotif Protocol

GeomMotif contains 57 motif-scaffolding tasks, divided into 35 single-motif tasks and 22 paired-motif tasks. We use the official single/paired split throughout the paper.

Task inputs. For each GeomMotif task, the public search input contains the target length or length range, the public motif mask, fixed motif amino-acid identities, motif placement constraints, and the distinction between fixed motif positions and editable scaffold positions. The controller may also use generated candidate sequences, predicted structures, and sanitized evaluator feedback from those generated candidates. The controller does not receive origin sequences, hidden native scaffold residues, homologs, templates, full reference structures, or native-reference novelty scores during candidate generation.

Success criterion. A generated candidate is counted as a GeomMotif Success using the benchmark's modality-specific official thresholds. For the sequence-generated Baseline LLM and ELMS candidates, success requires motif-residue C_α RMSD $\leq$ 1.0 Å and mean pLDDT $\geq$ 70. Published structure-generator baselines retain the benchmark's native structure-level protocol, which uses the same motif C_α RMSD gate together with a structure self-consistency RMSD gate rather than pLDDT. Novelty and Unique are computed only after generation using the official GeomMotif post hoc protocol. These hidden-reference or population-level metrics are not returned to the online MCTS reward.

Candidate interface. Table 2 uses the standard GeomMotif protocol (100 candidates per task) and benchmark Successful definition for all rows. Published generator rows are taken from the GeomMotif benchmark under its reported candidate interface. The Baseline LLM and ELMS rows are our blinded local runs on the same 57-task split, reported as GeomMotif Successful percentages under the benchmark candidate protocol.

Table 5: GeomMotif benchmark protocol used in this work.

Item	Protocol
Number of tasks	57 total
Single split	35 tasks
Paired split	22 tasks
Online search input	Public motif mask, fixed motif residues, length/placement constraints
Withheld from search	Origin sequence, native scaffold, homologs, templates, native TM-score
Success gate	Modality-specific: sequence candidates use motif C_α RMSD $\leq$ 1.0 Å and pLDDT $\geq$ 70; structure baselines use motif C_α RMSD and scRMSD gates
Candidate budget	100 candidates/task
Published baselines	Published GeomMotif benchmark results
Baseline LLM	Our blinded Baseline LLM run with gpt-5-nano
ELMS	Our blinded ELMS run with gpt-5-nano
Novelty/Unique	Post hoc only, computed after candidate generation

Baseline number sources. The GeomMotif rows for Genie2, La-Proteina, RFdiffusion, Protpardelle-1C, FrameFlow, RFdiffusion2, ESM3, and DPLM are taken from the published GeomMotif benchmark results and are used as descriptive benchmark context. They are not rerun in our compute environment for the main GeomMotif success table. The Baseline LLM (gpt-5-nano) row is our local LLM-only baseline. The ELMS (gpt-5-nano) row is our blinded ELMS run on the same 57-task split.

Table 6: Source of GeomMotif comparison numbers.

Row type	Source	Candidate interface
Published generators	GeomMotif benchmark table	Standard GeomMotif protocol (100 candidates/task)
Baseline LLM (gpt-5-nano)	Our local baseline run	Standard GeomMotif protocol (100 candidates/task)
ELMS (gpt-5-nano)	Our blinded ELMS run	Standard GeomMotif protocol (100 candidates/task)

A.2 MotifBench Protocol

MotifBench contains 30 functional motif-scaffolding tasks. We use MotifBench as a transfer benchmark for the same motif-locked decision framework. The main text does not report the full MotifBench leaderboard score; instead, it compares methods under a fixed-search-budget task-solving setting with

$$B_{\text{search}} = 100$$

evaluated candidates per task.

Task inputs. For each MotifBench task, the adapter provides the target motif definition, motif residue identities, motif coordinates, target length or placement constraints, allowed redesignable scaffold regions, motif segment order, and linker constraints. During search, the controller sees only the public motif-constrained design problem and feedback from generated candidates. It does not use the hidden native scaffold sequence as a candidate source.

Fixed-search-budget candidate protocol. MotifBench final success is a backbone-level decision. For each submitted or evaluated backbone, the protocol designs eight sequences with ProteinMPNN and refolds them with ESMFold. The backbone is successful only if the same refolded design simultaneously satisfies motif RMSD < 1.0 Å over motif-residue N, C_α, and C atoms and backbone self-consistency RMSD < 2.0 Å over backbone C_α atoms. Following MotifBench, no pLDDT or confidence filter is applied to the final backbone-success decision.

For every method and every MotifBench task, each independent run first generates and evaluates 100 candidates. Task Success is defined as the fraction of the 30 tasks for which at least one evaluated candidate is successful:

$$\text{SuccessRate}(B_{\text{search}} = 100) = \frac{1}{30} \sum_{t=1}^{30} \mathbb{I}\left[\exists\, S \in \mathcal{C}_t^{100} : \text{Success}_{\text{MB}}(S) = 1\right],$$

where $\mathcal{C}_t^{100}$ is the 100-candidate evaluated pool for task t in one run. Solutions is the corresponding number of solved tasks out of 30. The main text reports the mean over three independent runs, with Success Rate shown as mean $\pm$ standard error of the mean. Candidates may be ranked deterministically for reporting, but because the ranking uses the benchmark success indicator, the reported task-solving endpoint is whether the evaluated pool contains at least one successful candidate.

Baseline number sources. The RFdiffusion3, La-Proteina, Protpardelle-1c (cc91-large), and Genie3 MotifBench rows are derived from public MotifBench evaluation packages or released candidate pools, converted to the same 100-candidate fixed-search-budget task-solving endpoint. These are therefore not quoted full-candidate leaderboard scores. The Baseline LLM and ELMS rows are produced by blinded gpt-5-nano runs under the same 100-candidate interface.

Why full MotifBench Score is not the main comparison. The official MotifBench score is designed for a larger submission interface and depends on a full candidate population, clustering, and

Table 7: MotifBench fixed-search-budget protocol used in this work.

Item	Protocol
Number of tasks	30 functional motif-scaffolding tasks
Evaluated pool	100 candidates/task
Reporting interface	Deterministic ranking of evaluated candidates
Task Success	At least one successful candidate among 100 evaluated candidates
Solutions	Number of solved tasks out of 30
Replicates	Three independent runs
Reported main-text endpoint	Solutions and Success Rate at $B_{\text{search}} = 100$
Full MotifBench Score	Not reported as the main-text comparison

Table 8: Source of MotifBench fixed-search-budget comparison numbers.

Method	Source	Runs	Reported endpoint
RFdiffusion3	Public MotifBench package	3	Task Success at $B_{\text{search}} = 100$
La-Proteina	Public MotifBench package	3	Task Success at $B_{\text{search}} = 100$
Protpardelle-1c (cc91-large)	Public MotifBench package	3	Task Success at $B_{\text{search}} = 100$
Genie3	Public MotifBench package	3	Task Success at $B_{\text{search}} = 100$
Baseline LLM (gpt-5-nano)	Our blinded baseline run	3	Task Success at $B_{\text{search}} = 100$
ELMS (gpt-5-nano)	Our blinded ELMS run	3	Task Success at $B_{\text{search}} = 100$

bootstrap sampling. In contrast, our main MotifBench experiment asks a fixed-search-budget task-solving question: after evaluating 100 candidates, can a method find at least one successful scaffold? For this reason, the main text reports Solutions and Task Success rather than the aggregate MotifBench Score. Full-score-style analyses are treated as auxiliary diagnostics and are not mixed with the fixed-search-budget comparison.

B Post Hoc Novelty and Diversity Reporting

Main-text results emphasize Success because it is the online candidate-level gate used by the search controller and can be evaluated consistently during MCTS. Novelty and Unique are computed only after the final candidate set is frozen and are therefore reported here as post hoc characterization of the evaluated candidate pool rather than as primary online optimization targets.

The lower Unique scores of ELMS should be interpreted as reduced structural cluster breadth rather than reduced feasibility. The controller concentrates its search budget around high-value feasible basins and therefore trades some population-level diversity for higher Success and Novelty under the fixed-search budget setting studied in the main text. Diversity-aware tree policies or post hoc diversity-aware filtering are natural extensions when broad topology discovery is the primary objective.

C Operator Inventory and Routing

This appendix documents the operator inventory used by ELMS. The same framework is used across the GeomMotif and MotifBench experiments; benchmark-specific adapters only change task inputs and evaluation protocols. The operators are treated as immutable, generic protein-design priors. They do not expose origin sequences, hidden native structures, target PDB identities, homologs, templates, MSAs, or benchmark-specific retrieved fragments. Online search may use only public motif masks, generated candidate sequences, candidate-local diagnostics, and sanitized evaluator feedback. The operator inventory is described below at the level of search policy, evidence boundaries, and action realization rather than source-code organization.

Throughout this appendix, a *operator identifier* denotes a named entry in the adaptive operator inventory. These identifiers are labels used for routing, accounting, and reward backpropagation; they are not file names or function names.

Table 9: Post hoc GeomMotif success, novelty, and uniqueness comparison. All rows use the standard GeomMotif candidate protocol and benchmark Successful definition. Values are reported as mean ± standard error of the mean where available. Novel and Unique are reported as final evaluator statistics and are not used as online MCTS objectives.

Model / system	Successful (%)		Novel (%)		Unique (%)	
	Single	Paired	Single	Paired	Single	Paired
Genie2	60.1 ± 1.0	32.9 ± 0.4	60.1 ± 1.0	26.6 ± 0.5	59.9 ± 1.0	22.5 ± 0.3
La-Proteina	67.1 ± 0.7	62.7 ± 0.5	67.1 ± 0.6	35.2 ± 0.9	61.3 ± 0.6	22.7 ± 0.4
RFdiffusion	65.1 ± 0.4	43.7 ± 1.0	65.1 ± 0.4	25.0 ± 0.5	62.4 ± 0.4	20.5 ± 0.7
Protpardelle-1C	56.2 ± 0.7	44.6 ± 0.5	56.2 ± 0.7	25.2 ± 0.4	53.5 ± 0.7	22.6 ± 0.3
FrameFlow	30.6 ± 0.4	25.1 ± 0.4	30.6 ± 0.4	19.7 ± 0.3	30.6 ± 0.4	20.2 ± 0.5
RFdiffusion2	24.9 ± 0.6	13.5 ± 0.5	24.9 ± 0.6	11.4 ± 0.3	24.9 ± 0.6	12.7 ± 0.4
ESM3 (seq)	17.4 ± 0.3	6.5 ± 0.5	11.3 ± 0.5	0.1 ± 0.0	10.1 ± 0.2	0.1 ± 0.0
DPLM-3B	19.3 ± 0.7	11.0 ± 0.5	10.2 ± 0.5	0.9 ± 0.2	9.8 ± 0.5	0.6 ± 0.1
DPLM-650M	15.9 ± 0.4	6.5 ± 0.5	8.7 ± 0.2	0.4 ± 0.1	8.1 ± 0.3	0.3 ± 0.1
ESM3 (seq & struct)	2.0 ± 0.1	0.7 ± 0.2	2.0 ± 0.1	0.0 ± 0.0	2.0 ± 0.1	0.0 ± 0.0
Baseline LLM (gpt-5-nano)	13.06 ± 0.1	7.21 ± 0.2	3.7 ± 0.0	0.3 ± 0.0	8.1 ± 0.2	0.8 ± 0.1
ELMS (gpt-5-nano)	$\mathbf{86.4} \pm 0.8$	$\mathbf{84.6} \pm 0.6$	$\mathbf{83.3} \pm 0.5$	$\mathbf{65.7} \pm 0.6$	13.0 ± 0.2	8.0 ± 0.3

C.1 Operator Boundary and No-Leakage Contract

The operator inventory defines the evidence boundary for online search. Allowed online inputs are: anonymous task id, target length, and public masked motif; mask-derived features such as fixed-residue fraction, fixed-span count, gap lengths, and public motif composition; generated candidates produced inside the blinded run; candidate-local pLDDT, motif RMSD, online feasibility flags, low-confidence spans, and generated-only archive diagnostics; and generic literature-derived operators that are not queried with benchmark motifs or candidate sequences. Forbidden online inputs are: origin/reference/initial scaffold sequence; native PDB, motif coordinates, residue mappings, native DSSP, native contacts, native topology labels, or hidden structure-derived residue errors; BLAST, Foldseek, UniProt, PDB, MSA, or web retrieval using the motif, task id, or generated sequence; and any attempt to infer the hidden origin sequence from evaluator feedback.

The online feasibility proxy is fixed throughout search:

$$\text{pLDDT} \geq 70, \qquad \text{motif RMSD} \leq 1.0 \text{ Å}.$$

For MotifBench this proxy is used only for search control; final backbone success is recomputed with the MotifBench ProteinMPNN–ESMFold protocol. Official Novelty and Unique metrics are computed only after the final candidate set is frozen. They are not used to reveal hidden native information during action-guided candidate generation.

C.2 Operator Use During One MCTS Expansion

At each expansion step, the tree controller first selects a candidate node by UCB/value statistics. The operator router then reads sanitized feedback and the public mask, assigns a metric regime, and ranks eligible adaptive operators by

$$\text{score}(o) = \pi_{\text{regime}}(o) + 0.3Q_o + c\sqrt{\frac{\log(N_{\text{all}} + |\mathcal{O}| + 1)}{N_o + 1}}.$$

Here, o is a candidate operator identifier and $\mathcal{O}$ is the current legal operator menu. The term $\pi_{\text{regime}}(o)$, corresponding to the metric prior, is a regime-dependent prior score assigned before task-specific operator outcomes are observed; for example, local trust-region operators are favored when motif RMSD already passes but pLDDT is low, whereas restart/topology operators are favored when pLDDT is high but motif RMSD is far from the threshold. The term Q_o is the empirical mean reward of operator o on the current task. Let W_o be the cumulative reward backed up from evaluated

Table 10: Metric regimes used by the no-leakage operator router.

Regime	Trigger	Default operator behavior
`virtual_root_unmeasured`	no evaluated parent	complete motif-locked roots; budgeted sampling; model/topology diversification
`hard_success`	pLDDT and RMSD both pass	proxy-preserving local exploration; conservative support edits
`confidence_limited`	RMSD passes, pLDDT < 70	preserve the good geometry basin; use small edits or candidate-backbone inverse folding
`geometry_near`	pLDDT ≥ 70 and $1 <$ RMSD ≤ 3	medium partial re-noising or feedback-guided recombination; detect pLDDT-only plateaus
`geometry_far`	pLDDT ≥ 70 and RMSD > 3	treat as stable wrong topology; restart across topology/model arms
`dual_limited`	both metrics are poor	broaden model/topology family before local optimization

children generated under operator o. We use zero initialization:

$$Q_o = \begin{cases} 0, & N_o = 0, \\ W_o/N_o, & N_o > 0. \end{cases}$$

Only generated candidates that were folded and scored under the blinded evaluator contribute to W_o and N_o. The coefficient 0.3 downweights this empirical term so that early noisy outcomes do not dominate the regime prior and exploration term. The term N_o is the number of evaluated candidates previously generated under operator o. The term N_{all} is the total number of operator-labeled evaluations performed so far in the current task, and $|\mathcal{O}|$ is the number of legal candidate operators in the current menu; the $+1$ offsets avoid undefined values in the logarithm and exploration denominator at the beginning of search. Finally, c controls the strength of the UCB-style exploration bonus. Thus an operator can be selected because it is appropriate for the current metric regime, because it has produced high measured reward, or because it remains under-explored.

Retired operators receive a large negative score and are removed from consideration for the affected branch. The router normally returns the top one or two non-retired adaptive operator identifiers together with ranked static arms. The policy LLM then chooses an action specification within this legal menu. It receives only the anonymous public mask, generated-candidate summaries, allowed operators, allowed arms, legal parent ids, and legal repair positions. It returns an action specification containing the failure class, selected operator identifiers, search scale, arms, repair positions, repair fraction, temperature, and restart flag. The policy LLM does not write protein sequences; it chooses how to search.

The motif-lock backend realizes the decision as motif-locked children. Restart-scale actions perform full-mask sequence generation with all public motif residues fixed. Medium-scale actions perform parent-conditioned partial re-noising or multi-site recombination. Local actions perform single-site or small conservative scaffold-only edits. Candidate-backbone inverse folding uses the candidate's own predicted structure while fixing public motif positions. Every child is freshly validated and folded: motif edits, noncanonical residues, invalid length/linker arithmetic, duplicates, and previously evaluated sequences are rejected, and the evaluator recomputes pLDDT and motif RMSD. Candidate rewards update the MCTS path and the corresponding operator statistics, allowing task-specific operator selection without changing the operator specification.

Branch diagnostics may also retire operators. If local single-site probes change pLDDT without moving RMSD, the branch records a confidence-geometry plateau and retires local/additive operators for that parent. If medium children are all dominated by the parent, the corresponding parent-generator pair is retired.

C.3 Metric Regimes and Default Routing Priors

The metric regime determines the prior term $\pi_{\text{regime}}(o)$ in the operator score. These priors are not learned from hidden references; they encode which classes of blinded search actions are sensible under the current pLDDT and motif-RMSD pattern. The concrete operator identifiers that implement these behaviors are listed in Appendix C.4.

C.4 Adaptive Operator Inventory

The framework uses 19 adaptive operator identifiers. An operator identifier names an immutable search prior selected by the router and policy LLM, then realized by the motif-locked backend. Motif locks and validity checks are deterministic, although sequence decoding may be stochastic.

The operator identifier is also the unit whose visit count and empirical reward are updated after evaluation.

Four generic operator identifiers receive nonzero baseline prior weight in most metric regimes because they encode search safeguards that are useful across failure modes. `masked_motif_constraint_router` derives uncertainty from public-mask features only. `pareto_basin_archive` preserves separate pLDDT, motif-RMSD, and joint-threshold champions. `esmfold_oracle_calibration` keeps pLDDT and motif RMSD as separate signals rather than treating confidence as a substitute for motif geometry. `scale_matched_refinement` maps the metric regime to local, medium, or restart-scale actions. These are operator identifiers, not source-code names.

1. `masked_motif_constraint_router`: computes only public-mask features such as fixed fraction, span count, gap lengths, and motif composition, and uses them to diversify search hypotheses without asserting a source protein, fold family, or reference topology.
2. `pareto_basin_archive`: keeps separate roles for low motif RMSD, high pLDDT, joint-threshold proximity, and sequence/model-family diversity, so that candidates approaching different objectives are not prematurely discarded.
3. `esmfold_oracle_calibration`: treats pLDDT and motif RMSD as distinct evaluator signals. High pLDDT does not imply correct motif geometry, and low-RMSD seeds are not discarded solely because distal regions have lower confidence.
4. `scale_matched_refinement`: chooses perturbation scale from the metric regime: small edits for low-confidence/low-RMSD candidates, medium edits for confidence-passing near-geometry failures, restarts for far-RMSD failures, and broader model/topology exploration when neither metric is near threshold.
5. `joint_sequence_structure_prior`: used when sequence-only diffusion repeatedly produces high-confidence but geometry-wrong folds. It fixes public motif tokens, re-samples mutable scaffold tokens, and rejects collapsed low-entropy or non-improving arms.
6. `feedback_embedding_surrogate`: used after enough blinded evaluations exist. Candidate embeddings and measured pLDDT/RMSD are used only as action-selection priors; the structural evaluator remains the source of measured reward.
7. `feedback_guided_multisite_recombination`: used when several generated candidates occupy useful but incomplete basins or local mutation has plateaued. Parents and donors come only from generated candidates, and every recombinant is re-folded.
8. `near_threshold_trust_region`: used when a candidate is close to or passes the online proxy thresholds. It treats the near-threshold parent as a local trust-region center and switches scale when motif RMSD stops improving.
9. `confidence_geometry_plateau_test`: used when pLDDT improves but motif RMSD does not. It probes mutable scaffold sites and retires local point-mutation or additive recombination operators after repeated RMSD stagnation.
10. `single_site_feedback_recombination`: used near a pass threshold when candidate-local low-confidence scaffold positions are available. It evaluates single-site non-motif edits and combines only measured threshold-improving substitutions.
11. `candidate_backbone_inverse_folding`: used when motif geometry is good but confidence is insufficient, or when a fragile proxy-feasible state needs conservative support. It conditions only on the candidate's own predicted structure and fixes public motif positions.
12. `public_motif_repeat_register`: used when public motif spans suggest repeated turn signatures. The repeat register is inferred only from public fixed positions and is treated as a hypothesis rather than proof of a reference fold.
13. `public_generated_scaffold_transfer`: used when a generated proxy-feasible scaffold can be legally transferred to another public mask. Donors are generated candidates only, and motif residues are overwritten by the target mask.

Table 11: Static action arms used by the routing system.

Arm	Purpose	How it is used
plm_only	unbiased sequence-model control	baseline arm; least-biased model distribution
helix_bundle	alpha-helical scaffold hypothesis	helix-compatible residues and amphipathic periodicity
beta_core	buried beta-sheet hypothesis	alternating hydrophobic/polar beta-like patterning
beta_edge_turn	beta-edge plus turn hypothesis	turn-compatible residues near boundaries and edge-like polarity
globular_balance	soluble compact sequence prior	weak natural-composition regularization; avoids long hydrophobic/charged runs
minimal_frustration	local coherence repair	low-confidence local repair without global rewrite
junction_closure	repair linkers between fixed motif fragments	prioritized for multi-fragment masks and geometry-limited parents
transmembrane_bundle	hydrophobic multipass helix hypothesis	weak hypothesis from public hydrophobic blocks only
charge_balanced_globular	escape charge-rich low-complexity sequences	used when public or generated sequence is charge-heavy
segmented_helix_bundle	break long rods into helix-turn-helix segments	combats continuous coiled-coil-like artifacts
alpha_repeat	alpha-solenoid/repeat hypothesis	tested when public motif spans suggest repeat register
alpha_beta_layer	mixed alpha/beta topology	topology switch when pure helix/beta hypotheses fail

14. threshold_beam_recombination: used when several low-RMSD variants narrowly miss the confidence threshold. A small additive-effect beam selects combinations, but each generated child is still folded and scored.
15. robust_confidence_screen: used when pLDDT is brittle or high pLDDT repeatedly appears in wrong-geometry folds. Confidence is never treated as a substitute for motif RMSD.
16. reward_guided_partial_renoising: used when a parent is better than fresh samples but local mutation has plateaued. Motif residues remain fixed, scaffold residues are partially re-noised, and repair fraction is escalated only after plateau.
17. multifragment_topology_portfolio: used for many-span masks or repeated incompatible high-confidence geometry basins. It maintains separate topology islands and promotes an arm only after motif-RMSD evidence.
18. sequence_only_model_diversification: used when one generator repeatedly samples the same failing basin. It switches model family/capacity or latent prior while preserving public motif residues and excluding MSA-, homolog-, or reference-conditioned generation.
19. budgeted_sequence_only_sampling: used at virtual roots or when no better structural prior exists. It fixes motif residues, masks all scaffold residues, escalates independent samples in budget gates, and stops dominated arms.

C.5 Static Action Arms

Static arms are sequence-design hypotheses used by root generation, sequence model conditioning, patterned fallback generation, and topology-diversity schedules. They are not hard structural labels. The router ranks them from public mask features and candidate feedback.

The patterned fallback maps these arms to residue pools: transmembrane arms use hydrophobic residues in helix interiors and polar/charged residues in loop positions; helix arms use A/L/E/K/Q/M-like helix residues with heptad-like hydrophobic patterning; beta arms use V/I/T/F/Y alternating with polar or charged residues; junction_closure uses D/N/G/S/P near fixed motif boundaries; alpha_beta_layer cycles beta, turn, helix, and turn windows; and plm_only uses a broad canonical amino-acid pool.

C.6 Operator-Guided MCTS Expansion

The same operator inventory supports both success-oriented search and optional feasibility-preserving diversity search without introducing a separate staged method. In all cases, the pLDDT and motif-RMSD gates remain the primary online feasibility proxy. Virtual roots use budgeted sequence-only sampling, model diversification, and public-mask routing. Confidence-limited nodes use trust-region repair, single-site feedback, or candidate-backbone inverse folding. Geometry-near nodes use partial re-noising, feedback-guided recombination, and plateau tests. Geometry-far nodes use model diversification and topology portfolio restarts.

When the search objective includes diversity after feasibility, every inserted child must still pass the online feasibility proxy, and official native novelty remains unavailable online. The policy can

therefore use only generated-candidate signals, such as restart probes, parent re-noising, topology escape, motif-anchored chunk recombination, conservative mutation fill, and fragile proxy-feasible candidate-backbone support. A proxy-infeasible child is not inserted as a valid node for feasibility-preserving diversity search, but its penalty is backed up to discourage the branch.

Root and restart actions are full-mask generations with motif tokens fixed. Local actions mask exactly one scaffold position or a legal repair set. Medium actions partially re-noise scaffold residues with a bounded repair fraction. All generated sequences are validated before folding. For candidate-backbone inverse folding, the candidate's own predicted structure is used as the only structural conditioning source, public motif positions are fixed, high-consensus mutations are extracted relative to the parent sequence, and each child is folded again.

Operator specifications are never modified during a run. Each evaluated candidate carries an operator identifier; its reward is backed up through the selected tree path and also updates operator-level visit counts, accumulated reward, and mean reward once per backed-up evaluated child. Future operator selection combines the metric prior, empirical operator value, and exploration bonus. Plateau diagnostics can retire an operator for a parent or branch when measured children show that the operator is no longer useful. This gives the framework adaptive operator selection without changing the operator inventory or introducing hidden/native information into the action-execution path.

D Search Utility and Reward Computation

This appendix defines the scalar search utility backed up by MCTS. The utility is used only for tree control and is distinct from the final benchmark success definitions. In all main-text GeomMotif and MotifBench experiments, ELMS uses the online feasibility-proxy reward defined below. Benchmark metrics such as GeomMotif Successful, MotifBench final backbone success, Novelty, and Unique are recomputed from the evaluated candidate pools under the corresponding benchmark protocols; Novelty and Unique remain post hoc characterization metrics and do not enter the reported search objective.

Evaluator feedback. For an evaluated child state S_j, let

$$\varepsilon_j = (\nu_j, \kappa_j, \delta_j)$$

denote its sanitized evaluator feedback, where $\nu_j \in \{0, 1\}$ is a hard validity indicator, $\kappa_j \in [0, 100]$ is the predicted pLDDT, and $\delta_j \geq 0$ is the motif RMSD in Å. Hard validity requires that the sequence has the correct length, uses only canonical amino acids, preserves all locked motif residues, and returns parseable structural metrics.

We use the online feasibility-proxy thresholds

$$\kappa^\star = 70, \qquad \delta^\star = 1.0 \text{ Å}.$$

For reward arithmetic, motif RMSD is made dimensionless by

$$\bar{\delta}_j = \frac{\delta_j}{1 \text{ Å}}, \qquad \bar{\delta}^\star = \frac{\delta^\star}{1 \text{ Å}} = 1.0.$$

The online feasibility indicator for node j is

$$\tilde{y}_j = \mathbb{I}\left[\nu_j = 1,\ \kappa_j \geq \kappa^\star,\ \delta_j \leq \delta^\star\right].$$

This indicator matches the GeomMotif sequence-candidate feasibility gate, but for MotifBench it is only an online search proxy. Final MotifBench backbone success is recomputed after candidate generation by the benchmark protocol: eight ProteinMPNN sequence designs per backbone, ESMFold refolding, and a shared refolded design passing both motif RMSD and self-consistency RMSD gates without a pLDDT filter.

Threshold violation margins. For valid candidates with available pLDDT and RMSD, we define the normalized pLDDT violation and motif-RMSD violation as

$$\Delta_j^{\text{conf}} = \frac{\max(0, \kappa^\star - \kappa_j)}{10}, \qquad \Delta_j^{\text{geom}} = \max(0, \bar{\delta}_j - \bar{\delta}^\star).$$

The maximum and total threshold violations are

$$\Delta_j^{\max} = \max\left(\Delta_j^{\text{conf}}, \Delta_j^{\text{geom}}\right), \qquad \Delta_j^{\Sigma} = \Delta_j^{\text{conf}} + \Delta_j^{\text{geom}}.$$

Thus, a candidate that is below the pLDDT threshold or above the motif-RMSD threshold receives a shaped penalty even if it is not yet proxy-feasible.

We also use two small tie-breaking terms:

$$\omega_j^{\text{conf}} = \text{clip}\left(\frac{\kappa_j}{\kappa^\star}, 0, 1\right), \qquad \omega_j^{\text{geom}} = \exp(-\bar{\delta}_j).$$

Here ω_j^{conf} rewards candidates approaching the pLDDT threshold, and ω_j^{geom} mildly prefers lower motif RMSD. These terms are intentionally small relative to the proxy-feasibility bonus and serve as tie breakers among candidates with similar threshold violations.

Feasibility-proxy reward used in reported experiments. If the candidate is invalid, or if either pLDDT or motif RMSD is missing, we assign

$$R_j^{\text{proxy}} = -100.$$

Otherwise,

$$R_j^{\text{proxy}} = 10\tilde{y}_j - 2\Delta_j^{\max} - 0.5\Delta_j^{\Sigma} + 0.05\omega_j^{\text{conf}} + 0.05\omega_j^{\text{geom}}.$$

For the main-text experiments, the value backed up by MCTS is

$$R_j = R_j^{\text{proxy}}.$$

The large coefficient on $\tilde{y}_j$ separates proxy-feasible candidates from near misses, while the violation terms distinguish different failure modes. For example, a high-pLDDT structure with poor motif geometry is penalized by Δ_j^{geom}, whereas a geometry-preserving but low-confidence candidate is penalized by Δ_j^{conf}.

Optional generated-only diversity shaping. The implementation also supports exploratory diversity-shaped modes that are not used for the main-text results. These modes never use native-reference novelty or official Unique labels during search. They use only generated-candidate comparisons after feasibility has been assessed.

Let x_j be the sequence of candidate j. For two generated sequences x and y, define the normalized sequence distance

$$D_{\text{seq}}(x, y) = \frac{\sum_{i=1}^{\min(|x|,|y|)} \mathbb{I}[x_i \neq y_i] + |\,|x| - |y|\,|}{\max(|x|, |y|, 1)}.$$

Let $\mathcal{X}_{\text{arch}}$ be the archive of previously generated proxy-feasible sequences, and let $\mathcal{X}_{\text{leaf}}$ be the set of other proxy-feasible leaf sequences currently present in the tree. The generated-only archive and leaf-distance proxies are

$$\zeta_j^{\text{arch}} = \begin{cases} \min_{x \in \mathcal{X}_{\text{arch}}} D_{\text{seq}}(x_j, x), & |\mathcal{X}_{\text{arch}}| > 0, \\ 1, & |\mathcal{X}_{\text{arch}}| = 0, \end{cases}$$

and

$$\zeta_j^{\text{leaf}} = \begin{cases} \min_{x \in \mathcal{X}_{\text{leaf}}} D_{\text{seq}}(x_j, x), & |\mathcal{X}_{\text{leaf}}| > 0, \\ 1, & |\mathcal{X}_{\text{leaf}}| = 0. \end{cases}$$

The controller may also compute a generated-structure diversity proxy $\zeta_j^{\text{struct}} \in [0, 1]$ and a binary generated-structure novelty flag $\zeta_j^{\text{flag}} \in \{0, 1\}$. Both are computed only against generated candidates and do not use native reference structures.

For fresh de novo search with optional diversity shaping, proxy-infeasible candidates use the feasibility-proxy reward:

$$R_j^{\text{fresh}} = R_j^{\text{proxy}} \qquad \text{if } \tilde{y}_j = 0.$$

For proxy-feasible candidates, the reward adds generated-only diversity terms:

$$R_j^{\text{fresh}} = R_j^{\text{proxy}} + 1.5\zeta_j^{\text{arch}} + 1.0\zeta_j^{\text{leaf}} + 8.0\zeta_j^{\text{struct}} + 2.0\zeta_j^{\text{flag}} \qquad \text{if } \tilde{y}_j = 1.$$

In an optional feasible-diversity setting, the root is already proxy-feasible, and every inserted tree node is required to remain proxy-feasible. If a generated child fails the online feasibility proxy, it is not inserted as a valid child node and receives a large failure penalty,

$$R_j^{\text{feas}} = -10^6 \qquad \text{if } \tilde{y}_j = 0.$$

If the generated child remains proxy-feasible, the reward is

$$R_j^{\text{feas}} = 10 + 1.5\zeta_j^{\text{arch}} + 1.5\zeta_j^{\text{leaf}} + 20.0\zeta_j^{\text{struct}} + 5.0\zeta_j^{\text{flag}}.$$

Backpropagation and selection. After computing R_j, MCTS backs it up along the selected node path $\mathcal{G}_j$ from the root to child j. For every node $i \in \mathcal{G}_j$, we update

$$N_i \leftarrow N_i + 1, \qquad W_i \leftarrow W_i + R_j, \qquad Q_i \leftarrow W_i/N_i.$$

When a batch expansion creates multiple children, each child has its own reward and is backed up independently. Shared ancestor nodes receive one update per evaluated child. Consequently, N_i counts backed-up child evaluations through node i, rather than the number of outer MCTS iterations. Newly materialized nodes are initialized with $N_i = 0$ and $W_i = 0$; their Q_i value is not used until the first reward has been backed up through the node.

During tree traversal, legal but unevaluated children are expanded before UCB selection. Among already evaluated children with $N_j > 0$, a child node j of parent node i is selected by the same upper-confidence score used in the main text:

$$\text{UCB}(i, j) = Q_j + c_{\text{uct}} \sqrt{\frac{\log(N_i + 1)}{N_j + 1}}.$$

The constant c_{uct} corresponds to the exploration constant c in Section 4.5. The $+1$ terms stabilize the exploration factor for small visit counts but are not an initialization rule for Q_j at $N_j = 0$. Thus, the reward affects only search control: it determines which branches are expanded next, while final benchmark scores are computed separately from the evaluated candidate structures.

E Application of Motif-Locked Actions

This appendix describes how a selected scaffold-edit action is realized by the sequence-model backend. Each action specifies an operator identifier together with execution parameters. The key design principle is that the action is not executed as a free-form sequence edit. Instead, it is converted into a motif-locked masked decoding problem in which public motif residues are fixed and only legal scaffold positions are editable. All reported experiments use fixed-length action realization. Length-changing templates, if enabled in future work, would require updating the motif placement map Π, scaffold lengths $\boldsymbol{\ell}$, and public mask μ before the same validation and evaluation steps are applied.

Task mask and editable positions. Let $\mathcal{A}$ denote the set of 20 canonical amino acids. A motif-scaffolding task is represented by a length-L public mask

$$\mu = (\mu_1, \ldots, \mu_L),$$

where $\mu_i \in \mathcal{A}$ means that position i is a prescribed public motif residue, and $\mu_i = _$ means that position i is an editable scaffold position. We define the fixed motif set and scaffold set as

$$\Omega = \{i \in \{1, \ldots, L\} : \mu_i \in \mathcal{A}\}, \qquad \Gamma = \{i \in \{1, \ldots, L\} : \mu_i = _\}.$$

All actions must preserve the motif lock:

$$x_i = \mu_i, \qquad \forall i \in \Omega,$$

where $x = (x_1, \ldots, x_L)$ is any generated candidate sequence.

Action record. Each executable sequence-model action is represented as

$$\alpha = (o, h, \sigma, p_{\mathrm{src}}, I, \rho, \tau, n),$$

where o is the selected operator identifier, h is the optional decoding arm, σ is the search scale, p_{src} is the parent candidate identifier or for a restart action, $I \subseteq \{1, \ldots, L\}$ is the set of requested repair positions, $\rho \in (0, 1]$ is the repair fraction, $\tau > 0$ is the sampling temperature, and n is the number of child sequences requested from this action. The search scale satisfies

$$\sigma \in \{\mathrm{restart}, \mathrm{medium}, \mathrm{local}\}.$$

The decoding arm h is selected from a fixed set $\mathcal{H}$, including `plm_only`, `helix_bundle`, `beta_core`, `junction_closure`, `transmembrane_bundle`, `charge_balanced_globular`, `segmented_helix_bundle`, `alpha_repeat`, and `alpha_beta_layer`.

Construction of the mutable mask. The action first determines a mutable position set $\mathcal{M}_{\mathrm{edit}}(\alpha) \subseteq \Gamma$. For a restart action, no parent sequence is used and all scaffold positions are editable:

$$\mathcal{M}_{\mathrm{edit}}(\alpha) = \Gamma.$$

For a medium repair action, a generated parent sequence $x^{(p_{\mathrm{src}})}$ is available. The backend first keeps the legal requested positions

$$I_{\mathrm{legal}} = I \cap \Gamma,$$

then selects up to

$$n_\rho = \max(1, \lceil \rho |\Gamma| \rceil)$$

editable scaffold positions, prioritizing I_{legal} and, for junction-closure actions, internal gap-boundary positions. The resulting set is $\mathcal{M}_{\mathrm{edit}}(\alpha)$ with $|\mathcal{M}_{\mathrm{edit}}(\alpha)| \leq n_\rho$. For local repair actions, $\mathcal{M}_{\mathrm{edit}}(\alpha)$ contains one scaffold position at a time and is used to produce single-site conditional substitutions.

Masked sequence given to the model. Let ϕ be the tokenizer and let `[MASK]` denote the model mask token. The input token sequence $u = (u_1, \ldots, u_L)$ is constructed as

$$u_i = \begin{cases} \phi(\mu_i), & i \in \Omega, \\ \texttt{[MASK]}, & i \in \mathcal{M}_{\mathrm{edit}}(\alpha), \\ \phi(x_i^{(p_{\mathrm{src}})}), & i \in \Gamma \setminus \mathcal{M}_{\mathrm{edit}}(\alpha) \text{ and } p_{\mathrm{src}} \neq . \end{cases}$$

Thus, restart actions mask all non-motif scaffold positions, whereas repair actions start from a generated parent and mask only the selected repair subset. The origin or native sequence is never used to construct u.

Arm-conditioned logit bias. Let f_θ denote the masked protein language model, such as DPLM. At decoding step d, the model returns logits

$$\lambda_{i,d}(v) = f_\theta(u^{(d)})_{i,v},$$

where i is a sequence position, v is a vocabulary token, and $u^{(d)}$ is the current partially decoded sequence. Non-amino-acid tokens are disallowed by setting their logits to $-\infty$. For editable positions, the selected arm adds a deterministic bias to the amino-acid logits:

$$\tilde{\lambda}_{i,d}(z) = \lambda_{i,d}(\phi(z)) + b_h(i, z; u^{(d)}, \mu), \qquad z \in \mathcal{A},\ i \in \mathcal{M}_{\mathrm{edit}}(\alpha),$$

where b_h is the arm-specific additive bias. For `plm_only`, $b_h(i, z; u^{(d)}, \mu) = 0$, so the completion is an unbiased DPLM completion. Other arms implement weak scaffold priors. For example, `helix_bundle` increases helix-compatible residues and penalizes proline; `beta_core` encourages alternating hydrophobic and polar patterns; `junction_closure` adds turn-compatible residues near gap boundaries; and `transmembrane_bundle` increases hydrophobic residues in membrane-like phases while penalizing charged residues.

Iterative decoding. Let T_{dec} be the number of decoding iterations. In our implementation, $T_{\mathrm{dec}} = 100$ unless otherwise specified. At each decoding step, the backend samples an amino acid for each currently masked mutable position using Gumbel perturbation:

$$\hat{x}_{i,d} = \arg\max_{z \in \mathcal{A}} \left[\tilde{\lambda}_{i,d}(z) + \tau g_{i,z} \right],$$

where $g_{i,z}$ is an independent standard Gumbel random variable and τ is the sampling temperature. Larger τ gives more stochastic completions, while smaller τ makes the decoding more greedy.

To avoid low-complexity collapse, the decoder also applies a residue over-representation filter. Let $\eta \in (0, 1]$ be the maximum allowed mutable-site frequency for a single residue type, and let T_{retry} denote the fixed retry cap used for this validation check. We use $\eta = 0.25$ by default, with the same T_{retry} across all reported runs. This filter is applied only when $|\mathcal{M}_{\mathrm{edit}}(\alpha)| \geq 4$; for smaller edit masks it is disabled because a single residue occurrence would otherwise exceed the fractional threshold. When active, if one amino acid appears in more than $\eta|\mathcal{M}_{\mathrm{edit}}(\alpha)|$ mutable positions, those overrepresented choices are rejected at the affected positions and resampled from the same conditional logits, up to T_{retry} attempts. If the retry cap is reached, the child is rejected and the controller continues with the next legal action or sampled child until the evaluator budget is exhausted.

After sampling, the decoder performs reparameterized refinement. Let

$$q_d = 1 - \frac{d}{T_{\mathrm{dec}}}$$

be the remaining remasking rate at step d. The backend remasks approximately $q_d|\mathcal{M}_{\mathrm{edit}}(\alpha)|$ mutable positions with the lowest sampled confidence and keeps the remaining accepted residues fixed. This gradually converts a fully masked scaffold into a complete sequence while still allowing uncertain positions to be reconsidered.

Local single-site actions. For local repair actions, the backend uses a conditional single-site procedure instead of full iterative decoding. Let $\mathcal{J}_{\mathrm{rep}} \subseteq \Gamma$ be the set of candidate repair positions obtained from the requested positions and candidate-local fallback priorities. For each position $i \in \mathcal{J}_{\mathrm{rep}}$, the parent residue $x_i^{(p_{\mathrm{src}})}$ is replaced by `[MASK]` while all other positions remain fixed. The model produces log-probabilities over $\mathcal{A}$ at position i. The original parent residue is excluded, and the highest-scoring non-parent substitutions are converted into one-mutation child sequences. These children are useful near a success threshold because they test small edits without leaving the current structural basin.

Post-decoding validation. Every decoded child sequence x is accepted only if it passes all deterministic checks:

$$|x| = L, \qquad x_i \in \mathcal{A} \ \forall i, \qquad x_i = \mu_i \ \forall i \in \Omega.$$

The backend also removes duplicate sequences already present in the current search tree or current batch. Accepted children are then assigned metadata recording the operator o, arm h, search scale σ, parent identifier p_{src}, repair fraction ρ, edited positions, random seed, and sampling temperature τ. Only after this validation step is a child sequence submitted to structural evaluation and admitted into the MCTS tree.

F Strict No-Leakage LLM Interfaces

This appendix documents the strict no-leakage LLM interfaces used by the MCTS framework in this work. The same Critic–Policy interface is used across benchmarks; task-specific adapters provide the public motif mask, generated-candidate summaries, and evaluator feedback. The LLM agents operate as blinded reasoning components: the Critic Agent diagnoses the current evaluated state, and the Policy Agent chooses which operator to apply and with which execution parameters from a legal menu. Neither agent writes protein sequences, computes benchmark metrics, or accesses hidden benchmark references.

Table 12: Division of responsibility between LLM and non-LLM components.

Component	LLM?	Role
MCTS tree controller	No	selects a leaf by tree statistics and backs up reward
Evidence builder	No	converts evaluator outputs into state-local evidence records
Critic Agent	Yes	converts state-local evidence into constrained critic labels and rationales
Verifier	No	checks that critic outputs use legal labels and cite current-state evidence
Operator router	No	builds a legal menu of adaptive operators and static arms from public/sanitized evidence
Policy Agent	Yes	selects legal operators and execution parameters from the verified critic and legal menu
Motif-lock backend	No	realizes selected actions as motif-locked candidate sequences
Structure evaluator	No	refolds candidates and returns pLDDT, motif RMSD, online feasibility flags, and sanitized feedback
Official evaluator	No	computes benchmark metrics after the final candidate set is frozen

F.1 LLM Roles

The Critic Agent decides how to interpret the current state's failures or near-threshold behavior. The Policy Agent decides how to search next by choosing operator identifiers and execution parameters. Sequence construction, folding, scoring, and final benchmark evaluation remain outside the LLM agents.

F.2 Critic Agent Interface

The Critic Agent receives a compact record of the selected evaluated state. This record contains only public motif-mask information, parent action metadata, current pLDDT, motif RMSD, online feasibility-proxy margins, hard-validity flags, low-confidence spans, state-local evidence identifiers, and visit statistics. It does not receive origin sequences, native structures, hidden motif coordinates, homologs, MSAs, retrieved sequences, post hoc novelty labels, or benchmark-reference similarity information.

The Critic Agent emits constrained diagnostic labels such as `quality_failed`, `motif_geometry_failed`, `near_threshold`, or `state_successful`, together with short rationales grounded in the supplied evidence. The core critic prompt is shown below.

Critic Agent Core Prompt

```
You are the Critic Agent of a blinded protein motif-scaffolding MCTS.
Use only the supplied state-local evidence, including pLDDT, motif
RMSD, online feasibility-proxy margins, hard-validity flags,
low-confidence spans, and current node metadata.
Diagnose the current candidate using only allowed critic labels and
provide a concise evidence-grounded rationale.
Do not write protein sequences, do not choose edit positions, do not
request reference data, do not infer hidden origins, and do not use
native benchmark information.
Return only the structured critic output.
```

The verifier then checks that the labels are legal and that any referenced evidence belongs to the same selected state. Unsupported claims or illegal labels are rejected before the policy step.

F.3 Policy Agent System Prompt

The Policy Agent receives the verified critic output, generated-candidate summaries, the current metric regime, allowed operator identifiers, allowed static arms, legal parent ids, and legal mutable repair positions. Its role is to choose a search action from this menu, consisting of an operator identifier and execution parameters, not to write a sequence. The system prompt is shown below.

Policy Agent System Prompt

```
You are the policy component of a blinded protein motif-scaffolding
MCTS. You plan experiments but never write protein sequences. Use
only the supplied anonymous public mask, mask-derived features,
generated-candidate summaries, pLDDT, motif RMSD, and immutable
generic operators. Do not identify a source protein, reconstruct a
hidden scaffold, request reference data, or request
sequence/structure retrieval. For online search control, feasibility
requires the same candidate to have pLDDT ≥ 70 and motif RMSD ≤ 1 A;
final benchmark success is computed outside the prompt by the
benchmark adapter. When the objective is fresh_diversity, optimize
generated-only diversity after feasibility: prefer independent
restarts, model-family/arm diversity, and medium re-noising to find
distinct proxy-feasible basins before local polishing. A pLDDT rise
without RMSD movement is not geometry progress. Measured mutation
effects are background-local. Return one JSON object only.
```

F.4 Policy Agent User Prompt Template

For each call, the user message contains task-level instructions, a serialized blinded request payload, and the required JSON shape. The main policy template is shown below.

Policy Agent User Prompt Template

```
For every task choose one or two supplied adaptive operators and one
or two supplied static arms. Use restart at a virtual root or for
high-confidence far-RMSD topology. Use these disjoint measured
regimes: for confidence-passing 1.0 < RMSD ≤ 1.5 A use one-site/local
repair; for confidence-passing 1.5 < RMSD ≤ 3.0 A use medium partial
re-noising/recombination; for RMSD ≤ 1 A with low confidence use
candidate-backbone inverse folding when available. For DPLM choose
temperature in [0.6, 1.2] and medium repair_fraction in [0.10, 0.35];
local mode always masks exactly one scaffold site regardless of
repair_fraction. If an RMSD-passing parent remains more than 8 pLDDT
points below threshold after a local batch without a meaningful
confidence gain, use a conservative medium partial re-noising step
near 0.10 instead of repeating the same one-site action indefinitely.
Never select a retired operator. Parent IDs and repair positions
must come from the supplied legal lists. Preserve distinct pLDDT,
RMSD, joint-threshold, and sequence/model-family Pareto roles.
BLIND_REQUESTS=⟨json-serialized blinded request list⟩
REQUIRED_SHAPE=⟨json-serialized required response schema⟩
```

F.5 Blinded Request Payload

Each policy call receives a compact blinded request rather than hidden benchmark assets or implementation details. The request exposes only the public task identifier, target length, public motif mask, mask-derived features, generated candidate summaries, candidate-local pLDDT, motif RMSD, online feasibility flags, low-confidence spans, the current metric regime, non-retired allowed operators, allowed static arms, legal parent ids, and legal mutable repair positions.

Blinded Request Constraint

```
Use only supplied public-mask features and generated-candidate
feedback. Parent ids and repair positions must come from the legal
lists. Official native metrics are not available online.
```

The request excludes origin sequences, native structures, hidden motif coordinates, templates, homologs, MSAs, retrieved sequences, and official native Novelty or Unique labels. These restrictions prevent the LLM from jumping to an arbitrary branch, editing fixed motif positions, or using reference-derived information during online search.

F.6 Required Output and Validation

The Policy Agent returns one JSON decision per task. Each decision contains only the failure class, selected adaptive operator ids, search scale, Pareto role, selected arms, parent ids, repair positions, repair fraction, temperature, restart flag, short reason, stop condition, and no-leakage check.

Policy Output Constraint

```
Return one JSON object only; select only from supplied operators,
arms, parents, and repair positions.
```

Validation drops unexpected fields and enforces the legal menu. Selected operators must be non-retired allowed operators, arms must be allowed arms, parent ids and repair positions must come from their legal lists, and restart decisions clear parent and repair-position fields. Temperature is clamped to $[0.6, 1.2]$ and medium repair fraction to $[0.10, 0.35]$. If no valid decision remains, the controller falls back to the highest-ranked legal operator and arm selected by the router.

F.7 Strict No-Leakage Audit Example

In one audited root-state call, the policy saw only a length-247 public mask with 38 fixed residues across six spans, no generated parent feedback, the `virtual_root_unmeasured` regime, and a legal menu of restart-compatible operators and arms. No origin sequence, native PDB, native coordinates, homolog, MSA, or retrieved sequence was provided.

The validated decision selected `masked_motif_constraint_router` and `multifragment_topology_portfolio`, used restart-scale search, and chose `plm_only` together with `transmembrane_bundle`. The stated reason was that six fixed spans warranted diverse topology restarts with an unbiased control and a hydrophobic-topology arm. This example illustrates the intended behavior: the LLM routes the next search experiment from public and generated evidence only, while sequence construction and scoring remain outside the LLM.

F.8 Use of the Validated Decision by MCTS

The validated policy decision is attached to the rollout and consumed by the motif-lock backend. The selected operator ids define which adaptive-operator hypothesis is being tested; the selected arms define the sequence-design or topology/composition hypothesis; the search scale and restart flag determine whether the action is local, medium, or a full motif-locked restart; and the repair fraction, temperature, and repair positions parameterize the action. Generated children are validated, folded, and assigned an online search reward. The resulting reward is backed up to the MCTS path and to the corresponding operator statistics. Thus, the prompt controls operator selection and parameterization while sequence generation and scoring remain outside the LLM.

G AI Use Details

Generative AI assistance was used for language polishing after the scientific claims, method design, experimental setup, numerical results, and citations had been specified by the authors. The edits were limited to clarity, grammar, compression, and consistency of terminology. They did not introduce new experiments, results, methods, citations, or interpretations. Two representative wording-only examples are shown below.

Condensing the method overview. *Before polishing:*

> To address this question, we introduce an evidence-based, LLM-guided Monte Carlo Tree Search framework for evaluator-in-the-loop motif scaffolding. By organizing candidate scaffolds into an adaptive search tree, the framework uses MCTS to allocate a limited evaluation budget toward scaffold hypotheses that are both promising and insufficiently explored, rather than treating each generated candidate as an independent sample. By converting structural evaluation into state-local evidence, the framework makes foldability, motif preservation, and constraint satisfaction actionable signals for deciding how a candidate should be improved.

After polishing:

> To address this question, we introduce ELMS for evaluator-in-the-loop motif scaffolding. ELMS treats evaluated scaffolds as persistent search states: a Critic Agent summarizes state-local structural failures, a Policy Agent selects a legal operator with execution parameters, motif-locked operators realize the edit, and MCTS decides which states receive further design effort. This turns structural evaluation from a terminal filter into an inference-time signal for allocating a limited design budget.

Before polishing:

> Across GeomMotif and MotifBench, the framework consistently outperforms strong generative baselines. Under the standard GeomMotif protocol, it achieves Successful rates of 86.41% on single-motif tasks and 84.57% on paired-motif tasks, exceeding the strongest prior baseline by 19.3 and 21.9 percentage points, respectively. Under the matched 100-candidate MotifBench setting, it solves 26.7 of 30 tasks on average (88.89% Task Success), compared with 16.0 tasks (53.33%) for the strongest baseline.

After polishing:

> Across GeomMotif and MotifBench, the framework consistently outperforms the strongest generative baselines under matched candidate/evaluator budgets. These results highlight ELMS as a promising approach for combining structural evaluation, agentic reasoning, and adaptive search in constrained protein design.