LARGE LANGUAGE MODEL IS SECRETLY A PROTEIN SEQUENCE OPTIMIZER

Yinkai Wang* Tufts University
Jiaxing He Northeastern University
Yuanqi Du Cornell University
Xiaohui Chen Tufts University
Jianan Canal Li UC Berkeley
Li-Ping Liu Tufts University
Xiaolin Xu Northeastern University
Soha Hassoun* Tufts University

ABSTRACT

We consider the protein sequence engineering problem, which aims to find protein sequences with high fitness levels, starting from a given wild-type sequence. Directed evolution has been a dominating paradigm in this field which has an iterative process to generate variants and select via experimental feedback. We demonstrate large language models (LLMs), despite being trained on massive texts, are secretly protein sequence optimizers. With a directed evolutionary method, LLM can perform protein engineering through Pareto and experiment-budget constrained optimization, demonstrating success on both synthetic and experimental fitness landscapes.

1 INTRODUCTION

Protein engineering aims to develop novel protein sequences exhibiting improved or new-to-nature functions (Romero & Arnold, 2009). *Directed evolution* stands as a cornerstone paradigm of the field which leverages iterative rounds of mutagenesis and experimental selection to yield variants with gradually enhanced fitness (Arnold, 1998). While classical directed evolution has proven effective, it is generally acknowledged that its greedy optimization process often converges on suboptimal variants once a local maximum in the sequence fitness landscape of activity is reached (Yang et al., 2019). In recently proposed machine-learning guided directed evolution (MLDE) settings, sequence-to-function models have been incorporated as a computational surrogate to select candidates for experimental validation (Yang et al., 2019; Kirjner et al., 2023; Ren et al., 2022; Jain et al., 2022; Brookes et al., 2019; Yang et al., 2024).

With the grand success of AlphaFold2 on accurately predicting protein tertiary structures (Jumper et al., 2021), numerous work study protein language models (PLMs) which do not rely on multiple sequence alignment (MSA) and instead counting on learning the co-evolution information from multi-head attention transformers (Lin et al., 2023; Zhang et al., 2024). Motivated by the improved performance on structure prediction emerged from sequence-based pre-training, it has been employed as part of an evolutionary method which designs two masking strategies as mutation operators (Tran & Hy, 2024). More recently, increasing attention has been attracted to leverage large language models (LLMs) for problems in scientific discovery, e.g. molecule optimization (Wang et al., 2024), materials discovery (Lu et al., 2024). In protein engineering, Chen et al. (2024) propose a bi-level optimization to iteratively fine-tune pre-trained LLMs for protein optimization.

In this paper, we demonstrate LLMs themselves can already optimize protein fitness on-the-fly without further fine-tuning. Specifically, we build an evolutionary method that directly samples from pre-trained LLMs and select high fitness and low editing distance candidates for the next iteration. We count on LLMs to propose new candidates (i.e. mutation and crossover) to guide the search. Upon multiple experiments from 1) experiment-derived exact fitness landscapes, 2) simulated synthetic fitness landscapes, and 3) machine learning (ML) fitness landscape models trained on deep mutational scanning (DMS) datasets, we demonstrate LLMs can effectively propose new candidates that are much more efficient than the straightforward evolutionary algorithm with random muta-

[1]Correspondence to: Yinkai Wang <yinkai.wang@tufts.edu> and Soha Hassoun <soha@cs.tufts.edu>.

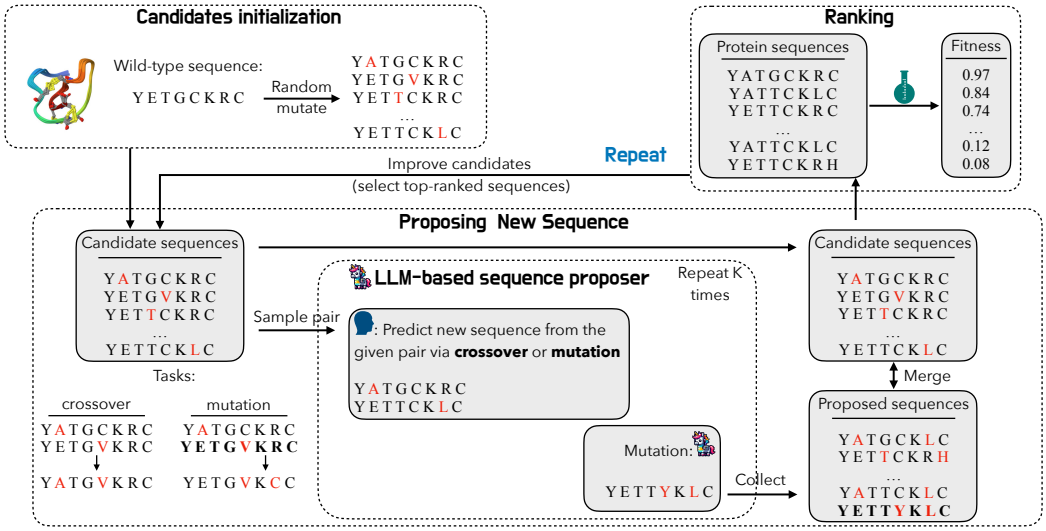


Figure 1: The overview of the optimization framework.

tion and crossover. We also extend the experiment setting to experiment-budget constrained and multi-objective optimization.

2 Preliminary: Protein Sequence Optimization

Single-objective optimization. Given an oracle function $f : \Omega \to \mathbb{R}$, where $\Omega := \{(a_1, a_2, \cdots, a_L)|a_i \in \mathcal{A}\}$, L is the maximum length of a protein sequence, and $\mathcal{A}$ is the set of 20 amino acid types, we aim to find the candidate x^* as follows:

$$x^* = \arg\max_{x \in \Omega} f(x) \tag{1}$$

Constrained optimization. Beyond merely optimizing the oracle function, we are often limited by experimental budget such that we constrain the maximum number of edits to be K from the reference wild type x_{ref}.

$$x^* = \arg\max_{x \in \Omega} f(x), \text{ s.t. } \text{dist}(x, x_{\text{ref}}) \leq K \tag{2}$$

where the distance function is taken as the Hamming distance $d_H(\cdot, \cdot)$ between two sequences.

Budget-constrained optimization. Instead of constraining the absolute Hamming distance, a more realistic setting in wet-lab experiments is to constrain the relative Hamming distance (i.e. minimum Hamming distance between the proposed sequence and all previous experiment trials).

$$\text{dist}(x, \mathcal{P}) = \min_{x_p \in \mathcal{P}} d_H(x, x_p) \tag{3}$$

where $\mathcal{P}$ is the set of all previously evaluated candidates.

Multi-objective optimization. In scenarios where we have multiple oracle functions to optimize, we solve a multi-objective optimization problem where the objective function becomes a vector-valued function $f : \Omega \to \mathbb{R}^d$:

$$x^* = \arg\max_{x \in \Omega} f(x) \tag{4}$$

One simple way to aggregate multiple objectives is to take a weighted sum over the output vector $\sum_j w_j f_j(x)$ and $\sum_j w_j = 1$, where we refer to as *sum of objectives*.

Nevertheless, the more rigorous formulation is to find the Pareto frontier $\mathcal{P}$, defined as follows:

$$\mathcal{P}(\mathcal{X}) = \left\{x \in \mathcal{X} : \{x' \in \mathcal{X} : x \preceq x', x \neq x'\} = \varnothing\right\} \tag{5}$$

where $\preceq$ defines a partial order such that $x \preceq x'$ or x is dominated by x' if and only if $\forall_j \ f_j(x') \geq f_j(x)$. We refer the problem to find the Pareto set to as *Pareto set selection*.

Dataset	Space size	Sequence length	# mutation sites	Oracle	Target range	Initial pool fitness	wild-type fitness
Syn-3bfo	N/A	85	85	SLIP	N/A	-4.11±2.22	0.00
GB1	149,361	56	4	exact	$[0, 8.76]$	0.08±0.40	1.00
TrpB	159,129	397	4	exact	$[0, 1]$	0.02±0.06	0.41
AAV	N/A	735	28	ML	N/A	0.31±0.05	0.56
GFP	N/A	238	237	ML	N/A	0.08±0.12	0.94

Table 1: Dataset statistics. Syn-3bfo refer to synthetic dataset constructed from PDB ID 3bfo.

3 METHODOLOGY

We propose an evolutionary method for protein sequence optimization. There are three main modules in our method: (1) initialization, (2) diversification and (3) selection. The framework is illustrated in Figure 1 and the pseudocode is included in Algorithm 1.

Initialization. We initialize a pool of candidates by randomly sampling from the entire space or a single mutation from the wild type.

Mutation/Crossover. The default mutation in evolutionary algorithm (EA) is to perform a random mutation over a single protein sequence; the default crossover in EA is to randomly swap amino acids of two protein sequences at the same position or swap the entire half sequence split by a random position. In our LLM-based method, we randomly sample a pair of protein sequences from our pool and encourage LLMs to propose a new candidate either through mutation or crossover.

Selection. For single-objective optimization, we simply select the top-k ranked protein sequences in both the previous pool and the newly proposed candidates. For constrained optimization, we employ a rejection sampling-based strategy: we discard all samples that violate the constraints (exceeding the maximum number of edits allowed. For multi-objective optimization, we optimize for two objectives: (1) **objective scalarization**: we sum over all objective values in the multi-objective vectors and treat it as a single-objective optimization problem; (2) **Pareto set selection**: we select only the candidates on the Pareto frontier to proceed the next iteration.

4 EXPERIMENT

4.1 EXPERIMENT SET-UP

Oracle function. We have three types of oracle functions: **exact oracle**, **synthetic SLIP model oracle**, and **ML oracle**. For **exact oracle**, directly measures the fitness values of all possible variants in a specified search space by deep mutational scanning (DMS) (Fowler & Fields, 2014). Due to experimental budget constraints, the number of sites to be mutated is often limited to four or fewer.

For the **synthetic SLIP oracle**, the statistical energy of protein variants evaluated by the Potts model has been demonstrated to correlate with observed empirical fitnes (Hopf et al., 2017), and the Synthetic Landscape Inference for Proteins (SLIP) based on Potts models has been proposed as a hard-to-optimize fitness landscape (Thomas et al., 2022).

For **ML oracle**, a machine learning model is trained on sequence–fitness pairs of single and multiple mutants for a wild-type protein through DMS (Dallago et al., 2021). Unlike the exact oracle, which focuses on a small subset of variants, the ML oracle can evaluate protein variants with any number of mutations away from the wild-type sequence, returning a predicted fitness value. However, its generalization ability remains a key limitation: because the model is typically trained on a comparatively small dataset, its predictions may be unreliable across the full sequence space.

Hyperparameters. We adopt the *Llama-3.1-8B-Instruct* model as our LLM. To mimic real-world protein engineering experiment procedure, we choose a set of 32/48/96 candidates in each iteration for a total of 4 iterations. For experiment settings allowing a larger number of mutations away from the wild-type, we increase to 8 iterations for better optimization.

Baselines. We use the exactly same hyperparameters and initial pools for the baseline evolutionary algorithm as our model, we adopt the default mutation and crossover operators for EA in Section 3.

Datasets. Here we list the datasets used for each type of oracle function:

- For the exact oracle setting, we test our framework on two combinatorial landscape datasets GB1 (Wu et al., 2016) and TrpB (Johnston et al., 2024). On these landscapes, four amino acids

Dataset	Method	Population × iteration	Fitness score		
			Top 1	Top 10	Top 50
GB1	EA	32×4	**5.38±1.77**	**3.81±1.10**	**2.31±0.71**
		48×4	**4.88±0.33**	3.72±0.38	2.17±0.27
		96×4	**5.72±0.56**	**4.32±0.53**	2.84±0.60
	Ours	32×4	4.34±0.53	3.22±0.23	1.94±0.28
		48×4	4.31±0.82	**3.76±0.82**	**2.45±0.61**
		96×4	4.80±0.52	4.09±0.19	**3.04±0.19**
TrpB	EA	32×4	0.20±0.18	0.14±0.12	0.07±0.05
		48×4	0.67±0.14	0.52±0.11	0.19±0.04
		96×4	0.74±0.01	0.59±0.03	0.35±0.10
	Ours	32×4	**0.60±0.10**	**0.50±0.07**	**0.35±0.07**
		48×4	**0.68±0.04**	**0.58±0.01**	**0.36±0.01**
		96×4	**0.78±0.20**	**0.60±0.16**	**0.39±0.16**
Syn-3bfo	EA	32×8	0.57±0.21	-0.44±0.11	-1.35±0.17
		48×8	1.29±0.36	0.42±0.24	-0.63±0.07
		96×8	1.85±0.47	1.10±0.28	0.07±0.28
	Ours	32×8	**2.51±0.23**	**1.33±0.14**	**0.28±0.20**
		48×8	**2.35±0.26**	**1.36±0.11**	**0.04±0.09**
		96×8	**2.83±0.20**	**2.02±0.36**	**0.96±0.36**
AAV	EA	32×8	0.42±0.03	0.36±0.01	0.32±0.00
		48×8	0.44±0.00	0.38±0.01	0.33±0.00
		96×8	0.44±0.00	0.40±0.01	0.36±0.00
	Ours	32×8	**0.74±0.00**	**0.69±0.02**	**0.62±0.03**
		48×8	**0.75±0.01**	**0.71±0.01**	**0.64±0.02**
		96×8	**0.76±0.03**	**0.73±0.03**	**0.68±0.03**
GFP	EA	32×8	0.43±0.13	0.21±0.02	0.12±0.01
		48×8	0.43±0.14	0.26±0.05	0.12±0.01
		96×8	0.50±0.11	0.34±0.05	0.18±0.01
	Ours	32×8	**0.96±0.02**	**0.94±0.01**	**0.88±0.03**
		48×8	**0.96±0.02**	**0.93±0.01**	**0.84±0.02**
		96×8	**0.97±0.01**	**0.95±0.01**	**0.92±0.01**

Table 2: Single-objective optimization results for fitness optimization. We record the mean of top-k ranked candidates and report the mean and std over three random seeds. The best score for different population sizes and landscapes is **bold**.

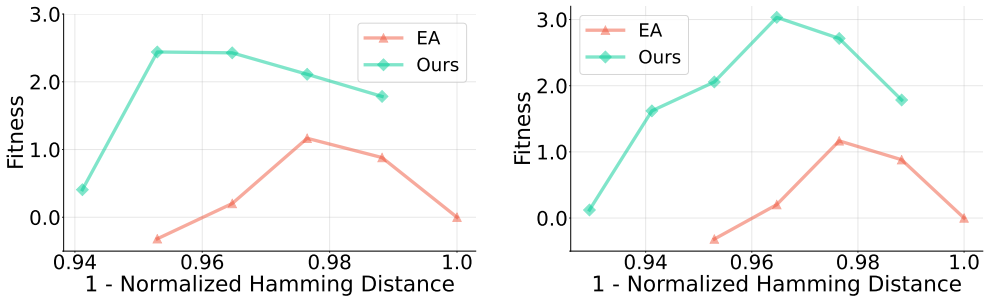
are picked to be mutated, therefore having a total of 20^4 variants. The fitness is measured by wet-lab experiments for nearly all the variants in the library.

- For the synthetic SLIP oracle setting, we create the tuned synthetic landscape constructed from the multiple sequence alignment for PDB ID 3bfo follow the guidance in the SLIP paper (Thomas et al., 2022).
- For the ML oracle setting, we evaluate our framework on two DMS datasets: Green Fluorescent Proteins (GFP) (Sarkisyan et al., 2016) and Adeno-Associated Virus (AAV) (Bryant et al., 2021). These DMS experiments include up to 15 mutations from the wild-type sequence. The fitness metric for GFP is based on its fluorescence properties as a biomarker, while for AAV, it is based by its ability to package a DNA payload for gene delivery. An ML oracle model is trained following Kirjner et al. (2023) to predict fitness for any variant.

4.2 Main Experiment

We validate our method in four settings to evaluate our method on protein sequence optimization.

Single-objective optimization. We conduct single-objective optimization on all five datasets. In this experiment, we follow the traditional directed evolution protocol, setting the number of proposed variants per iteration to 32, 48, and 96. For GB1 and TrpB, the number of iterations is set to 4, while for the other landscapes, the number of iterations is increased to 8 due to the larger number

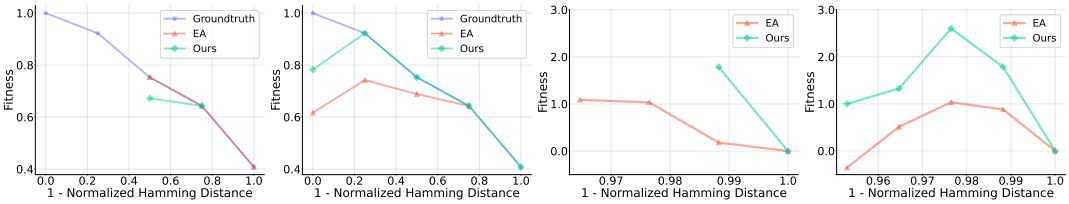


(a) Constrained optimization (b) Budget-constrained optimization

Figure 2: Pareto frontiers identified under constrained and budget-constrained optimization settings.

of possible mutation sites. The optimization objective is to maximize the fitness value from the oracle function, as detailed in Table 1. Among five datasets, GB1 and TrpB have more linear fitness landscapes, where finding a favorable amino acid at a position often leads to its presence in the optimal sequence (demonstrated by Figure 4). This allows EA to find strong variants early, sometimes outperforming our method. As shown in Figure 7, EA only outperforms one of three random seeds for GB1, while our framework performs better in the other two.

Since linear relationships between positions are less likely in more complex landscapes with larger search spaces, we also evaluate our framework on Syn-3bfo, AAV, and GFP, which have more mutation sites and nonlinear fitness landscapes (Table 1). For Syn-3bfo, the initial pool is generated from single mutations of the wild-type protein 3bfo, with fitness values calculated using the SLIP model. For the AAV and GFP datasets, our initial pool setting follows the medium difficulty criteria outlined in (Kirjner et al., 2023). This involves restricting the fitness range of the initial pool proteins to fall between a certain range and ensuring that the mutational gap from the highest-score protein in the given dataset is greater than 6. The predicted fitness value is normalized by min-max values of dataset. Our method consistently outperforms EA in these datasets (Table 2).



(a) TrpB (b) Syn-3bfo

Figure 3: Pareto frontiers identified under multi-objective optimizations. We display the Pareto frontiers found for TrpB (a) and Syn-3bfo (b), using *Pareto set selection* (left) and *sum of objectives* (right), respectively. We also show the groundtruth Pareto frontiers for TrpB.

Constrained optimization. In constrained optimization, we limit the number of mutations at each iteration to 3, 5, and 10, rejecting sequences that exceed these limits on the Syn-3bfo landscape. For our method, we add the prompt: *"The proposed sequence must have a Hamming distance between 1 and* $\{H\}$ *from the* $\{$*wild-type sequence*$\}$*"*, where H represents the Hamming distance constraint.

As shown in Table 4, our method demonstrates stable performance compared to EA. Notably, the performances of EA at constrained Hamming distances of 5 and 10 are the same, as the maximum Hamming distance of sequences proposed by EA does not exceed 5 within eight iterations. Our framework performs best when the constrained Hamming distance is set to 3. Additionally, we illustrate the Pareto frontier discovered during the constrained optimization tasks by selecting the best fitness value for each Hamming distance from the wild-type in Figure 2.

Budget-constrained optimization. In budget-constrained optimization, we restrict the maximum number of amino acids edited in a single iteration to 1, 2, and 4 on the Syn-3bfo landscape. Sequences exceeding this limit are dropped by rejection sampling. For our method, we include the prompt: *"The proposed sequence must have a Hamming distance between 1 and* $\{BH\}$ *from the* $\{$*parent sequence*$\}$*"*, where BH represents the constrained Hamming distance, and $\{$*parent sequence*$\}$ refers to the two parent sequences provided to the LLM for optimization.

Dataset	Method	Pareto (Top-k)			Sum (Top-k)		
		Top1	Top10	Top50	Top1	Top10	Top50
Syn-3bfo	EA	1.38±1.26	**0.82±0.79**	**0.82±0.79**	1.17±0.27	0.38±0.22	-0.64±0.33
	Ours	**1.52±0.37**	0.71±0.26	0.71±0.26	**1.73±0.27**	**0.82±0.28**	**-0.19±0.37**
TrpB	EA	**0.84±0.12**	**0.67±0.05**	**0.67±0.05**	0.67±0.02	0.60±0.01	0.47±0.01
	Ours	0.73±0.14	0.66±0.08	0.66±0.08	**0.69±0.04**	**0.62±0.05**	**0.50±0.06**

Table 3: Multi-objective optimization (Pareto and sum of objectives) with parameter 48×8.

Dataset	Method	H=3			H=5			H=10		
		Top1	Top10	Top50	Top1	Top10	Top50	Top1	Top10	Top50
Syn-3bfo	EA	1.20±0.42	0.51±0.26	-0.57±0.11	1.29±0.36	0.42±0.24	-0.63±0.07	1.29±0.36	0.42±0.24	-0.63±0.07
	Ours	**2.46±0.22**	**1.74±0.16**	**0.66±0.10**	**2.21±0.19**	**1.59±0.35**	**0.49±0.44**	**2.28±0.29**	**1.74±0.33**	**0.73±0.40**

Table 4: Constrained optimization results on Syn-3bfo. Each set of columns shows a different H.

Dataset	Method	H=1			H=2			H=4		
		Top1	Top10	Top50	Top1	Top10	Top50	Top1	Top10	Top50
Syn-3bfo	EA	1.36±0.45	0.78±0.33	-0.24±0.17	1.42±0.54	0.62±0.52	-0.48±0.35	1.29±0.36	0.42±0.24	-0.63±0.07
	Ours	**2.10±0.09**	**1.29±0.17**	**0.28±0.26**	**2.52±0.53**	**1.61±0.41**	**0.46±0.42**	**2.34±0.26**	**1.28±0.16**	**-0.01±0.07**

Table 5: Budget-constrained optimization on Syn-3bfo with different budget H.

The results in Table 5 show that our model outperforms EA across all three settings. Our method performs best when the maximum number of amino acids edited in a single iteration is limited to 2. Additionally, we present the Pareto frontier obtained from the budget-constrained optimization task, using the same settings as the constrained optimization shown in Figure 2.

Multi-objective optimization. We perform multi-objective optimization to simultaneously optimize the Hamming distance and fitness on the Syn-3bfo landscape. In the *sum of objectives* approach, the fitness value and $1 -$ normalized Hamming distance are combined into a single objective with equal weight. In the *Pareto set selection* approach, all dominated points are rejected, and optimization is restricted to points on the Pareto frontier.

The results from the first approach are summarized in Table 3 and compared against the evolutionary algorithm (EA). The Pareto frontiers identified by our framework and the EA for both approaches are illustrated in Figure 3. For the TrpB landscape, the true Pareto frontier can be determined as it is fully enumerated, and our method identifies more Pareto frontier points than EA in the sum-of-objectives setting. Moreover, the Pareto frontiers found by our method in the sum-of-objectives task dominate or are equivalent to those found by EA on both landscapes.

In the Pareto set selection setting, our method does not dominate all the Pareto frontiers identified by EA. This is because restricting the experiment pool to only include Pareto frontier points limits the LLM's access to sufficient information about the sequence space for optimization. However, our method still identifies Pareto frontier points that dominate those found by EA on Syn-3bfo landscape.

5 Conclusion

In this paper, we introduce an LLM-guided directed evolution framework for protein sequence optimization. We investigate a range of tasks, employing oracle functions of varying complexity—from synthetic landscapes to experimental ground-truth measurements and machine learning–based oracles. We conduct experiments on multiple optimization tasks from single-objective to constrained and multi-objective optimization. Our results consistently demonstrate the efficacy of LLMs in proposing high-fitness variants. Moving forward, integrating LLM-based optimization into real-world experimental pipelines can accelerate directed evolution experiments, allowing for more efficient exploration of the protein sequence space.

Acknowledgments

We thank Jason Yang for helpful discussions. This work was sponsored by Army Research Office, MURI program, contract #W911NF2210239.

MEANINGFULNESS STATEMENT

Our work demonstrates that large language models (LLMs) can serve as effective protein sequence optimizers by leveraging their learned representations of sequential structures. By integrating LLMs into a directed evolution framework, we provide a novel approach to exploring protein fitness landscapes, optimizing sequences efficiently without explicit fine-tuning.

REFERENCES

Frances H Arnold. Design by directed evolution. *Accounts of chemical research*, 31(3):125–131, 1998.

David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pp. 773–782. PMLR, 2019.

Drew H Bryant, Ali Bashir, Sam Sinai, Nina K Jain, Pierce J Ogden, Patrick F Riley, George M Church, Lucy J Colwell, and Eric D Kelsic. Deep diversification of an aav capsid protein by machine learning. *Nature Biotechnology*, 39(6):691–696, 2021.

Angelica Chen, Samuel D Stanton, Robert G Alberstein, Andrew M Watkins, Richard Bonneau, Vladimir Gligorijevi, Kyunghyun Cho, and Nathan C Frey. Llms are highly-constrained biophysical sequence optimizers. *arXiv preprint arXiv:2410.22296*, 2024.

Christian Dallago, Jody Mou, Kadina E Johnston, Bruce J Wittmann, Nicholas Bhattacharya, Samuel Goldman, Ali Madani, and Kevin K Yang. Flip: Benchmark tasks in fitness landscape inference for proteins. *bioRxiv*, pp. 2021–11, 2021.

Douglas M Fowler and Stanley Fields. Deep mutational scanning: a new style of protein science. *Nature methods*, 11(8):801–807, 2014.

Thomas A Hopf, John B Ingraham, Frank J Poelwijk, Charlotta PI Schärfe, Michael Springer, Chris Sander, and Debora S Marks. Mutation effects predicted from sequence co-variation. *Nature biotechnology*, 35(2):128–135, 2017.

Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pp. 9786–9801. PMLR, 2022.

Kadina E Johnston, Patrick J Almhjell, Ella J Watkins-Dulaney, Grace Liu, Nicholas J Porter, Jason Yang, and Frances H Arnold. A combinatorially complete epistatic fitness landscape in an enzyme active site. *Proceedings of the National Academy of Sciences*, 121(32):e2400439121, 2024.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Andrew Kirjner, Jason Yim, Raman Samusevich, Shahar Bracha, Tommi S Jaakkola, Regina Barzilay, and Ila R Fiete. Improving protein optimization with smoothed fitness landscapes. In *The Twelfth International Conference on Learning Representations*, 2023.

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.

Jieyu Lu, Zhangde Song, Qiyuan Zhao, Yuanqi Du, Yirui Cao, Haojun Jia, and Chenru Duan. Generative design of functional metal complexes utilizing the internal knowledge of large language models. *arXiv preprint arXiv:2410.18136*, 2024.

Zhizhou Ren, Jiahan Li, Fan Ding, Yuan Zhou, Jianzhu Ma, and Jian Peng. Proximal exploration for model-guided protein sequence design. In *International Conference on Machine Learning*, pp. 18520–18536. PMLR, 2022.

Philip A Romero and Frances H Arnold. Exploring protein fitness landscapes by directed evolution. *Nature reviews Molecular cell biology*, 10(12):866–876, 2009.

Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

Neil Thomas, Atish Agarwala, David Belanger, Yun S Song, and Lucy J Colwell. Tuned fitness landscapes for benchmarking model-guided protein design. *bioRxiv*, pp. 2022–10, 2022.

Thanh VT Tran and Truong Son Hy. Protein design by directed evolution guided by large language models. *IEEE Transactions on Evolutionary Computation*, 2024.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, et al. Efficient evolutionary search over chemical space with large language models. *arXiv preprint arXiv:2406.16976*, 2024.

Nicholas C Wu, Lei Dai, C Anders Olson, James O Lloyd-Smith, and Ren Sun. Adaptation in protein fitness landscapes is facilitated by indirect paths. *Elife*, 5:e16965, 2016.

Jason Yang, Ravi G Lal, James C Bowden, Raul Astudillo, Mikhail A Hameedi, Sukhvinder Kaur, Matthew Hill, Yisong Yue, and Frances H Arnold. Active learning-assisted directed evolution. *bioRxiv*, pp. 2024–07, 2024.

Kevin K Yang, Zachary Wu, and Frances H Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8):687–694, 2019.

Zhidian Zhang, Hannah K Wayment-Steele, Garyk Brixi, Haobo Wang, Dorothee Kern, and Sergey Ovchinnikov. Protein language models learn evolutionary statistics of interacting sequence motifs. *Proceedings of the National Academy of Sciences*, 121(45):e2406285121, 2024.

A APPENDIX

A.1 PSEUDOCODE

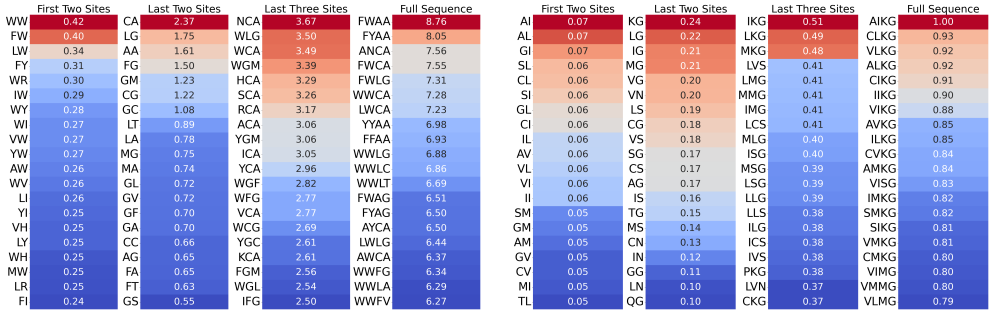
We show the pseudocode of our framework below.

Algorithm 1: Protein Sequence Optimization with LLM

Data: Initial population $\mathcal{P}_0$; mutation rate r_m; population size K; number of iterations N; the fitness function $F(\cdot)$; the default crossover function $C(\cdot, \cdot)$; the default mutation function $M(\cdot)$.
Result: Optimized protein population $\mathcal{P}_N$.
begin
  for $s \in \mathcal{P}_0$ **do**
    Compute $F(s)$;
  for $t \in [1, N]$ **do**
    `offspring = []`;
    for $k \in [1, K]$ **do**
      Draw parent sequences $(s_0, s_1) \sim \mathcal{P}_t \times \mathcal{P}_t$;
      `proposed_seq` $\leftarrow$ `LLM_propose`(s_0, s_1);
      if `proposed_seq` *is* `None` **then**
        `offspring.append`$(C(s_0, s_1))$;
        $r \sim$ `Uniform`$[0, 1]$
        if $r \leq r_m$ **then**
          `offspring.append`$(M(s_0))$;
      else
        `offspring.append(proposed_seq)`;
    for $s \in$ `offspring` **do**
      Compute $F(s)$;
    `merged_population` $\leftarrow$ `merge`$(\mathcal{P}_t,$ `offspring`$)$;
    $\mathcal{P}_t \leftarrow$ `sorted(merged_population)`$[:K]$;
  Return $\mathcal{P}_N$;

A.2 DATASETS ANALYZE

We present a heatmap of the average scores for specific combinations at different positions: the first two, last two, last three, and the full sequence for GB1 and TrpB in Figure 4. The heatmap reveals that certain combinations in the last two positions, such as CA, LG, and AA in GB1, and KG, LG, and IG in TrpB, exhibit higher fitness scores compared to others. Preserving these combinations can significantly simplify the path to identifying sequences with improved fitness.



(a) GB1 (b) TrpB

Figure 4: The fitness heatmaps of first two, last two, last three, and full sequence on two datasets.

A.3 PROMPTS

Prompt

system

You are a world-class assistant specializing in protein engineering, fitness optimization, and sequence design. Your expertise lies in analyzing sequence-function relationships, interpreting experimental data, and proposing rational modifications to optimize protein fitness.

user

You will carry out a multi-round directed evolution experiment with the following protein sequence, aimed at improving protein's ability to bind affinity-based sequence enrichment via protein fitness optimization.

Protein fitness optimization

The fitness score reflects the efficacy or functionality for a desired application, from chemical synthesis to bioremediation and therapeutics. Protein fitness optimization can be thought of as navigating a protein fitness landscape, a mapping of amino acid sequences to fitness values, to find higher-fitness variants. Specifically, it is achieved by making crossover and mutations on the given sequences.

We are focusing on changes to a limited subset of amino acids within the sequence. The provided subset protein sequences come from B1 domain of streptococcal protein G, with sequence:

```
MTYKLILNGKTLKGETTTEAVDAATAEKVFKQYANDNGVDGEWTYDDAT
KTFTVTE
```

Each subset protein sequence represents specific amino acid substitutions at four key positions: 39, 40, 41, and 54, denoted using the single-letter amino acid code.

Parent protein sequences

Here are the parent protein sequences that you will be modifying from. Each sequence comes with 4 amino acids and its fitness score is also provided.

Protein sequence 1 (fitness score: 0.0018)

```
Q H V R
```

Protein sequence 2 (fitness score: 0.0021)

```
R L I V
```

Instructions

Follow the instructions below to propose a new protein:

* Your proposal should focus on maximizing fitness and minimizing humming distance from the wild type while considering structural and functional plausibility.
* You can propose it via making crossover or mutation on the parent sequences.
* You can also propose a new sequence based on your knowledge.
* Your proposed sequence MUST have the same length as the parent sequences.
* DO NOT propose sequence that is identical with the parent or the wild type sequences.
* Your output MUST ONLY include: `\box{{Protein}}`.

A.4 PARETO FRONTIER

We present the Pareto frontiers identified through constrained optimization tasks with different Budget H and H, as shown in Figure 5. The figures shows our method consistently dominate EA.

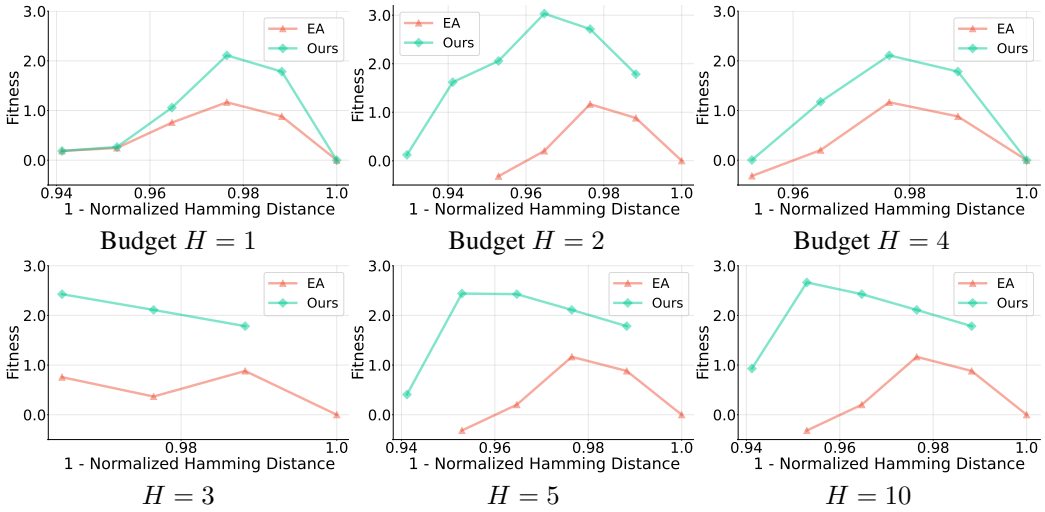


Figure 5: The Pareto frontiers identified by EA and our method in both constrained and budget-constrained optimization settings for all parameter configurations.

We also present the Pareto frontiers identified by our method for different tasks on the Syn-3bfo dataset in Figure 6, illustrating how the choice of objectives to optimize influences the discovery of the Pareto frontier.

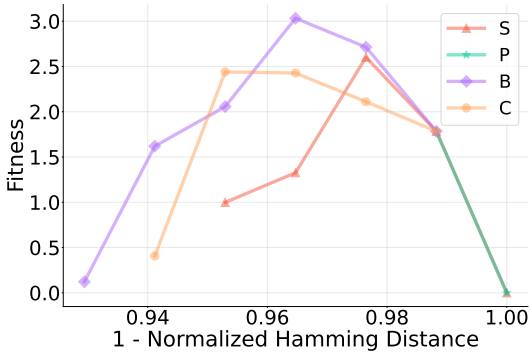


Figure 6: The Pareto frontier found by our method via different task. S stand for *Sum of objective*, P stand for *Pareto frontier set selection*, B stand for Budget-constrained, and C stand for the Constrained.

A.5 Ablation Study of the Number of Iterations

We analyze the impact of varying the number of iterations on the results in Table 6. The analysis shows that as the number of iterations increases, both EA and our framework improve in performance. However, our method consistently maintains its advantage over EA.

Dataset	Method	Iteration	Fitness score		
			Top 1	Top 10	Top 50
Syn-3bfo	EA	8	1.85±0.47	1.10±0.28	0.07±0.28
		12	1.97±0.58	1.52±0.41	0.77±0.27
		16	2.38±0.45	1.99±0.42	1.34±0.34
	Ours	8	**2.83±0.20**	**2.02±0.36**	**0.96±0.36**
		12	**3.03±0.29**	**2.51±0.36**	**1.66±0.44**
		16	**3.84±0.44**	**3.29±0.36**	**2.48±0.34**

Table 6: Ablation study on different iterations for Syn-3bfo landscape with 96 population size.

A.6 Additional Experiments Result

We present additional experiments,show results across three random seeds and datasets in Figure 7.

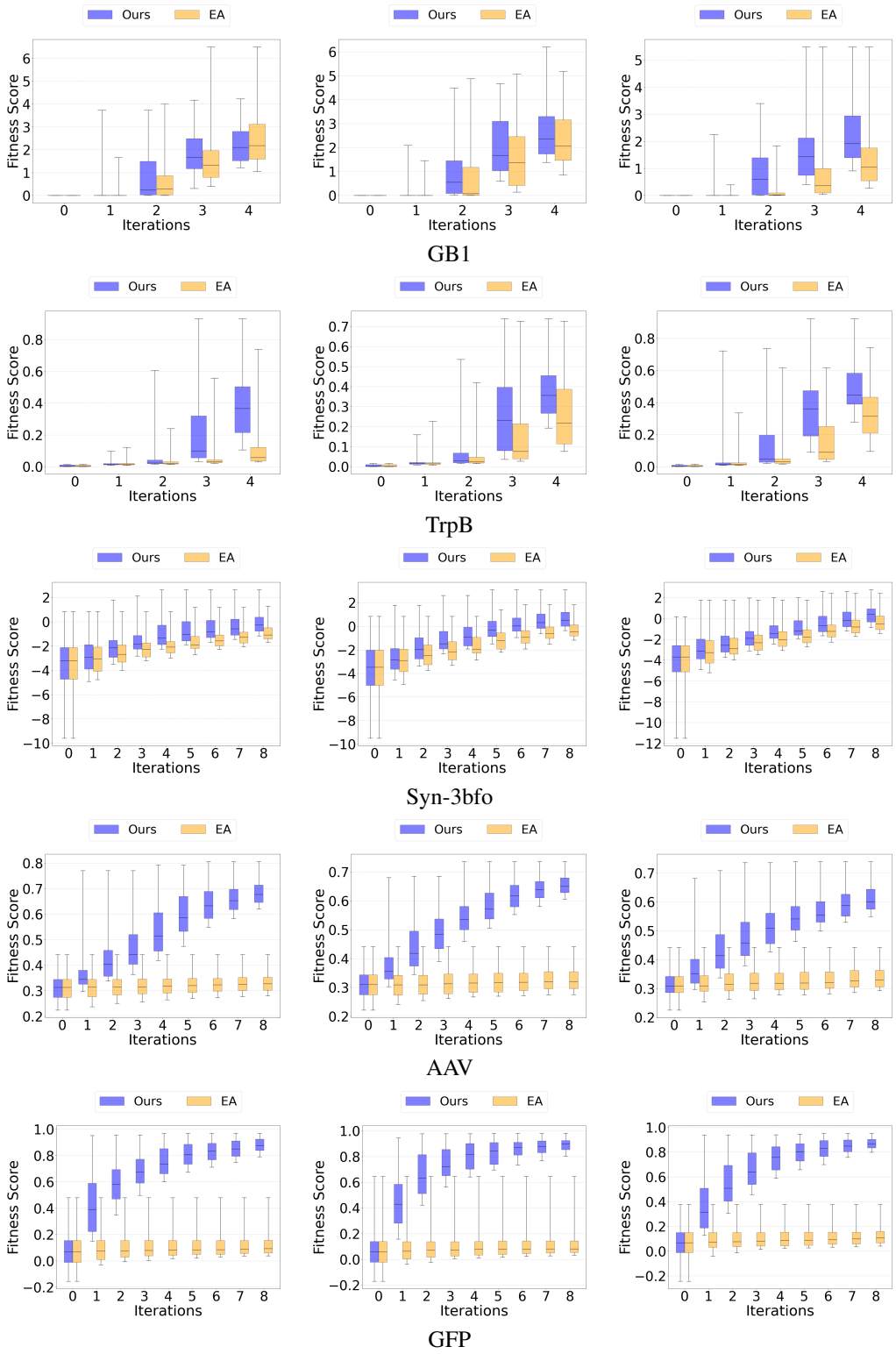


Figure 7: Fitness score across all iterations for five datasets with three random seeds.