Sampling Protein Language Models for Functional Protein Design

Jeremie Theddy Darmawan[*]
Department of Bioinformatics
School of Life Sciences
Indonesia International Institute for Life Sciences

Yarin Gal
OATML Group
Department of Computer Science
University of Oxford

Pascal Notin
OATML Group
Department of Computer Science
University of Oxford

Abstract

Protein language models have emerged as powerful ways to learn complex representations of proteins, thereby improving their performance on several downstream tasks, from structure prediction to fitness prediction, property prediction, homology detection, and more. By learning a distribution over protein sequences, they are also very promising tools for designing novel and functional proteins, with broad applications in healthcare, new material, or sustainability. Given the vastness of the corresponding sample space, efficient exploration methods are critical to the success of protein engineering efforts. However, the methodologies for adequately sampling these models to achieve core protein design objectives remain underexplored and have predominantly leaned on techniques developed for Natural Language Processing. In this work, we first develop a holistic *in silico* protein design evaluation framework, to comprehensively compare different sampling methods. After performing a thorough review of sampling methods for language models, we introduce several sampling strategies tailored to protein design. Lastly, we compare the various strategies on our *in silico* benchmark, investigating the effects of key hyperparameters and highlighting practical guidance on the relative strengths of different methods.

1 Introduction

Proteins possess a wide variety of functions and diversity. Despite sequencing technologies, there is still limited knowledge of the whole protein space. Protein engineering explores this protein space for creating novel proteins, through iterative mutations [2, 45, 44] or by extrapolating sequences [18, 7]. Protein language models (PLMs) have emerged as powerful tools to learn complex distributions over protein sequences, leading to superior performance on a broad range of downstream tasks, As generative models of protein sequences, PLMs are also ideally suited to sample novel functional proteins that have never been observed before and, as such, are very promising tools to support protein engineering workflows. As the protein sample size is massive, there is not only a need for better models but also for efficient sampling methods. While the field has witnessed the emergence of several PLMs for protein design [1, 10, 13, 25, 22], efficient sampling methods have been understudied to date. Prior studies were mostly done with NLP samplings, without an understanding of the effect

[*]Correspondence to jeremie.darmawan@student.i3l.ac.id, pascal.notin@cs.ox.ac.uk

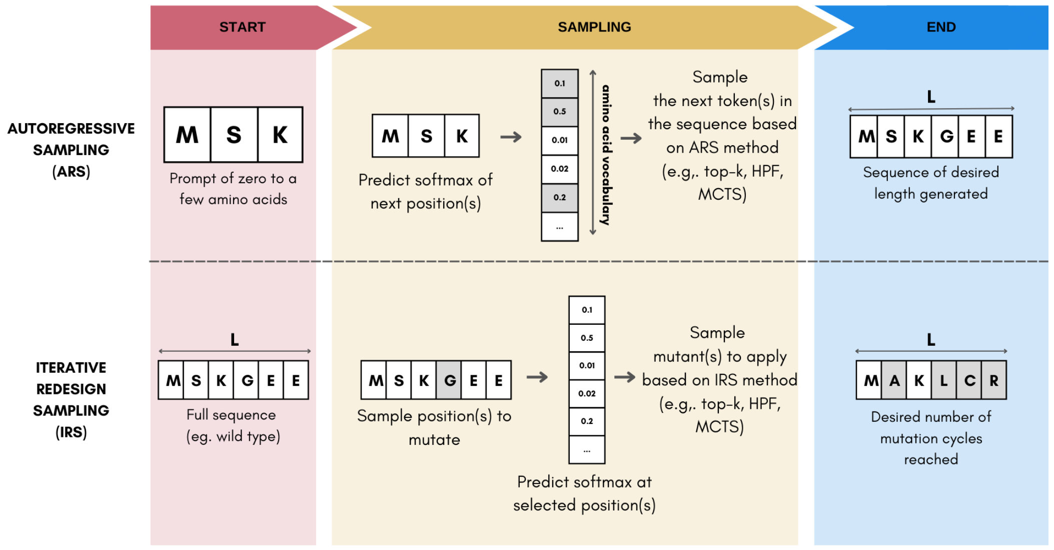

START
SAMPLING
END
AUTOREGRESSIVE SAMPLING (ARS)
M S K
Prompt of zero to a few amino acids
Predict softmax of next position(s)
amino acid vocabulary
Sample the next token(s) in the sequence based on ARS method (e.g,. top-k, HPF, MCTS)
L
M S K G E E
Sequence of desired length generated
ITERATIVE REDESIGN SAMPLING (IRS)
Full sequence (eg. wild type)
Sample position(s) to mutate
Predict softmax at selected position(s)
Sample mutant(s) to apply based on IRS method (e.g,. top-k, HPF, MCTS)
M A K L C R
Desired number of mutation cycles reached

Figure 1: **Taxonomy of protein language sampling methods.** We discriminate between two strategies for sampling PLMs: autoregressive sampling and iterative redesign sampling.

of each method on protein property (§ A). In this work, we explore different sampling methods for PLMs and perform a systematic analysis of the generated sequence based on protein properties. To that end, we explore the effects of sampling methods in protein design through a systematic analysis based on protein properties. We created a robust *in silico* evaluation framework and focused on zero-shot, autoregressive PLMs, due to their versatility (§ 2.1).

Our contributions are as follows:

- We develop 3 efficient sampling strategies tailored to protein design objectives: High-Probability and Quantitative-Function Filter, and a Monte Carlo Tree Search (MCTS) scheme (§ 2).
- We devise a holistic framework for the *in sillico* evaluation of protein engineering methods, spanning core objectives such as functional relevance, sequence diversity, and fitness (§ 3).
- We conduct an in-depth comparison of the various PLM sampling methods, analyze the effect of key sampling hyperparameters on performance and discuss the relative strengths of the different methods depending on design objectives (§ 4).
- We open-source our codebase to provide convenient access to our in silico evaluation framework as well as all sampling methods in a unified interface at `https://github.com/i3LBI19-OATML/sampling-proteins`.

2 Method

2.1 Taxonomy for PLM sampling methods

Strategies for sampling Protein Language Models can be broadly categorized into two groups: Iterative Redesign Sampling (IRS) and AutoRegressive Sampling (ARS), as depicted in Figure 1.

Autoregressive sampling (ARS) generation: Based on an initial prompt of a few amino acid tokens, ARS methods iteratively construct a novel sequence by adding one token at a time, until the desired sequence length is reached or an "end of sentence" token is sampled. Further details are in § B.1. Any of the sampling methods mentioned in § A.1 can be applied to this method. Pseudo-code for the autoregressive pipeline is available in Algorithm S2.

Iterative redesign sampling (IRS) generation: IRS mimics the process of directed evolution [2]. It starts with an initial template sequence from the family of interest (e.g., wild-type sequence) and then iteratively samples one or multiple mutations, until the desired number of mutation rounds (or other metric) has been reached. Further details are available in § B.2. An illustration of the process is provided in Figure S1 while a detailed algorithm pseudocode is available in Algorithm S1.

2.2 Single-Mutant Protein Generation

Single-mutant IRS methods evaluate at each round all possible single amino acid substitutions of the working sequence, score them with the PLM, and sample a mutant with one of the methods described in § A.1, such as random, greedy, top-k, top-p, typical, or mirostat sampling. Single-mutant ARS methods add a single token per iteration, where the sampling process can also be supported by any of the methods described in § A.1. While they are very simple in practice and computationally effective, single-mutant approaches may ignore potentially critical epistatic interactions between multiple mutated positions.

2.3 Multiple-Mutant Protein Generation

Instead of applying a single mutant at each iteration, multiple-mutant strategies apply several mutants simultaneously to better capture epistatic effects [40, 19, 37]. However, this comes at the cost of a significantly larger computational budget given the combinatorial growth of possible mutant sets that can be considered. To that end, we develop three different strategies to make the search of optimal mutant sets tractable, which we describe below and illustrate in Figure S1 and Algorithm S3.

High-Probability Filter (HPF): We first score all possible single amino acid substitutions with the PLM and then filter them to keep only the top 100 single mutants with the highest predicted fitness. We then construct the multiple mutants by only considering combinations involving these 100 mutants, thereby significantly reducing the number of theoretically possible combinations. We then randomly sample N of the resulting multi-mutant set, score them with the PLM and sample the final mutant to apply with the top-k sampling scheme.

Quantitative-Function Filter (QFF): We first generate all possible double-mutant pairs, then sample 100 of them at random. The resulting pairs are then assessed with a separate fitness prediction model (here ProteinBERT [5]). The best N mutants based on these scores are then selected, and scored with the PLM and we sample the final multi-mutant to apply with top-k sampling.

Monte Carlo Tree Search (MCTS): The Upper Confidence bounds applied to Trees (UCT) equation is used to strike the right exploration-exploitation trade-off, with the PLM used to predict node reward values. At each iteration, the node with the highest reward (ie., the most visited) is selected. The scheme can be applied in both ARS and IRS settings. Further details are provided in § D.8. This can be used in both ARS and IRS schemes, for IRS, this can be done with modifications to Algorithm S1 and Algorithm S3.

3 Holistic *in silico* evaluation framework for protein design

When designing functional proteins, there are three high-level criteria that are critical to performance. (1) We want the generated protein sequences to be *relevant* for the design objective, i.e., carry out the function of interest, assessed through their structure. (2) The generated protein sequences ought to *properly execute* the specified function, i.e., have high fitness [19]. To that end, we leveraged a diverse set of highly-performing zero-shot fitness predictors, such as EVmutation [16] (alignment-based Potts model), ESM-1v [23] (alignment-free protein language model), and ESM-MSA [31] (language model across alignment inputs). These models were chosen since their respective inductive biases are different from the autoregressive protein language models with which we generate novel sequences, and they form a representative set of the different types of fitness prediction models [26]. (3) The generated sequences should be *diverse*, i.e., differ from the template wild-type sequence or known natural sequences. We refer to these three design objectives as **Relevance**, **Quality**, and **Diversity** respectively, and crafted two separate metrics for each category (Fig. S2). We define each of the models involved as evaluation metrics in § C

4 Experiments

Through our framework in § 2, we seek to address these questions: (1) What are the benefits of ARS over IRS? (2) What are the benefits of multiple-mutant (HPF and QFF) over single-mutant in IRS? (3) What are the effects of different sampling hyperparameters on generated proteins?

We used Tranception [26] as our PLM guide, applied it to the *Aequorea victoria* (avGFP) protein (details in § D.1), and performed evaluation following the framework discussed in § 3. Within the experiment, different sampling hyperparameters were explored. We included 94 sequences generated by Biswas et al. [4] as an additional baseline. Further details are in § B.3.

4.1 Effects of broad sampling approaches

Effects of using ARS and IRS protein generation are reported in Table S1. IRS method generates proteins that are more similar to the template sequence leading to metrics based on similarity (e.g., TM-score) to be higher and metrics based on distance (e.g., SD and FED) to be lower. ARS approach can generate more diverse proteins at the risk of straying from the desired protein family. As ARS methods produce sequences that are farther away in sequence space relative to the template sequence, they produce diverse sequences that are less fit on average.

4.2 Benefits of multiple-mutant strategies

As there are significant cost implications to multi-mutant scoring, filtering through our proposed strategies between subsequent mutations could be a solution to minimize this cost. Table 1 and Table S3 show results for single and double-mutant IRS and ARS, respectively. Scoring multiple mutants with IRS is computationally costly due to the numerous possible mutations. Filtering the mutants between subsequent scoring and sampling could minimize inference costs through our proposed strategies, HPF and QFF, with a filter value of 96 mutations. An estimate of the reduced cost is from $\sim$60 hours to $\sim$5 minutes.

Metric		**Relevance**		**Quality**		**Diversity**	
		TM-score ↑	**pLDDT ↑**	**Overall Fitness ↑**	**Max. Fitness ↑**	**SD ↑**	**FED ↑**
Single-Mutant	Greedy sampling	0.48	36.7	-0.35	-0.35	0.04	**0.22**
	Beam Search (1)	0.48	36.4	-0.35	-0.35	0.04	**0.22**
Double-Mutant	HPF	0.34	33.0	-1.81	-0.88	0.08	0.17
	QFF	0.33	32.1	-1.71	-0.95	0.08	0.09
	Beam Search (2)	0.39	34.5	-1.90	-1.90	**0.08**	0.15
	MCTS	**0.59**	**44.1**	**0.84**	**0.84**	0.01	0.16

Table 1: Comparison between single-mutant and double-mutant IRS protein generation with Tranception PLM. Only the best sampling methods are included, where HPF is greedy and QFF is top-k. Detailed results for single, double-HPF, and double-QFF IRS are available in Table S7, Table S8, and Table S9. Bold denotes best scores for that metric.

Of the various multi-mutant IRS strategies, only the MCTS-based scheme leads to compelling improvements in the sequence generation, with sequences that are more functionally relevant (eg., TM-Score and pLDDT), higher quality (higher fitness), at the cost of slightly reduced diversity. IRS with MCTS does produce better relevance and quality metrics but compromises on diversity. While ARS approach benefits limitedly by using double-mutants, there is no apparent performance trade-off across the metrics. Further boost of results may be obtained via fine-tuning to target protein families as done in Biswas et al. [4] (Table S2).

4.3 Effects of different sampling hyperparameters

Previous experiments show IRS generation with MCTS produces the best proteins with respect to our evaluation framework (Table S4). Different hyperparameter leads to longer inference cost with no impact on protein property for this sampling method. Detailed results in § G show that sampling methods with larger sampling pools produce lower fitness proteins with limited diversity. In contrast, smaller sampling pools may lead to very distant proteins (due to overcommitting) with higher fitness.

5 Conclusion and Future Directions

This study presents a systematic evaluation of ARS and IRS and different sampling methods on the fitness, diversity, and functional relevance properties. There seems to be an inverse relation between fitness and diversity unless diversity is explicitly accounted for in the generation procedure. Further improvements may be obtained through fine-tuning to target protein families. Methods that can adapt to both properties based on user input are the next frontier. We look forward to future works that may augment this study with evaluations across other protein families, PLMs, varying initial residue prompts, and sampling methods or strategies.

References

[1] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods 2019 16:12*, 16: 1315–1322, 10 2019. ISSN 1548-7105. doi: 10.1038/s41592-019-0598-1.

[2] F. H. Arnold. Directed evolution: Bringing new chemistry to life. *Angewandte Chemie International Edition*, 57:4143–4148, 4 2018. ISSN 1521-3773. doi: 10.1002/ANIE.201708408.

[3] S. Basu, G. S. Ramachandran, N. S. Keskar, and L. R. Varshney. Mirostat: A neural text decoding algorithm that directly controls perplexity. In *International Conference on Learning Representations*, 2021.

[4] S. Biswas, G. Khimulya, E. C. Alley, K. M. Esvelt, and G. M. Church. Low-n protein engineering with data-efficient deep learning. *Nature Methods 2021 18:4*, 18:389–396, 4 2021. ISSN 1548-7105. doi: 10.1038/s41592-021-01100-y.

[5] N. Brandes, D. Ofer, Y. Peleg, N. Rappoport, and M. Linial. Proteinbert: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38:2102–2110, 4 2022. ISSN 1367-4803. doi: 10.1093/BIOINFORMATICS/BTAC020.

[6] R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games*, pages 72–83. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-75538-8_7.

[7] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022. doi: 10.1126/science.add2187.

[8] D. Dowson and B. Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, Sept. 1982. doi: 10.1016/0047-259x(82)900 77-x.

[9] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik, and B. Rost. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:7112–7127, 10 2022. ISSN 19393539. doi: 10.1109/TPAMI.2021.3 095381.

[10] N. Ferruz, S. Schmidt, and B. Höcker. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1), July 2022. doi: 10.1038/s41467-022-32007-7.

[11] M. Fréchet. Sur la distance de deux lois de probabilité. *Annales de l'ISUP*, VI(3):183–198, 1957.

[12] M. Heinzinger, M. Littmann, I. P. W. Sillitoe, N. Bordin, C. A. Orengo, and B. Rost. Contrastive learning on protein embeddings enlightens midnight zone. *NAR Genomics and Bioinformatics*, 4, 2021.

[13] D. Hesslow, N. Zanichelli, P. Notin, I. Poli, and D. Marks. Rita: a study on scaling up generative protein sequence models. *arXiv preprint arXiv:2205.05789*, 2022.

[14] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[15] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. *CEUR Workshop Proceedings*, 2540, 4 2019. ISSN 16130073.

[16] T. A. Hopf, J. B. Ingraham, F. J. Poelwijk, C. P. I. Schärfe, M. Springer, C. Sander, and D. S. Marks. Mutation effects predicted from sequence co-variation. *Nature Biotechnology*, 35(2): 128–135, Jan. 2017. doi: 10.1038/nbt.3769.

[17] T. A. Hopf, A. G. Green, B. Schubert, S. Mersmann, C. P. I. Schärfe, J. B. Ingraham, A. Toth-Petroczy, K. Brock, A. J. Riesselman, P. Palmedo, C. Kang, R. Sheridan, E. J. Draizen, C. Dallago, C. Sander, and D. S. Marks. The EVcouplings python framework for coevolutionary sequence analysis. *Bioinformatics*, 35(9):1582–1584, Oct. 2018. doi: 10.1093/bioinformatics/bty862.

[18] P.-S. Huang, S. E. Boyken, and D. Baker. The coming of age of de novo protein design. *Nature*, 537:320–327, 2016.

[19] S. R. Johnson, X. Fu, S. Viknander, C. Goldin, S. Monaco, A. Zelezniak, and K. K. Yang. Computational scoring and experimental evaluation of enzymes generated by neural networks. *bioRxiv*, page 2023.03.04.531015, 4 2023. doi: 10.1101/2023.03.04.531015.

[20] J. Li, T. Tang, W. X. Zhao, J.-Y. Nie, and J.-R. Wen. Pretrained language models for text generation: A survey, 2022.

[21] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, A. dos Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido, and A. Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637): 1123–1130, Mar. 2023. doi: 10.1126/science.ade2574.

[22] A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos, C. Xiong, Z. Z. Sun, R. Socher, J. S. Fraser, and N. Naik. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology 2023*, pages 1–8, 1 2023. ISSN 1546-1696. doi: 10.1038/s41587-022-01618-2.

[23] J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in Neural Information Processing Systems*, 34:29287–29303, 12 2021.

[24] C. Meister, T. Pimentel, G. Wiher, and R. Cotterell. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121, 2 2022. ISSN 2307387X. doi: 10.1162/tacl_a_00536.

[25] E. Nijkamp, J. Ruffolo, E. N. Weinstein, N. Naik, and A. Madani. Progen2: Exploring the boundaries of protein language models, 2022.

[26] P. Notin, M. Dias, J. Frazer, J. M. Hurtado, A. N. Gomez, D. Marks, and Y. Gal. Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16990–17017. PMLR, 17–23 Jul 2022.

[27] J. Parker and J. Z. Chen. Neural machine translation with monte-carlo tree search, 2020.

[28] D. Peñas-Utrilla and E. Marcos. Identifying well-folded de novo proteins in the new era of accurate structure prediction. *Frontiers in Molecular Biosciences*, 9, 2022.

[29] G. Raghava and G. J. Barton. Quantification of the variation in percentage identity for protein sequence alignments. *BMC Bioinformatics*, 7(1), Sept. 2006. doi: 10.1186/1471-2105-7-415.

[30] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, P. Chen, J. Canny, P. Abbeel, and Y. Song. Evaluating protein transfer learning with tape. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[31] R. M. Rao, J. Liu, R. Verkuil, J. Meier, J. Canny, P. Abbeel, T. Sercu, and A. Rives. Msa transformer. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8844–8856. PMLR, 18–24 Jul 2021.

[32] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 118:e2016239118, 4 2021. ISSN 10916490. doi: 10.1073/PNAS.2016239118/SUPPL_FILE/PNAS.2016239118.SAPP.PDF.

[33] K. S. Sarkisyan, D. A. Bolotin, M. V. Meer, D. R. Usmanova, A. S. Mishin, G. V. Sharonov, D. N. Ivankov, N. G. Bozhanova, M. S. Baranov, O. Soylemez, N. S. Bogatyreva, P. K. Vlasov, E. S. Egorov, M. D. Logacheva, A. S. Kondrashov, D. M. Chudakov, E. V. Putintseva, I. Z. Mamedov, D. S. Tawfik, K. A. Lukyanov, and F. A. Kondrashov. Local fitness landscape of the green fluorescent protein. *Nature 2015 533:7603*, 533:397–401, 5 2016. ISSN 1476-4687. doi: 10.1038/nature17995.

[34] M. Seitzer. pytorch-fid: FID Score for PyTorch. `https://github.com/mseitzer/pytorch-fid`, August 2020. Version 0.3.0.

[35] D. Sgarbossa, U. Lupo, and A.-F. Bitbol. Generative power of a protein language model trained on multiple sequence alignments. *eLife*, 12:e79854, feb 2023. ISSN 2050-084X. doi: 10.7554/eLife.79854.

[36] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, Oct. 2017. doi: 10.1038/nature24270.

[37] T. N. Starr and J. W. Thornton. Epistasis in protein evolution. *Protein Science*, 25(7):1204–1218, Feb. 2016. doi: 10.1002/pro.2897.

[38] B. E. Suzek, H. Huang, P. McGarvey, R. Mazumder, and C. H. Wu. UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, 23(10):1282–1288, Mar. 2007. doi: 10.1093/bioinformatics/btm098.

[39] R. Y. Tsien. The green fluorescent protein. *Annual Review of Biochemistry*, 67(1):509–544, 1998. doi: 10.1146/annurev.biochem.67.1.509. PMID: 9759496.

[40] V. Upadhyay, C. Patrick, A. Lucas, and K. M. Mallela. Convergent evolution of multiple mutations improves the viral fitness of sars-cov-2 variants by balancing positive and negative selection. *Biochemistry*, 61:963–980, 6 2022. ISSN 15204995. doi: 10.1021/ACS.BIOCHEM.2C00132/ASSET/IMAGES/MEDIUM/BI2C00132_0012.GIF.

[41] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[42] Y. Wang, H. Tang, L. Huang, L. Pan, L. Yang, H. Yang, F. Mu, and M. Yang. Self-play reinforcement learning guides protein engineering. *Nature Machine Intelligence*, July 2023. doi: 10.1038/s42256-023-00691-9.

[43] L. N. Wasserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Probl. Peredachi Inf.*, 5:47–52, 1969.

[44] B. J. Wittmann, K. E. Johnston, Z. Wu, and F. H. Arnold. Advances in machine learning for directed evolution. *Current Opinion in Structural Biology*, 69:11–18, Aug. 2021. doi: 10.1016/j.sbi.2021.01.008.

[45] K. K. Yang, Z. Wu, and F. H. Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature Methods*, 16(8):687–694, July 2019. doi: 10.1038/s41592-019-0496-6.

[46] Y. Zhang and J. Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004. doi: https://doi.org/10.1002/prot.20264.

Appendix

A Background and related work

A.1 Sampling Language Models

Language models are generative models seeking to approximate a distribution over sequential inputs. Once trained, we can then sample from these generative models to create new objects that may not have been part of the initial training data. Since the creation of novel sequential objects may be driven by various objectives (e.g., creativity, coherence, fluency, factual information), a wide range of sampling strategies has been developed to promote certain characteristics of generated sequences [20]. The most simple strategy is **random sampling**, which samples items based on their probabilities as provided by the softmax of output logits u_i (Equation 1), divided by the smoothing temperature parameter T which controls the flatness of the distribution.

$$P(x_i|x_{1:i-1}) = \frac{exp(u_i/T)}{\sum_j exp(u_j/T)} \tag{1}$$

In order to avoid sampling items with extremely low probability under the language model, **top-k sampling** selects the subset $V^{(k)}$ of the k elements with highest probabilities, sets the probabilities of the other items to zero, and renormalizes probabilities for the top-k items as expressed in Equation 2. **Greedy sampling** is an extreme version of top-k sampling that only takes the item with the highest (top-1) probability in the distribution.

$$P^k(x|x_{1:i-1}) = \begin{cases} P(x|x_{1:i-1})/p', & \text{if } x \in V^{(k)}. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where $x_{1:i-1}$ corresponds to sequence items previously generated, and $p' = \sum_{x \in V^{(k)}} P(x|x_{1:i-1})$ is the probability renormalization factor.

In practice, using a fixed-size k may be suboptimal when the flatness of the distribution varies significantly across generated tokens in the sequence: in some instances, relevant items will be censored, while in others certain low-likelihood items may be sampled. To address this problem, top-p sampling [15] (or nucleus sampling) first computes the corresponding cumulative distribution and censors it as soon as the CDF exceeds p, giving rise to the set $V^{(p)}$ of all non-censored items. Mathematically, this leads to a censored probability analogous to the one in Equation 2, except for the renormalization factor $p' = \sum_{x \in V^{(p)}} P(x|x_{1:i-1})$.

Beam search views a sequential sampling process as a search problem in which one explores the most promising items at each step in a tree-like structure, maintaining a fixed number of possible solutions at each depth called a "beam width." Other techniques placing a greater emphasis on human expectations and preferences were also developed for NLP. **Typical sampling** by Meister et al. [24] was inspired by information theory on the premise that texts generated from language models are not aligned with the expected information content of the user. Hence, it attempts to minimize the difference between the generated text's information content and the expected information content. Basu et al. [3] identified that humans prefer generated texts with a certain level of perplexity that is neither too low nor too high, which led to the development of **Mirostat sampling**. This sampling method will self-adapt and guide the text decoding toward a predetermined perplexity value without any model finetuning. Therefore, it provides better maintenance over perplexity levels in NLP-generated texts. Language model decoding can also be considered as a sequence of decisions, similar to that in a game, where the current generated tokens (state) will influence the next tokens (moves) being generated in order to achieve certain objectives. **Monte Carlo Tree Search (MCTS)** was initially coined by Coulom [6], and later used to identify the best future moves for the game of Go [36]. It was initially developed as a decision-making, search algorithm, it has then been used in machine translation and, more recently, for protein design [27, 42]. Its mechanism involves the use of a UCT to manage the exploration-exploitation decision as well as the backpropagation of information regarding each node visit and reward. As a policy-value network, it may use search results to improve itself and provide guidance for more accurate updates compared to deterministic sampling methods. See Appendix D for more details on these various sampling methods.

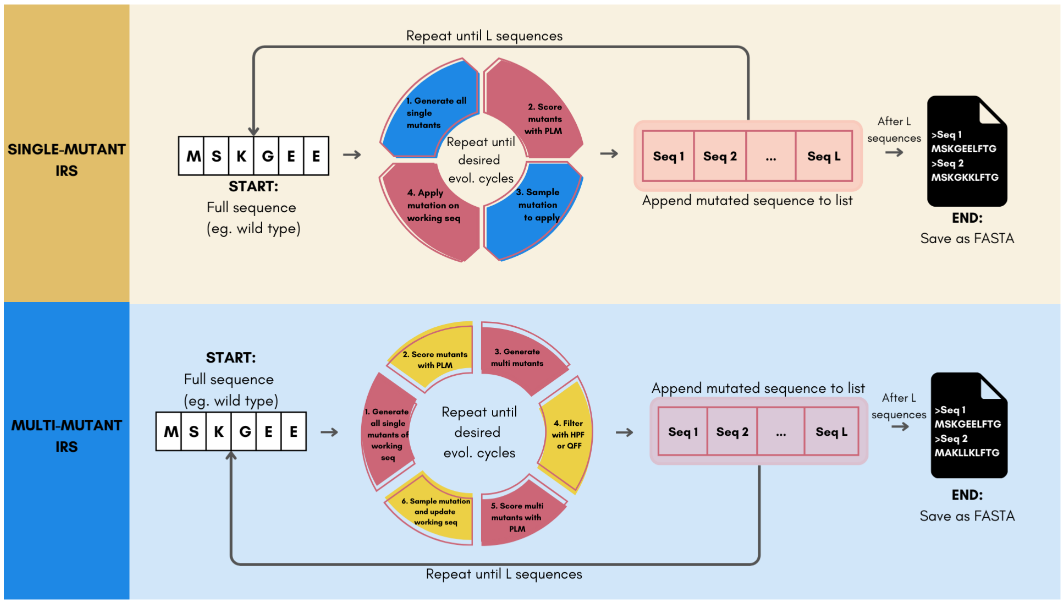

SINGLE-MUTANT IRS
Repeat until L sequences
M S K G E E
START:
Full sequence
(eg. wild type)
1. Generate all single mutants
2. Score mutants with PLM
3. Sample mutation to apply
4. Apply mutation on working seq
Repeat until desired evol. cycles
Seq 1
Seq 2
...
Seq L
Append mutated sequence to list
After L sequences
>Seq 1
MSKGEELFTG
>Seq 2
MSKGKKLFTG
END:
Save as FASTA
MULTI-MUTANT IRS
START:
Full sequence
(eg. wild type)
M S K G E E
1. Generate all single mutants of working seq
2. Score mutants with PLM
3. Generate multi mutants
4. Filter with HPF or QFF
5. Score multi mutants with PLM
6. Sample mutation and update working seq
Repeat until desired evol. cycles
Append mutated sequence to list
Seq 1
Seq 2
...
Seq L
After L sequences
>Seq 1
MSKGEELFTG
>Seq 2
MAKLLKLFTG
END:
Save as FASTA
Repeat until L sequences

Figure S1: **High-level single and multi-mutant IRS algorithms for generating novel protein sequences**. Detailed algorithms are provided in supplementary (Algorithms S1 and S3).

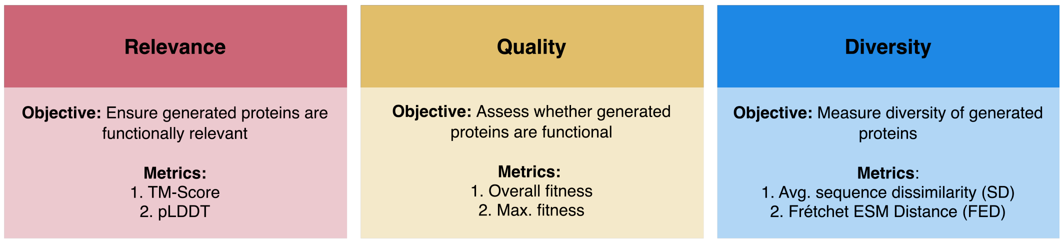
A.2 Protein Language Models

The majority of protein language models can be classified into two broad categories: autoregressive (AR) and masked-language models (MLM) although other learning paradigms such as seq2seq [9] or FIM [12] have also been explored. Unirep [1] was the first PLM trained autoregressively across unaligned protein sequences across protein families. With the improvements seen in Transformers [41] for NLP, masked-language models such as ProteinBERT, ESM-1b, TAPE-BERT, and MSA Transformer [5, 32, 30, 31], as well as autoregressive models, such as Tranception, RITA, and ProtGPT2 [26, 13, 10]. These PLMs have been shown to achieve remarkable performance across diverse tasks, such as zero-shot fitness prediction [23, 26]. For protein design, prior PLM works have relied on the simple sampling methods discussed in § A.1, such as top-k, top-p, random, or greedy sampling [35, 4, 25, 22, 10]. Only recently have Wang et al. [42] leveraged a MCTS-based approach to carry out a 35 residue-restricted iterative protein design. Sgarbossa et al. [35] developed a method that iteratively applies random masking of residues in the protein sequence and fills them greedily based on MSA Transformer [31] predictions.

B Protein Design Approaches

B.1 Autoregressive sampling (ARS) generation

This approach can be used with initial-residue prompting or unprompted generation. As previously reported in Nijkamp et al. [25], prompted generation allows for a more targeted generation, meaning that generated protein sequences are closer in identity to a certain target protein sequence. Hence, when given an initial residue prompt, the pipeline will extrapolate the AA sequence while keeping certain information about the initial residue throughout the generation process. Generating protein sequences using this method is an iterative process that will last until a desired sequence length is achieved. At each round, we add one (or more) of the 20 possible amino acids to the working sequence and a zero-shot PLM will then generate a fitness score for each of them. Then, a predefined sampling method will select the next residue using the scores as guidance. We could generate multiple sequences by repeating this process.



Relevance

Objective: Ensure generated proteins are functionally relevant

Metrics:
1. TM-Score
2. pLDDT

Quality

Objective: Assess whether generated proteins are functional

Metrics:
1. Overall fitness
2. Max. fitness

Diversity

Objective: Measure diversity of generated proteins

Metrics:
1. Avg. sequence dissimilarity (SD)
2. Frétchet ESM Distance (FED)

Figure S2: **Our holistic *in silico* evaluation framework for functional protein design revolves around three core performance criteria: relevance, quality, and diversity**. For each criterion, two complementary metrics enable a thorough comparison of the relative benefits of the various protein language model sampling methods.

B.2 Iterative redesign sampling (IRS)

We used an iterative redesign protein design pipeline, that takes in a template protein sequence and generates all possible single mutations of the sequence. For multiple mutations, we could either introduce subsequent mutations after the first one or directly introduce two or more mutations at each cycle. All the mutations could then be scored using PLMs to guide the process. This pipeline only considers substitution-based mutations and is similar to the method used by Sgarbossa et al. [35], Johnson et al. [19]. All mutant scores are sampled using a predefined sampling method which will then be used to mutate the sequence. This mutation cycle may be repeated multiple times to introduce multiple single mutations. Any sampling strategies that are mentioned in Section A.1 can be used along with this method. To generate multiple protein sequences, we repeated this pipeline several times with the same initial template sequence. An illustration of the code for the single-mutant IRS is provided in Figure S1 while a detailed algorithm pseudocode is available in Algorithm S1.

B.3 Experimental Protocol

Generation Protocol: Based on its superior performance to ESM-1v and ESM-MSA, we used Tranception to guide our design process [23, 31, 26]. Each sampling method and hyperparameter were used to generate 100 sequences. Evolution cycles for IRS is set at 10 while the desired length for ATS is equal to the length of the protein wild-type.

Evalutation Protocol: We applied the relevance, quality, and diversity evaluations contained in our *in silico* evaluation framework discussed in § 3 to provide us with the properties of the proteins generated. Both target and reference sequences in FASTA format are required. Target and reference sequences correspond to the generated sequences and naturally occurring sequences, respectively.

Data: Generation using IRS and ARS methods requires a certain length of a protein sequence. Particularly for QFF, additional DMS data for model fine-tuning is required. Proteins that have extensive DMS reports are valid options, including the ones in the ProteinGym benchmarks [26].

Selected protein family: We used *Aequorea victoria* (avGFP) protein throughout our experiments. Details of the protein are in Appendix D. We took the whole sequence for the IRS approach and only the first 10 residues for the ARS approach. References were obtained from the ProteinGym MSA [26].

Sampling & Hyperparameters: Sampling methods that are part of this experiment include Random, Greedy, Top-k, Top-p, Mirostat, and Typical sampling as well as Beam Search and MCTS. Hyperparameters for each are as follows: Mirostat sampling at the best-reported threshold of 3.0 [3], Top-p sampling from 0.1 to 1.0, Typical sampling from 0.1 to 0.95, and Top-k sampling with k values of 2, 3, 5, 10, 15, 20, 30, 40, and 50. Hyperparameter values for Beam search and MCTS are 1 and 2.

C Evaluation Metrics

1. **TM-score:** Assesses the topological similarity between the predicted structure of the generated sequences and the structure of the template wild type sequence [46]. The final score is the average of the pairwise TM-scores. Structures were predicted with ESMFold [21].

Ideally, the structure of generated sequences is as close to that of the template sequence as possible to ensure functional relevance.

2. **pLDDT:** Quantifies the confidence of a structure prediction model (here ESMFold [21]), averaged over the entire sequence. Higher pLDDT has been observed to correlate with higher functional designs, e.g., for the design of binders to specific targets [28].
3. **Avg. Fitness:** Reports the average fitness gained by the generated sequences to that of the template wild-type sequence. In practice, we compute this metric for the three aforementioned fitness predictors and average the corresponding scores.
4. **Max. fitness:** Evaluates the ability of the sampling procedure to generate sequences that significantly enhance the fitness of the template wild-type sequence. For each fitness predictor, we compute the maximum fitness value in the generated set and average the corresponding score across models. To normalize across models, ratios are standard-scaled before averaging based on the distribution of fitness scores observed over a reference set (e.g., a prior DMS experiment).
5. **Avg. Sequence Dissimilarity (SD):** Reports the average primary sequence dissimilarity between generated sequences and the template wild-type sequence (i.e., the proportion of amino acids that are different). Ideally, we generate novel sequences that are distant in sequence space from the initial wild-type sequence.
6. **Fréchet ESM Distance (FED):** Assesses the extent to which the generated sequences are distant in embedding space to known natural sequences (e.g., sequences in an MSA for that protein family). It can be seen as a protein-related analog of the Fréchet Inception Distance (FID), initially introduced to evaluate GAN-generated images [29, 14, 34]. It leverages the embeddings from the penultimate layer of ESM-1v [23], and computes the Fréchet distance between the embeddings of the generated sequences and the sequences in an MSA for that protein family.

D Glossary

D.1 avGFP Protein

The jellyfish *Aequorea victoria* has a particular protein that exhibits a green fluorescent when exposed to light [39]. Our experiments were conducted using the *Aequorea victoria* green fluorescent protein (avGFP) data by Sarkisyan et al. [33]. It is also one of the widely used biophysical property benchmarks and used in other protein generation attempts by Biswas et al. [4].

D.2 Top-K Sampling

The sorting of probabilities (or scores) in descending order and zero-masking the values that are beyond the *k* rank. It truncates unreliable, lower probability tokens and considers only the highest *k* probable tokens.

D.3 Greedy Sampling

A variant of top-k sampling that only considers the highest (top-1) probable token.

D.4 Beam Search

Beam search is a greedy, heuristic search algorithm that looks ahead into the next *N* tokens. It considers the best combination of all proceeding tokens ahead of the current token. Although computationally costly, this approach would generally generate better results over top-k sampling as it takes into account the next *N* combinations that might be missed when selecting individual tokens for each position.

D.5 Top-p Sampling

As a solution to the top-k sampling and beam search that is prone to the boredom trap, top-p sampling (or nucleus sampling) was developed Holtzman et al. [15]. It samples from a pool of the smallest possible set of tokens in which the cumulative probability exceeds the probability p, also called the nucleus. Compared to top-k sampling, this approach has a dynamically adjusting sampling pool according to the predicted probability distribution of the language model.

D.6 Mirostat Sampling

Based on reports that text quality is most desirable at certain likelihood ranges, mirostat sampling was developed to keep the generated text within a predetermined perplexity value [3]. Therefore, high-quality text could be obtained without any fine-tuning. This approach is also able to prevent the generated text from the boredom trap (repetitions) as well as the confusion trap (incoherence). The main notation for this sampling is $k = (\hat{\epsilon}2^{\mu}/(1 - N^{-\hat{\epsilon}}))^{(1/\hat{s})}$, where $\hat{s} = \sum_{i=1}^{m-1} t_i b_i / \sum_{i=1}^{m-1} t_i^2$ and $\mu = \mu - \eta e$ in which, e is the difference between the observed surprise and τ

D.7 Typical Sampling

Inspired by the psycholinguistic theory of human communication, Meister et al. [24] formally defines a criterion for text generation that minimizes the next generated token information content as close as possible to the model's conditional entropy (or expected information content). Using this information theory foundation, typical sampling efficiently enforces this criterion upon text generation. As written in the original paper, typical sampling is a minimization optimization problem of the following subset:

$$\sum_{y \epsilon C'(y_{<t})} |H(Y_t | Y_{<t} = y_{<t}) + logp(y | y_{<t})| \tag{3}$$

which is subject to $\sum_{y \epsilon C(y_{<t})} |p(y|y_{<t} \leq \tau$.

D.8 Monte Carlo Tree Search (MCTS)

Within the MCTS mechanism, the Upper Confidence bounds applied to Trees (UCT) equation determine the step to perform an exploration-exploitation decision. Simulation rewards for each node are obtained through PLM scoring. The equation can be defined as

$$UCT(node_i) = \frac{w_i}{s_i} + c\sqrt{\frac{\ln S_p}{S_i}} \tag{4}$$

where w_i is the total number of simulations of that node that resulted in rewards, s_i is the total number of simulations of that node, s_p is the total number of simulations of the parent node, and c is the exploration parameter.

D.9 Fréchet distance

The objective of generative models is to imitate the original data as best as possible. Hence, the distance between the generated data $p_w(.)$ and $p(.)$, or also called the Fréchet distance, can be a measure of generation quality [11, 14]. This measure is also known as the Wasserstein-2 distance [43]. The Fréchet distance $d(.,.)$ is calculated between the $p(.)$ Gaussian with mean (m, C) and the $p_w(.)$ Gaussian with mean (m_w, C_w). Since we only modified the Inception model used in Heusel et al. [14] into ESM-1v that is suitable for protein sequences, the "Fréchet Inception Distance" (FID) equation [8] still applies:

$$d^2((m, C), (m_w, C_w)) = (||m - m_w||)_2^2 + Tr(C + C_w - 2(CC_w)^{\frac{1}{2}}) \tag{5}$$

E EVmutation Initialization

Initializing the EVmutation [16] model requires two steps: (1) generating deep multiple sequence alignments (MSA) and (2) generating the model parameters. To generate the MSA, we used the official EVcouplings [17] tool, which can be found at `https://evcouplings.org/`, and search against the UniRef100 [38] database. We opted to use the v1 tool as it seems to produce deeper alignments than the v2. Advanced configurations on the tool were left to the default settings except for the position gap threshold and sequence gap threshold which were set to 1.0, as we mainly work with substitutions. The run that has the highest quality score was selected for generating the model parameters. We used the recommended plmc [16] commands to accomplish this with L2-regularization on the couplings set at 47.4, according to their formula.

F ProteinBERT fine-tuning

The pre-trained ProteinBERT model and fine-tuning protocol were provided by Brandes et al. [5]. The process of fine-tuning for biophysical function only requires experimental labels (quantitative function) and following their protocol. All the layers of pretrained model are initially frozen and a fully connected layer is added on top of the model. This state was trained for 40 epochs. After this, all the layers are unfrozen and trained for another 40 epochs. Hence, fine-tuning was done on the embeddings, and no additional conditioning on GO annotations. Learning rate reduction on plateau and early topping callback was applied in the fine-tuning protocol.

G Tables

Method	**Relevance**		**Quality**		**Diversity**	
	TM-score ↑	**pLDDT ↑**	**Avg. Fitness ↑**	**Max. Fitness ↑**	**SD ↑**	**FED ↑**
ARS	0.31	**38.4**	-16.11	-14.53	**0.52**	**7.12**
IRS	**0.40**	34.8	**-0.49**	**0.01**	0.04	0.06

Table S1: Comparison of ARS and IRS methods on the different aspects of the evaluation framework. Bold denotes best scores.

Metric		**Relevance**		**Quality**		**Diversity**	
		TM-score ↑	**pLDDT ↑**	**Overall Fitness ↑**	**Max. Fitness ↑**	**SD ↑**	**FED ↑**
Single-Mutant	Greedy sampling	0.48	36.7	-0.35	-0.35	0.04	**0.22**
	Beam Search (1)	0.48	36.4	-0.35	-0.35	0.04	**0.22**
Double-Mutant	HPF	0.34	33.0	-1.81	-0.88	0.08	0.17
	QFF	0.33	32.1	-1.71	-0.95	0.08	0.09
	Beam Search (2)	0.39	34.5	-1.90	-1.90	**0.08**	0.15
	MCTS	0.59	**44.1**	**0.84**	0.84	0.01	0.16
Biswas et al. [4] (Supervised)		**0.60**	44.0	0.27	**1.35**	0.01	0.00

Table S2: Comparison between single-mutant and double-mutant IRS protein generation with Tranception PLM. Only the best sampling methods are included, where HPF is greedy and QFF is top-k. Detailed results for single, double-HPF, and double-QFF IRS are available in Table S7, Table S8, and Table S9. Bold denotes best scores for that metric.

Metric		**Relevance**		**Quality**		**Diversity**	
		TM-score ↑	**pLDDT ↑**	**Avg. Fitness ↑**	**Max. Fitness ↑**	**SD ↑**	**FED ↑**
Single-Mutant	Mirostat sampling	0.14	41.8	-14.68	-14.68	0.56	61.84
	Beam Search (1)	0.10	75.9	-14.96	-14.96	0.54	4.41
	MCTS	0.14	41.8	-14.68	-14.68	0.56	61.84
Double-Mutant	Greedy sampling	0.13	**76.4**	-14.54	-14.54	0.53	4.46
	Beam Search (2)	0.07	74.5	**-14.15**	**-14.15**	0.53	**78.19**
	MCTS	**0.33**	36.9	-18.44	-18.44	**0.58**	1.87

Table S3: Comparison between single-mutant and double-mutant ARS protein generation. Best approaches for NLP-based samplings are presented. Other results for single and double-mutant ARS are available in Table S5 and Table S6, respectively. Bold denotes best scores.

Sampling Method	Hyper. Value	Relevance		Quality		Diversity		Time (s) ↓
		TM-score ↑	pLDDT ↑	Avg. Fitness ↑	Max. Fitness ↑	SD ↑	FED ↑	
MCTS	1	**0.59**	**44.1**	**0.84**	**0.84**	**0.01**	**0.16**	**1660.26**
	2	**0.59**	**44.1**	**0.84**	**0.84**	**0.01**	**0.16**	5467.85

Table S4: Effect of hyperparameters on ARS double-mutant protein generation with MCTS sampling

Sampling Method	Hyper. Value	Relevance		Quality		Diversity	
		TM-score ↑	pLDDT ↑	Avg. Fitness ↑	Max. Fitness ↑	SD ↑	FED ↑
Random		0.33	33.5	-16.78	-15.36	0.52	1.19
Greedy		0.08	**75.9**	-14.96	-14.96	0.54	4.41
Top-k	2	0.21	69.4	-15.17	-13.81	0.52	27.76
	3	0.24	62.6	-15.10	-13.71	0.52	23.02
	5	0.30	55.7	-14.93	**-13.13**	0.52	17.17
	10	0.31	41.5	-15.35	-13.14	0.51	1.24
	15	0.33	36.1	-15.64	-13.22	0.51	1.21
	20	0.33	33.0	-16.72	-14.98	0.52	1.01
Top-p	0.1	0.34	33.7	-16.67	-14.72	0.52	1.01
	0.2	0.34	33.1	-16.75	-15.34	0.52	1.25
	0.3	0.34	33.4	-16.68	-15.14	0.52	1.17
	0.4	0.34	33.7	-16.71	-14.86	0.52	1.08
	0.5	**0.35**	33.5	-16.62	-15.14	0.52	0.95
	0.6	0.33	33.6	-16.68	-15.12	0.52	1.06
	0.7	0.33	33.3	-16.68	-15.08	0.52	0.96
	0.8	0.34	33.0	-16.63	-15.26	0.52	0.98
	0.9	0.33	33.1	-16.70	-14.95	0.53	0.94
	1.0	0.34	33.1	-16.82	-15.68	0.53	1.04
Typical	0.1	0.30	32.4	-15.91	-14.12	0.51	10.74
	0.2	0.31	32.9	-16.11	-14.32	0.52	10.34
	0.3	0.32	33.0	-16.12	-14.42	0.52	9.09
	0.4	0.31	32.7	-16.11	-14.15	0.52	8.13
	0.5	0.32	33.0	-16.15	-14.04	0.52	6.38
	0.6	0.33	32.8	-16.11	-14.05	0.51	4.87
	0.7	0.32	33.7	-15.99	-14.13	0.52	3.68
	0.8	0.34	33.3	-16.15	-14.74	0.52	2.29
	0.9	0.34	33.1	-16.19	-14.12	0.52	1.19
	0.95	0.33	33.5	-16.21	-15.01	0.52	0.55
Mirostat	3.0	0.14	41.8	**-14.68**	-14.68	**0.56**	**61.84**
MCTS	1	0.14	41.8	**-14.68**	-14.68	**0.56**	**61.84**
	2	0.14	41.8	**-14.68**	-14.68	**0.56**	**61.84**
Beam Search	1	0.10	75.9	-14.96	-14.96	0.54	4.41

Table S5: Performance of the different sampling methods and hyperparameter values on single-ARS protein generation. Bold scores denote best values.

Sampling Method	Hyper. Value	Relevance		Quality		Diversity	
		TM-score ↑	**pLDDT ↑**	**Avg. Fitness ↑**	**Max. Fitness ↑**	**SD ↑**	**FED ↑**
Random		0.34	32.8	-16.57	-14.76	0.52	1.34
Greedy		0.13	**76.4**	-14.54	-14.54	0.53	4.46
Top-k	2	0.20	68.1	-14.98	**-12.97**	0.52	37.52
	3	0.21	64.0	-15.00	-13.57	0.52	19.79
	5	0.23	62.1	-14.91	-13.21	0.52	21.49
	10	0.27	59.7	-14.86	-13.55	0.51	22.37
	15	0.29	58.2	-14.95	-13.54	0.51	19.90
	20	0.29	57.3	-15.01	-13.09	0.51	16.07
	30	0.30	56.7	-14.97	-13.65	0.51	11.95
	40	0.30	55.3	-15.06	-13.48	0.51	10.22
	50	0.30	53.6	-15.02	-13.31	0.51	7.63
Top-p	0.1	0.34	33.4	-16.66	-14.41	0.52	1.10
	0.2	0.34	33.4	-16.67	-14.49	0.52	1.23
	0.3	0.34	33.2	-16.50	-14.68	0.53	1.04
	0.4	0.34	33.3	-16.64	-15.20	0.52	1.01
	0.5	**0.35**	33.1	-16.67	-14.96	0.52	1.05
	0.6	0.34	33.2	-16.73	-15.15	0.52	1.05
	0.7	0.34	33.3	-16.66	-15.10	0.52	1.16
	0.8	0.33	33.4	-16.80	-15.06	0.52	1.02
	0.9	0.34	32.9	-16.61	-14.99	0.52	1.04
	1.0	0.34	33.0	-16.75	-14.62	0.52	1.39
Typical	0.1	0.34	33.1	-16.32	-14.59	0.52	4.09
	0.2	0.33	32.8	-16.39	-14.63	0.51	3.46
	0.3	0.34	32.9	-16.48	-14.95	0.52	3.12
	0.4	0.33	32.8	-16.30	-14.88	0.52	2.51
	0.5	0.34	32.8	-16.47	-14.51	0.52	2.23
	0.6	0.34	33.1	-16.38	-13.87	0.52	1.82
	0.7	0.34	32.8	-16.25	-14.99	0.52	1.26
	0.8	**0.35**	32.7	-16.45	-14.97	0.52	1.25
	0.9	0.33	33.3	-16.52	-14.76	0.52	1.03
	0.95	0.34	33.0	-16.53	-14.88	0.52	1.07
Mirostat	3.0	0.33	36.9	-18.44	-18.44	**0.58**	1.87
MCTS	1	0.33	36.9	-18.44	-18.44	**0.58**	1.87
	2	0.33	36.9	-18.44	-18.44	**0.58**	1.87
Beam Search	2	0.07	74.5	**-14.15**	-14.15	0.53	**78.19**

Table S6: Performance of the different sampling methods and hyperparameter values on double-ARS protein generation. Bold scores denote best values.

Sampling Method	Hyper. Value	Relevance		Quality		Diversity	
		TM-score ↑	pLDDT ↑	Avg. Fitness ↑	Max. Fitness ↑	SD ↑	FED ↑
Random		0.44	35.8	-0.44	-0.02	0.04	0.04
Greedy		0.48	36.7	-0.35	-0.35	0.04	**0.22**
Top-k	2	0.39	33.1	-0.48	-0.20	0.04	0.08
	3	0.41	34.9	-0.46	-0.08	0.04	0.10
	5	0.36	33.9	-0.44	-0.12	0.04	0.10
	10	0.39	35.0	-0.43	-0.07	0.04	0.15
	15	0.36	33.9	-0.43	-0.01	0.04	0.19
	20	0.37	34.4	-0.41	0.13	0.04	0.19
	30	0.38	34.5	-0.36	0.07	0.04	0.15
	40	0.37	34.7	-0.40	0.10	0.04	0.13
	50	0.38	35.0	-0.35	0.15	0.04	0.18
Top-p	0.1	0.35	33.3	-0.71	-0.26	0.04	0.03
	0.2	0.37	33.6	-0.71	-0.23	0.04	0.02
	0.3	0.36	33.2	-0.68	-0.07	0.04	0.02
	0.4	0.39	33.6	-0.68	0.01	0.04	0.01
	0.5	0.38	34.4	-0.69	-0.23	0.04	0.03
	0.6	0.35	33.3	-0.64	-0.11	0.04	0.01
	0.7	0.38	33.6	-0.65	-0.07	0.04	0.01
	0.8	0.36	33.0	-0.65	-0.21	0.04	0.01
	0.9	0.38	34.0	-0.64	0.20	0.04	0.01
	1.0	0.37	33.1	-0.63	-0.02	0.04	0.02
Typical	0.1	0.44	36.6	-0.38	0.25	0.04	0.04
	0.2	0.45	36.8	-0.41	0.03	0.04	0.02
	0.3	0.46	37.4	-0.40	0.18	0.04	0.01
	0.4	0.43	36.1	-0.39	0.14	0.04	0.02
	0.5	0.43	35.9	-0.39	0.08	0.04	0.02
	0.6	0.45	35.6	-0.41	0.22	0.04	0.02
	0.7	0.44	35.6	-0.42	0.12	0.04	0.04
	0.8	0.40	34.8	-0.40	0.08	0.04	0.03
	0.9	0.44	36.6	-0.41	0.17	0.04	0.01
	0.95	0.44	36.2	-0.41	0.09	0.04	0.02
Mirostat	3	0.45	36.6	-0.38	0.27	0.04	0.04
MCTS	1	**0.59**	**44.1**	**0.84**	0.84	0.01	0.16
	2	**0.59**	**44.1**	**0.84**	0.84	0.01	0.16
Beam Search	1	0.48	36.4	-0.35	-0.35	0.04	**0.22**
	2	0.39	34.5	-1.90	-1.90	**0.08**	0.15

Table S7: Performance of the different sampling methods and hyperparameter values on single-IRS protein generation. Bold scores denote best values.

Sampling Method	Hyper. Value	Relevance		Quality		Diversity	
		TM-score ↑	pLDDT ↑	Avg. Fitness ↑	Max. Fitness ↑	SD ↑	FED ↑
Random		0.33	32.7	-1.78	-0.95	0.08	0.10
Greedy		0.32	33.5	-1.86	-1.12	0.08	0.20
Top-k	2	0.32	**33.7**	-1.82	-1.32	**0.08**	**0.21**
	3	0.33	33.3	-1.86	-1.13	**0.08**	0.09
	5	0.32	33.1	-1.82	-1.21	0.08	0.17
	10	0.32	33.3	-1.80	-1.17	0.08	0.20
	15	0.33	33.7	-1.81	-0.99	0.08	0.18
	20	0.32	33.0	-1.81	-1.11	0.08	0.20
	30	0.34	33.2	-1.77	-1.16	0.08	0.15
	40	0.32	32.8	-1.80	-1.23	0.08	0.12
	50	0.33	33.2	-1.79	-0.95	0.08	0.13
Top-p	0.1	0.33	33.5	-1.85	-1.25	0.08	0.12
	0.2	0.32	33.1	**-1.76**	-1.18	0.08	0.16
	0.3	0.33	33.5	-1.79	-1.04	0.08	0.15
	0.4	0.32	32.4	-1.84	-1.09	0.08	0.15
	0.5	0.32	32.5	-1.85	-0.64	0.08	0.10
	0.6	0.31	32.7	-1.76	-0.86	0.08	0.15
	0.7	0.31	32.6	-1.76	-1.12	0.08	0.13
	0.8	0.30	31.9	-1.79	-0.84	0.08	0.13
	0.9	0.34	33.1	-1.80	-0.90	0.08	0.07
	1.0	0.34	32.8	-1.76	-0.73	0.08	0.15
Typical	0.1	0.32	32.3	-1.84	-1.05	0.08	0.14
	0.2	0.30	32.1	-1.82	-1.12	0.08	0.12
	0.3	0.31	32.4	-1.85	-1.00	0.08	0.17
	0.4	0.30	32.5	-1.84	-1.04	0.08	0.11
	0.5	0.32	32.7	-1.83	-1.05	0.08	0.14
	0.6	0.31	32.7	-1.78	-1.19	0.08	0.09
	0.7	0.31	32.4	-1.85	-1.08	0.08	0.18
	0.8	0.31	32.3	-1.85	-0.93	0.08	0.09
	0.9	0.32	32.9	-1.79	-0.81	0.08	0.15
	0.95	0.32	32.8	-1.81	-0.78	0.08	0.11
Mirostat	3.0	**0.34**	33.0	-1.81	-0.88	0.08	0.17

Table S8: Performance of the different sampling methods and hyperparameter values on double-IRS-HPF protein generation. Bold scores denote best values.

Sampling Method	Hyper. Value	Relevance		Quality		Diversity	
		TM-score ↑	pLDDT ↑	Avg. Fitness ↑	Max. Fitness ↑	SD ↑	FED ↑
Random		0.34	32.8	-1.97	-1.20	0.08	0.02
Greedy		0.34	33.6	-1.77	-1.03	**0.08**	0.19
Top-k	2	0.36	**34.5**	-1.70	-1.02	**0.08**	0.17
	3	0.34	34.1	-1.70	-0.89	**0.08**	0.10
	5	0.33	33.9	-1.75	-1.11	**0.08**	0.16
	10	0.37	34.4	-1.71	-0.88	0.08	0.10
	15	0.35	33.8	-1.75	-1.00	0.08	0.08
	20	**0.37**	34.3	-1.63	-0.96	0.08	0.07
	30	0.35	33.9	-1.68	**-0.76**	0.08	0.07
	40	0.36	34.4	-1.72	-1.10	0.08	0.06
	50	0.35	33.4	-1.79	-0.85	0.08	0.03
Top-p	0.1	0.32	31.7	-2.05	-1.14	0.08	0.03
	0.2	0.32	31.9	-2.03	-1.35	0.08	0.02
	0.3	0.33	32.2	-1.96	-0.95	0.08	0.01
	0.4	0.34	32.7	-1.95	-1.11	0.08	0.01
	0.5	0.33	32.6	-1.98	-1.18	0.08	0.03
	0.6	0.34	32.7	-1.95	-1.07	0.08	0.05
	0.7	0.31	31.8	-1.97	-1.25	0.08	0.01
	0.8	0.33	32.2	-1.97	-1.22	0.08	0.01
	0.9	0.32	32.4	-1.95	-1.01	0.08	0.05
	1.0	0.35	32.8	-1.95	-1.02	0.08	0.01
Typical	0.1	0.33	32.2	-1.96	-1.17	**0.08**	0.01
	0.2	0.34	32.9	-1.95	-1.08	0.08	0.02
	0.3	0.31	32.5	-1.98	-1.34	0.08	0.01
	0.4	0.35	32.6	-1.93	-1.35	0.08	0.02
	0.5	0.35	32.6	-1.89	-1.33	0.08	0.01
	0.6	0.35	32.7	-1.98	-1.00	0.08	0.01
	0.7	0.32	32.3	-1.95	-0.96	0.08	0.03
	0.8	0.33	33.0	-1.92	-1.19	0.08	0.03
	0.9	0.33	32.9	-1.95	-1.31	0.08	0.01
	0.95	0.33	32.7	-1.95	-1.11	0.08	0.03
Mirostat	3.0	0.33	32.1	-1.89	-1.14	0.08	0.02

Table S9: Performance of the different sampling methods and hyperparameter values on double-IRS-QFF protein generation. Bold scores denote best values.

H Algorithms

Algorithm S1 Single-mutant IRS Protein Generation Pseudocode

```
Require: $initial\_sequence, num\_sequences, cycles, output\_path$    ▷ Main Inputs
Require: $sampler, sampling\_threshold$    ▷ Final Sampler Params
Require: $PLM$    ▷ Protein Language Model (e.g., Tranception, RITA, Progen)
Ensure: $num > 0$ and $cycles > 0$
  $SN \leftarrow num\_sequences$    ▷ Number of sequence(s) to generate
  $EC \leftarrow cycles$    ▷ Number of evolution cycle(s) per sequence
  $generated\_sequences \leftarrow list()$
  $final\_sampler \leftarrow sampler$    ▷ Random, Greedy, Top-k, Top-p, Typical, or Mirostat
  $final\_threshold \leftarrow sampling\_threshold$    ▷ Threshold for final sampling
  while $len(generated\_sequences) < SN$ do
      $seq \leftarrow initial\_sequence$
      while $iteration < EC$ do
          $mutants \leftarrow generate\_all\_single\_mutants(seq)$
          $scores \leftarrow PLM(mutants)$
          $mutation \leftarrow final\_sampler(scores, final\_threshold)$
          $seq \leftarrow get\_mutated\_sequence(seq, mutation)$
      end while
      $generated\_sequences.append(seq)$
  end while
  $save\_as\_fasta(generated\_sequences)$    ▷ Output FASTA file
```

Algorithm S2 Autoregressive Protein Generation Pseudocode

```
Require: $initial\_sequence, num\_sequences, length, output\_path$    ▷ Main Inputs
Require: $sampler, sampling\_threshold$    ▷ Sampling Params
Require: $PLM$    ▷ Protein Language Model (e.g., Tranception, RITA, Progen)
Ensure: $num > 0$ and $length > 0$
  $SN \leftarrow num\_sequences$    ▷ Number of sequence(s) to generate
  $SL \leftarrow length$    ▷ Desired length of each sequence
  $EF \leftarrow extension\_factor$    ▷ No. of AA(s) to extend at each cycle
  $generated\_sequences \leftarrow list()$
  $final\_sampler \leftarrow sampler$    ▷ Random, Greedy, Top-k, Top-p, Typical, or Mirostat
  $final\_threshold \leftarrow sampling\_threshold$    ▷ Threshold for final sampling
  while $len(generated\_sequences) < SN$ do
      if $initial\_sequence$ then
          $seq \leftarrow initial\_sequence$
      else
          $seq \leftarrow random(AA\_residue)$
      end if
      while $seq\_length < SL$ do
          $mutants \leftarrow extend\_seq\_by\_N(seq, EF)$    ▷ Extend seq with EF residues
          $scores \leftarrow PLM(mutants)$
          $seq \leftarrow final\_sampler(scores, final\_threshold)$
          $seq\_length \leftarrow len(seq)$
      end while
      $generated\_sequences.append(seq)$
  end while
  $save\_as\_fasta(generated\_sequences)$    ▷ Output FASTA file
```

Algorithm S3 Multi-mutant IRS Protein Generation with HPF or QFF strategy Pseudocode

```
Require: $initial\_sequence, num, cycles, output\_path$                              ▷ Main Inputs
Require: $sampler, sampling\_threshold$                                      ▷ Final Sampler Params
Require: $multi\_mutant, inter\_threshold$                              ▷ Additional for multi-mutant
Require: $use\_fitness\_oracle, model\_path$                                  ▷ Only if using QFF
Require: $PLM$                  ▷ Protein Language Model (e.g., Tranception, RITA, Progen)
Require: $fitness\_oracle$                 ▷ Separate fitness prediction oracle (e.g., ProteinBERT)
Ensure: $num > 0$ and $cycles > 0$ and $multi\_mutant \geq 2$
  $SN \leftarrow num\_sequences$                                  ▷ Number of sequence(s) to generate
  $EC \leftarrow cycles$                                  ▷ Number of evolution cycle(s) per sequence
  $MM \leftarrow multi\_mutant$                 ▷ Number of multiple mutations to apply at each round
  $IST \leftarrow inter\_threshold$                                ▷ Threshold for intermediate sampling
  $final\_sampler \leftarrow sampler$              ▷ Random, Greedy, Top-k, Top-p, Typical, or Mirostat
  $final\_threshold \leftarrow sampling\_threshold$                            ▷ Threshold for final sampling
  $generated\_sequences \leftarrow list()$
  if $use\_fitness\_oracle$ then                                ▷ Using QFF intermediate sampling
      $intsampling \leftarrow load\_model(model\_path)$
  else                                                            ▷ Using HPF intermediate sampling
      $intsampling \leftarrow topk\_sampler()$
  end if
  while $len(generated\_sequences) < SN$ do
      $seq \leftarrow initial\_sequence$
      while $iteration < EC$ do
          $mutation\_count \leftarrow 0$
          while $mutation\_count < MM$ do
              $mutation\_count \mathrel{+}= 1$
              if $mutation\_count = 1$ then                                          ▷ First Mutation
                  $allmutants \leftarrow generate\_all\_single\_mutants(seq)$
                  $lastmutants \leftarrow PLM(allmutants)$
              end if
              if $mutation\_count > 1$ and $mutation\_count < MM$ then  ▷ Subsequent Mutation
                  $allmutants \leftarrow generate\_extra\_mutants(lastmutants)$
                  $extramutants \leftarrow intsampling(allmutants, IST)$
                  $lastmutants \leftarrow PLM(extramutants)$
              end if
              if $mutation\_count = MM$ then                                         ▷ Final Mutation
                  $allmutants \leftarrow generate\_extra\_mutants(lastmutants)$
                  $extramutants \leftarrow intsampling(allmutants, IST)$
                  $scoredmutants \leftarrow PLM(extramutants)$
                  $mutation \leftarrow final\_sampler(scoredmutants, final\_threshold)$
              end if
          end while
          $seq \leftarrow get\_mutated\_sequence(seq, mutation)$
      end while
      $generated\_sequences.append(seq)$
  end while
  $save\_as\_fasta(generated\_sequences)$                                         ▷ Output FASTA file
```