SAMPLING PROTEIN LANGUAGE MODELS FOR FUNCTIONAL PROTEIN DESIGN

Jeremie Theddy Darmawan[*†] **Yarin Gal**[‡] **Pascal Notin**[§]

ABSTRACT

Protein language models have emerged as powerful tools for learning rich representations of proteins, enhancing performance across various downstream tasks such as structure prediction, mutation effects prediction, and homology detection. Their ability to learn complex distributions over protein sequences also shows significant potential for designing novel and functional proteins, with broad applications in therapeutics, new materials, and sustainability. Given the vastness of the protein sequence space, efficient exploration methods are critical to the success of protein engineering efforts. However, the methodologies for effectively sampling from these models to achieve core protein design objectives remain underexplored and have predominantly relied on techniques initially developed for Natural Language Processing tasks. In this work, we first develop a comprehensive *in silico* protein design evaluation framework to systematically compare different sampling methods. After a thorough review of existing sampling strategies for language models, we introduce several approaches specifically tailored for protein design. We then evaluate these strategies using our *in silico* benchmark, investigating the effects of key hyperparameters and providing practical guidance on the relative strengths of each method depending on design objectives.

1 INTRODUCTION

Proteins are complex macromolecules that perform a wide diversity of essential functions in biological systems, such as catalyzing biochemical reactions, facilitating cellular signaling, and providing structural support. They cover a vast sequence space, representing all possible variations of amino acid chains that can fold into functional structures. Despite significant advancements in sequencing technologies, the protein sequences currently known and characterized represent only a small fraction of this immense sequence space (Notin et al., 2024). Protein design methodologies seek to efficiently explore this vast space to create novel protein sequences. This exploration can be achieved either by iteratively mutating existing functional sequences of interest (Arnold, 2018; Yang et al., 2019; Wittmann et al., 2021) or by designing entirely new sequences from scratch, drawing on existing templates and biochemical constraints (Huang et al., 2016; Dauparas et al., 2022). 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, such as structure prediction Lin et al. (2023); Lu et al. (2024), fitness and mutation effects prediction Meier et al. (2021); Brandes et al. (2022); Notin et al. (2022b); Su et al. (2024), homology detection Heinzinger et al. (2021), or viral evolution Hie et al. (2020); Thadani et al. (2023). As generative models of protein sequences, pLMs are ideally suited to sample novel functional proteins that have not yet been observed, making them invaluable in supporting protein engineering workflows. Given the massive size of the corresponding sample space, together with the diversity of protein design objectives, the success of pLM-based protein engineering efforts depends not only on ever-improving models but also on efficient methods to *sample* from these models. While the field has witnessed the emergence of several pLMs for protein design Alley et al. (2019); Ferruz et al. (2022); Hesslow et al. (2022); Nijkamp et al. (2022); Madani et al. (2023), efficient sampling methods have been understudied to date. Prior works have

[*]Correspondence to jeremie.darmawan@u.nus.edu, pascal_notin@hms.harvard.edu
[†]Department of Biochemistry, Yong Loo Lin School of Medicine, National University of Singapore
[‡]OATML Group, Department of Computer Science, University of Oxford
[§]Marks Lab, Department of Systems Biology, Harvard Medical School

so far primarily leveraged sampling strategies initially developed for Natural Language Processing (NLP) tasks (§ B.1), with limited insight into the impact of various strategies on the properties of the generated sequences, and without taking advantage of the unique characteristics of protein sequences relative to natural language sequences. In this work, we sought to develop and explore different methods for sampling from pLMs, systematically analyzing the characteristics of the generated sequences based on the chosen sampling strategy or core underlying hyperparameters. To that end, we first devise a robust *in silico* evaluation framework to benchmark different protein design methods. While our focus is on pLMs in general, we place a greater emphasis on autoregressive pLMs, given their ability to support a broader range of sampling methods (§ 2.1). We also place ourselves in the zero-shot design setting, aiming to design novel functional sequences without relying on experimental labels acquisition. Our contributions in this work are as follows. We introduce a high-level framework and taxonomy that encapsulate the landscape of existing protein language model (pLM) sampling methods (§ 2.2). Building on this foundation, we develop a suite of efficient sampling strategies tailored to protein design objectives, including the High-Probability Filter (HPF), Quantitative-Function Filter (QFF), Attention-Matrix Sampling (AMS), Random Stratified Filtering (RSF), and Monte Carlo Tree Search (MCTS) schemes (§ 2.3). Additionally, we devise a holistic framework for the *in silico* evaluation of protein engineering methods that spans core objectives such as functional relevance, sequence diversity, and fitness (§ 3). We also conduct an in-depth comparison of the various pLM sampling methods, analyzing the impact of key sampling hyperparameters on performance and discussing the relative strengths of the different approaches depending on design objectives (§ 4). The code is made publicly available for further research and development (§ A)

2 Methods

2.1 Taxonomy for pLM sampling methods

Sampling strategies for ML-based protein design can be broadly grouped into either Iterative Redesign Sampling (IRS) or AutoRegressive Sampling (ARS), as depicted in Figure S1. We can leverage Protein Language Models (pLMs) to generate novel protein sequences using either of these approaches (Strokach & Kim, 2022). Additionally, any of the sampling methods outlined in § B.1 can be applied in conjunction with both IRS and ARS approaches. **AutoRegressive Sampling:** Given an initial residue prompt, ARS sampling methods iteratively extend the amino acid (AA) sequence by one or multiple residues at each step. ARS strategies can be conditioned with a prompt containing a few residues, or be fully unconditional (empty prompt). A prompted generation typically allows for more targeted and family-specific generation (Nijkamp et al., 2022). The process stops when the desired sequence length has been reached or an 'end of sequence' token is generated. Pseudocode for the approach is provided in Algorithm S8. **Iterative Redesign Sampling:** IRS-based sampling starts by taking a template protein sequence (e.g., a natural protein sequence) and generating all possible single (or a subset of multiples) mutations, either at every position or at specific Position(s) of Interest (PoI). At each iteration, one (or multiple) substitution mutations are introduced and the process stops once a predefined number of evolution cycles is achieved. IRS encompasses Markov Chain Monte Carlo (MCMC) approaches such as Gibbs sampling (Sgarbossa et al., 2023; Johnson et al., 2023; Geman & Geman, 1984). Optimal mutation is determined by a fitness score from the underlying pLM and a predefined sampling strategy. This strategy can be applied not only to traditional MLM pLMs, but also to newer masked diffusion language modeling pLMs. An illustration of the code for single-mutant IRS is provided in Figure S2 and the detailed algorithm is available in Algorithm S7.

2.2 Single-Mutant Protein Generation

Single mutant IRS (IRS-singles) strategies require a template sequence as a starting point, and subsequently generate all possible single amino acid substitutions at each position, which are then scored with a fitness predictor to select which of these possible mutations to apply at every iteration. A predefined number of iterations determines the end of the process. In contrast, single-mutant ARS (ARS-singles) strategies iteratively build up a full sequence by adding a single residue at each cycle. This residue is sampled autoregressively from all 20 possible amino acids with the underlying generative model. In both ARS and IRS single strategies, we sample with one of the standard approaches introduced in § B.1. Single-mutant-based methods are straightforward to instantiate and have a linear computational complexity in the sequence length.

2.3 Multiple-Mutant Protein Generation

Table 1: **Comparison for ARS strategies**. Values reflect the best sampling method and hyperparameter combination. Detailed results are in Table S8. Bold indicates the highest scores.

Depth	Sampling	Relevance		Quality		Diversity		Overall Score ↑
		TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	Uniqueness ↑	FED ↑	
Singles	Top-k	**0.65**	**83.74**	96.5	2.19	**1.00**	1.32	**96.5**
Singles	Beam Search	0.54	60.13	**98.5**	**5.32**	**1.00**	99.11	91.5
Doubles	Top-k	0.55	56.14	**98.5**	3.91	**1.00**	109.44	89.5
Doubles	Beam Search	0.55	53.68	96.5	3.83	**1.00**	**118.55**	83.5

While simpler to implement and computationally tractable, methods that sample a single mutant at each iteration may result in suboptimal design performance as they ignore potentially critical epistatic effects between positions (Johnson et al., 2023; Starr & Thornton, 2016). Uncovering these relationships requires to consider the interactions between multiple mutated positions simultaneously (Upadhyay et al., 2022). An approach to this is through beam search which views a sequential sampling process as a search problem exploring the most promising mutations at each step in a tree-like structure, maintaining a fixed number of possible solutions called "beam width". However, given the combinatorial number of such interactions, an exhaustive enumeration and scoring of all possible mutants is impractical. Efficient multiple-mutant strategies thus rely on simplifying assumptions to make that search tractable. In this work, we introduce several such strategies that focus on efficient sampling of pairs of mutants. By adjusting the key variable N in the following strategies, we control the number of proteins explored at each step. In all our experiments, we set $N = 96$, as previously identified to be the minimal for exploration (Biswas et al., 2021). We refer to these methods as ARS-doubles and IRS-doubles. We provide a high-level illustration for the IRS-doubles strategies in Figure S2 and detail the corresponding pseudocode in Algorithm S9. **Quantitative-Function Filter (QFF).** We first generate all possible mutant pairs. We then quantify the protein fitness for all possible pairs with a Potts model, EVmutation (Hopf et al., 2017). This significantly reduces the computational complexity compared with doing the same with a large pLM, while keeping the ability to quantify epistatic effects between residue pairs. We then select the top-N predicted mutations, score these with the pLM. Lastly, we sample the double mutant to apply out of these N options with one of the sampling strategies discussed in § B.1. **High-Probability Filter (HPF).** We first generate and score all possible single amino acid substitutions. The Top-k (eg., $k = 100$) single mutants with the highest fitness scores are then selected and combined together to generate a set of high-quality double mutants. This drastically reduces the number of double mutations to consider, reducing the overall computational complexity of the algorithm to being linear in sequence length. From this set, we randomly sample *N* mutant pairs which we score with the pLM. While it is possible to focus on the mutant pairs with the highest average scores, we instead randomly sample a number N of them to promote exploration in sequence space. The double mutant to apply in the current cycle is then sampled out of these *N* options with one of the standard sampling strategies discussed in § B.1. **Attention-Matrix Sampling (AMS).** We first score all possible single mutants and select the top-K with the highest predicted fitness. Using the same pLM, we identify PoIs as residues with the highest average attention values in the self-attention layers. From this PoI set, we generate all possible mutant pairs and score them using a lightweight fitness model (EVmutation), selecting the top-N for further evaluation with the pLM. A double mutant is then sampled from these N options using a standard strategy. Alternative selection methods, such as identifying spatially proximal residues from experimental or predicted protein structures, may also be used instead of attention scores. **Random-Stratified Filter (RSF).** All possible double mutations are generated from the top-K single mutants. Each sequence is scored with a lightweight fitness prediction model (EVmutation) and stratified into four bins based on their scores: $\{S_{\text{very low}}, S_{\text{low}}, S_{\text{high}}, S_{\text{very high}}\}$. We then take a total of *N* double-mutants by taking the top-$N/4$ samples from each stratum to increase the diversity of mutations, intuitively mutants in different EVmutation score bins will be non-redundant and diverse in sequence space. This introduces mutation diversity at an intermediate level before further protein scoring. Lastly, we sample the double mutant to apply out of these N options with one of the standard sampling strategies. **Monte Carlo Tree Search (MCTS).** We start from a full sequence (natural or mutated) and consider each of its residues as a potential parent node (Parker & Chen, 2020; Wang et al., 2023). In each search round, the Upper Confidence bounds applied to Trees (UCT) algorithm (Kocsis & Szepesvári, 2006) balances exploration and exploitation of each parent node to identify desirable mutation nodes. To limit the computational complexity of the search and enable exploration of more nodes, we leverage our proposed efficient multi-mutant strategies described above (eg., QFF, HPF) after generating possible mutations. The pLM then scores the filtered mutations to quantify the reward for each node. Finally, the mutation node with the highest reward $\max(R_m)$ is selected.

3 Holistic *in silico* evaluation framework for protein design

When designing functional proteins, three high-level criteria are critical for performance. First, we want the generated protein sequences to be ***relevant*** to the design objective, i.e., carry out the function of interest. Since function is primarily encoded via the tertiary structure of the protein, the metrics we devised to assess functional relevance are structure-based, including TM-score and pLDDT. Second, the ***quality*** of the generated protein sequences can be assessed by their quality to *properly execute* the specified function, i.e., have high fitness Johnson et al. (2023). We measured this based on overall fitness and maximum fitness. Fitness prediction models have been shown to exhibit relatively high correlation with hundreds of experimental assays (Notin et al., 2023b) and may therefore be used as an in-silico oracle for true fitness. While these in-silico predictions are imperfect oracles, it should be noted that experiments themselves are imperfect and noisy measurements of the true fitness of proteins (Frazer et al., 2021). Third, the generated sequences should be ***diverse***, i.e., differ from the template wild-type sequence or known natural sequences, evaluated through Uniqueness and Fréchet ESM Distance (FED). Additionally, we computed an **Overall Score** that comprehensively summarises the evaluation by considering metrics from each objective. Computed by taking unique sequences with both a TM-score above 0.5 and fitness above 75th percentile relative to wild-type MSA, averaged across the two fitness oracles involved. We then divide this number by the total number of generated sequences to obtain a score between 0 (bad sampling) and 1 (good sampling). Further details on the metrics included are discussed in Appendix C.2

4 Experiments and Results

Table 2: **Comparison for IRS strategies**. Best combination of sampling and hyperparameter, averaged across protein families. Detailed results are in Tables S4 to S6. Best scores are bolded.

Depth	Sampling	Relevance		Quality		Diversity		Overall Score ↑
		TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	Uniqueness ↑	FED ↑	
Singles	Top-k	**0.99**	95.57	92.1	2.07	0.84	5.45	77.0
Doubles	QFF	0.96	**95.89**	**100.0**	**2.36**	**1.00**	**6.21**	**100.0**
	HPF	**0.99**	95.76	97.0	2.20	**1.00**	5.27	97.0
	AMS	**0.99**	95.74	**100.0**	**2.36**	0.99	5.97	99.0
	RSF	0.98	95.82	96.0	2.24	**1.00**	5.02	96.0
	MCTS	0.98	95.69	90.5	2.24	**1.00**	5.34	90.5
	Beam Search	**0.99**	95.84	99.5	2.18	**1.00**	5.42	99.5

In the experimental evaluation of the methods introduced in § 2, we seek to address the following questions: (1) What are the relative strengths of the various sampling strategies introduced above, and their impact on the properties of the generated sequences? (2) What are the benefits of multiple-mutant strategies over single-mutant strategies? (3) What are the effects of different sampling hyperparameters on the generated proteins? To that end, we chose Tranception (Notin et al., 2022a) for its state-of-the-art fitness performance against other autoregressive architectures (Notin et al., 2023a). Details on datasets, protein families, and sampling hyperparameters are available in Appendix C.
Effects of broad sampling approaches: We evaluated the properties of protein generation using both Autoregressive Sampling (ARS) and Iterative Redesign Sampling (IRS) across all categories in our evaluation framework. The best outcomes based on overall score across hyperparameters tested for each sampling method and averaged across protein families, are reported in Figure S4. We adjust parameters across experiments to ensure a comparable compute budget is used across sampling strategies. We observe that the IRS-doubles approach outperforms *in aggregate* all other sampling strategies. However, while IRS strategies tend to produce generally more relevant and functional samples in aggregate compared with ARS methods, the latter enable the design of a few highly performing designs (i.e., significantly higher maximum fitness value). As proteins generated through IRS method only mutate residues at certain positions, the generated proteins tend to be more similar to the starting wild type sequences (eg., higher TM-score). Conversely, by providing a fraction of the whole template sequence as prompt, ARS approaches have more flexibility to generate more diverse proteins, at the cost of relevance or quality. Additionally, we collected sequences generated through

MCMC-based sampling in Biswas et al. (2021) as additional baselines, focusing our analyses on the 2 protein families covered in that work (Table S2). We observe that MCMC-based sampling in Biswas et al. (2021) underperforms the various ARS and IRS sampling strategies introduced above. **Benefits of multiple-mutant strategies:** A comparative summary of the best sampling method and hyperparameter combinations for IRS-singles (single mutation per iteration) and IRS-doubles (double mutations per iteration) based on their overall scores, incorporating the strategies HPF, QFF, AMS, RSF, beam search, and MCTS, is shown in Table 2. In general, IRS-doubles consistently outperform IRS-singles on all metrics. Detailed results are available in Table S2. We also performed similar experiments for the generation of ARS proteins to compare single-mutant (ARS singles) and double-mutant (ARS doubles) generation. The sampling methods used for comparison are NLP-sampling strategies discussed in § B.1 and beam search. The beam search implemented in ARS can generate either single or multiple AA extensions, hence its inclusion in both singles and doubles. Results for the best sampling method and hyperparameter combinations for the ARS-singles and ARS-doubles experiments are selected based on the overall score and presented in Table 1. Detailed results are provided in Table S3. Unlike IRS strategies, we do not observe any significant benefit of ARS-doubles approaches over ARS-singles. Within the IRS approach, there are several key observations to be highlighted. The combination of Beam Search with HPF produced proteins with the highest relevance scores, indicating that this method is ideal for generating proteins with the desired function. Detailed performance breakdown across sampling strategies is provided in Appendix H. In contrast, QFF with random sampling and AMS produced novel proteins with the highest general score, with relevant generated sequences of high fitness and diversity, scoring highly on the two fitness-related metrics. For discovering new and unexplored protein sequences, almost all IRS-doubles methods were highly effective in generating novel proteins. Based on the overall scores, QFF offers a well-rounded balance across all metrics and can thus be considered the most optimal protein generation method. Based on their specific design objectives, practitioners may choose the most appropriate method. Furthermore, we find that there are significant performance differences across protein families (see detailed results in Tables S4 to S7). The BLAT protein was the most easily evolvable, leading to the highest maximum fitness level across proteins. However, when double-mutations (IRS-doubles) are introduced, other proteins climbed to higher fitness levels and approached the gains achieved in the BLAT experiments (Table S5 and Table S6). This confirmed that multi-mutant approaches can be useful by detecting epistatic effects and making less myopic decisions about which mutations to apply at each step. The ARS results show that almost all the proteins generated have fitness over 75th percentile of natural sequences. Adjusting sampling temperature to lower values (e.g. 0.01) for beam search and nucleus sampling results in better evaluation scores, consistent with previous findings (Ruffolo et al., 2024). Interestingly, when comparing the ARS-singles and ARS-doubles approaches, transitioning to double mutations lead to worse protein generation in terms of relevance (lower TM score and pLDDT), but with positive results in terms of maximum fitness and diversity in aggregate (Xu & Zhang, 2010). When searching for balance between all the metrics, the ARS-singles approach with Top-k achieves the highest overall score, pLDDT, and Uniqueness. Detailed results across different protein families are provided in Table S8. We observe differences in evolvability across the 3 protein families, with BLAT proving to be the most difficult to achieve significant leaps in maximum fitness.

5 Conclusion

In this work, we systematically evaluated the effects of different sampling strategies on the quality, diversity, and relevance of designed protein sequences in silico. We observed a general trade-off between the overall fitness and diversity of the proteins produce across methods: sampling methods that produce greater novelty and diversity in sequences yield proteins that are less functional on average. Selecting the appropriate sampling strategy and fine-tuning hyperparameters is thus crucial for striking the right balance between these competing objectives. Our research also highlights the benefits resulting from developing custom sampling methodologies tailored specifically for protein design, leading to higher aggregate performance across design objectives. Lastly, the observation that the optimal sampling strategy varies across protein family suggests that leveraging several sampling strategies simultaneously could be beneficial. This combined approach would help generate more diverse samples and improve the likelihood of discovering designs with maximum fitness. As the field progressively moves towards developing multi-modal protein language models combining sequence, structure and functional annotations, we anticipate that novel sampling approaches will be needed to fully take advantage of the dependencies across modalities.

REFERENCES

Ethan C. Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George 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.

Frances 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.

Sourya Basu, Govardana Sachitanandam Ramachandran, Nitish Shirish Keskar, and Lav R. Varshney. Mirostat: A neural text decoding algorithm that directly controls perplexity. In *International Conference on Learning Representations*, 2021.

Surojit Biswas, Grigory Khimulya, Ethan C. Alley, Kevin M. Esvelt, and George 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.

Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal 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.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games*, pp. 72–83. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-75538-8_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.

D.C Dowson and B.V Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, September 1982. doi: 10.1016/0047-259x(82)90077-x.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard 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.2021.3095381.

Noelia Ferruz, Steffen Schmidt, and Birte Höcker. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1), July 2022. doi: 10.1038/s41467-022-32007-7.

Julia M Flynn, Neha Samant, Gily Schneider-Nachum, David T Barkan, Nese Kurt Yilmaz, Celia A Schiffer, Stephanie A Moquin, Dustin Dovala, and Daniel NA Bolon. Comprehensive fitness landscape of sars-cov-2 m^{pro} reveals insights into viral resistance mechanisms. *eLife*, 11:e77433, jun 2022. ISSN 2050-084X. doi: 10.7554/eLife.77433.

Jonathan Frazer, Pascal Notin, Mafalda Dias, Aidan Gomez, Joseph K Min, Kelly P. Brock, Yarin Gal, and Debora S. Marks. Disease variant prediction with deep generative models of evolutionary data. *Nature*, 2021.

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

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6 (6):721–741, November 1984. ISSN 0162-8828. doi: 10.1109/tpami.1984.4767596.

Michael Heinzinger, Maria Littmann, Ian P. W. Sillitoe, Nicola Bordin, Christine A. Orengo, and Burkhard Rost. Contrastive learning on protein embeddings enlightens midnight zone. *NAR Genomics and Bioinformatics*, 4, 2021.

Daniel Hesslow, Niccoló Zanichelli, Pascal Notin, Iacopo Poli, and Debora Marks. Rita: a study on scaling up generative protein sequence models. *arXiv preprint arXiv:2205.05789*, 2022.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

Brian L. Hie, Ellen D. Zhong, Bonnie Berger, and Bryan D. Bryson. Learning the language of viral evolution and escape. *Science*, 371:284 – 288, 2020.

Ari Holtzman, Jan Buys, Leo Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CEUR Workshop Proceedings*, 2540, 4 2019. ISSN 16130073.

Thomas A Hopf, John B Ingraham, Frank J Poelwijk, Charlotta P I Schärfe, Michael Springer, Chris Sander, and Debora S Marks. Mutation effects predicted from sequence co-variation. *Nature Biotechnology*, 35(2):128–135, January 2017. doi: 10.1038/nbt.3769.

Po-Ssu Huang, Scott E. Boyken, and David Baker. The coming of age of de novo protein design. *Nature*, 537:320–327, 2016.

Hervé Jacquier, André Birgy, Hervé Le Nagard, Yves Mechulam, Emmanuelle Schmitt, Jérémy Glodt, Beatrice Bercot, Emmanuelle Petit, Julie Poulain, Guilène Barnaud, Pierre-Alexis Gros, and Olivier Tenaillon. Capturing the mutational landscape of the beta-lactamase tem-1. *Proceedings of the National Academy of Sciences*, 110(32):13067–13072, July 2013. ISSN 1091-6490. doi: 10.1073/pnas.1215206110.

Sean R. Johnson, Xiaozhi Fu, Sandra Viknander, Clara Goldin, Sarah Monaco, Aleksej Zelezniak, and Kevin K. Yang. Computational scoring and experimental evaluation of enzymes generated by neural networks. *bioRxiv*, pp. 2023.03.04.531015, 4 2023. doi: 10.1101/2023.03.04.531015.

Timothy Fei Truong Jr and Tristan Bepler. PoET: A generative model of protein families as sequences-of-sequences. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (eds.), *Machine Learning: ECML 2006*, pp. 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46056-5.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Pretrained language models for text generation: A survey, 2022.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. doi: 10.1126/science.ade2574.

Amy X. Lu, Wilson Yan, Sarah A. Robinson, Kevin K. Yang, Vladimir Gligorijević, Kyunghyun Cho, Richard Bonneau, Pieter Abbeel, and Nathan C. Frey. Generating all-atom protein structure from sequence-only training data. *bioRxiv*, 2024.

Ali Madani, Ben Krause, Eric R. Greene, Subu Subramanian, Benjamin P. Mohr, James M. Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z. Sun, Richard Socher, James S. Fraser, and Nikhil Naik. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology 2023*, pp. 1–8, 1 2023. ISSN 1546-1696. doi: 10.1038/s41587-022-01618-2.

Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex 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.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121, 2 2022. ISSN 2307387X. doi: 10.1162/tacl_a_00536.

Geraldene Munsamy, Sebastian Lindner, Philipp Lorenz, and Noelia Ferruz. Zymctrl: a conditional language model for the controllable generation of artificial enzymes. *Machine Learning for Structural Biology Workshop. NeurIPS 2022.*, 2022.

Erik Nijkamp, Jeffrey Ruffolo, Eli N. Weinstein, Nikhil Naik, and Ali Madani. Progen2: Exploring the boundaries of protein language models, 2022.

Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16990–17017. PMLR, 17–23 Jul 2022a.

Pascal Notin, Lood Van Niekerk, Aaron W Kollasch, Daniel Ritter, Yarin Gal, and Debora S. Marks. Trancepteve: Combining family-specific and family-agnostic models of protein sequences for improved fitness prediction. *bioRxiv*, 2022b.

Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, Jonathan Frazer, Mafalda Dias, Dinko Franceschi, Yarin Gal, and Debora Marks. Proteingym: Large-scale benchmarks for protein fitness prediction and design. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 64331–64379. Curran Associates, Inc., 2023a.

Pascal Notin, Aaron W. Kollasch, Daniel Ritter, Lood van Niekerk, Steffanie Paul, Hansen Spinner, Nathan Rollins, Ada Shaw, Ruben Weitzman, Jonathan Frazer, Mafalda Dias, Dinko Franceschi, Rose Orenbuch, Yarin Gal, and Debora S. Marks. Proteingym: Large-scale benchmarks for protein design and fitness prediction. *bioRxiv*, 2023b. doi: 10.1101/2023.12.07.570727.

Pascal Notin, Nathan J. Rollins, Yarin Gal, Chris Sander, and Debora Marks. Machine learning for functional protein design. *Nature Biotechnology*, 42:216–228, 2024.

Alexey A. Pakhomov and Vladimir I. Martynov. Gfp family: Structural insights into spectral tuning. *Chemistry & Biology*, 15(8):755–764, August 2008. ISSN 1074-5521. doi: 10.1016/j.chembiol.2008.07.009.

Jerrod Parker and Jerry Zikun Chen. Neural machine translation with monte-carlo tree search, 2020.

Daniel Peñas-Utrilla and Enrique Marcos. Identifying well-folded de novo proteins in the new era of accurate structure prediction. *Frontiers in Molecular Biosciences*, 9, 2022.

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

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with tape. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8844–8856. PMLR, 18–24 Jul 2021.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob 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.

Jeffrey A. Ruffolo, Stephen Nayfach, Joseph Gallagher, Aadyot Bhatnagar, Joel Beazer, Riffat Hussain, Jordan Russ, Jennifer Yip, Emily Hill, Martin Pacesa, Alexander J. Meeske, Peter Cameron, and Ali Madani. Design of highly functional genome editors by modeling the universe of crispr-cas sequences. *bioRxiv*, 2024. doi: 10.1101/2024.04.22.590591.

Merijn L.M. Salverda, J. Arjan G.M. De Visser, and Miriam Barlow. Natural evolution of tem-1 β-lactamase: experimental reconstruction and clinical relevance. *FEMS Microbiology Reviews*, 34 (6):1015–1036, November 2010. ISSN 1574-6976. doi: 10.1111/j.1574-6976.2010.00222.x.

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, Natalya S. Bogatyreva, Peter K. Vlasov, Evgeny S. Egorov, Maria D. Logacheva, Alexey S. Kondrashov, Dmitry M. Chudakov, Ekaterina V. Putintseva, Ilgar Z. Mamedov, Dan S. Tawfik, Konstantin A. Lukyanov, and Fyodor 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.

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

Damiano Sgarbossa, Umberto Lupo, and Anne-Florence 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.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, October 2017. doi: 10.1038/nature24270.

Tyler N. Starr and Joseph W. Thornton. Epistasis in protein evolution. *Protein Science*, 25(7): 1204–1218, February 2016. doi: 10.1002/pro.2897.

Michael A. Stiffler, Doeke R. Hekstra, and Rama Ranganathan. Evolvability as a function of purifying selection in tem-1 β-lactamase. *Cell*, 160(5):882–892, February 2015. ISSN 0092-8674. doi: 10.1016/j.cell.2015.01.035.

Alexey Strokach and Philip M. Kim. Deep generative modeling for protein design. *Current Opinion in Structural Biology*, 72:226–236, 2 2022. ISSN 0959-440X. doi: 10.1016/J.SBI.2021.11.008.

Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. Saprot: Protein language modeling with structure-aware vocabulary. *bioRxiv*, 2024.

Nicole N. Thadani, Sarah Gurev, Pascal Notin, Noor Youssef, Nathan J. Rollins, Chris Sander, Yarin Gal, and Debora S. Marks. Learning from prepandemic data to forecast viral escape. *Nature*, 622:818 – 825, 2023. URL `https://api.semanticscholar.org/CorpusID:263906690`.

Roger 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.

Vaibhav Upadhyay, Casey Patrick, Alexandra Lucas, and Krishna M.G. 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.

Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Jue Wang, Sidney Lisanza, David Juergens, Doug Tischer, Joseph L. Watson, Karla M. Castro, Robert Ragotte, Amijai Saragovi, Lukas F. Milles, Minkyung Baek, Ivan Anishchenko, Wei Yang, Derrick R. Hicks, Marc Expòsit, Thomas Schlichthaerle, Jung-Ho Chun, Justas Dauparas, Nathaniel Bennett, Basile I. M. Wicky, Andrew Muenks, Frank DiMaio, Bruno Correia, Sergey Ovchinnikov, and David Baker. Scaffolding protein functional sites using deep learning. *Science*, 377(6604):387–394, 2022. doi: 10.1126/science.abn2100.

Yi Wang, Hui Tang, Lichao Huang, Lulu Pan, Lixiang Yang, Huanming Yang, Feng Mu, and Meng Yang. Self-play reinforcement learning guides protein engineering. *Nature Machine Intelligence*, July 2023. doi: 10.1038/s42256-023-00691-9.

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

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, Basile I M Wicky, Nikita Hanikel, Samuel J Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976): 1089–1100, August 2023.

Bruce J Wittmann, Kadina E Johnston, Zachary Wu, and Frances H Arnold. Advances in machine learning for directed evolution. *Current Opinion in Structural Biology*, 69:11–18, August 2021. doi: 10.1016/j.sbi.2021.01.008.

Jinrui Xu and Yang Zhang. How significant is a protein structure similarity with tm-score = 0.5? *Bioinformatics*, 26(7):889–895, February 2010. ISSN 1367-4803. doi: 10.1093/bioinformatics/btq066.

Kevin K. Yang, Zachary Wu, and Frances 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.

Yang Zhang and Jeffrey 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 DATA AND CODE AVAILABILITY

to facilitate further research and development, we open-source our codebase, providing convenient access to our unified *in silico* generation and evaluation framework, which includes all the sampling methods, at `https://github.com/i3LBI19-OATML/sampling_plm`.

B BACKGROUND AND RELATED WORK

B.1 SAMPLING LANGUAGE MODELS

Language models are generative models seeking to approximate a distribution over sequential inputs. Once trained, we can sample from these generative models to create new sequences that may not have been part of the initial training data. In NLP, since the creation of novel sequences 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 (Li et al., 2022). Throughout the text, we refer to these sampling approaches as the *standard sampling strategies*. We describe below some of these strategies, where x represents the sequence token. The most simple strategy is **random sampling**, which samples items based on their probabilities as provided in Equation (1), where $\hat{x}_t^{(i)}$ is the unnormalized logits at step t for item i in the vocabulary V, divided by the smoothing temperature parameter T which controls the flatness of the distribution.

$$P(\hat{x}_t^{(i)} \mid x_{<t}) = \frac{\exp(\hat{x}_t^{(i)}/T)}{\sum_{j \in V} \exp(\hat{x}_t^{(j)}/T)} \tag{1}$$

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).

$$P^k(\hat{x}_t^{(i)} \mid x_{<t}) = \begin{cases} P(\hat{x}_t^{(i)} \mid x_{<t})/p', & \text{if } \hat{x}_t^{(i)} \in V^{(k)}. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where $x_{<t}$ corresponds to sequence items previously generated, and $p' = \sum_{\hat{x}_t^{(i)} \in V^{(k)}} P(\hat{x}_t^{(i)} \mid x_{<t})$ is the probability re-normalization factor. **Greedy sampling** is an extreme version of Top-k sampling that only takes the item with the highest (top-1) probability in the distribution.

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 Holtzman et al. (2019) (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 $p' = \sum_{\hat{x}_t^{(i)} \in V^{(p)}} P(\hat{x}_t^{(i)} \mid x_{<t})$.

Beam search views a sequential sampling process as a search problem exploring the most promising mutations at each step in a tree-like structure, maintaining a fixed number of possible solutions called "beam width". Equation (3) outlines this approach, where elements x_t of the beam width at step t are the ones that maximizes the scoring function $s(.)$ (typically the likelihood) of the concatenation (denoted by the operator [:]) of sequences $x_{<t}$ in the beam width $X_{<t}$ at the prior step, together with any other element $\hat{x}_t^{(i)}$ from the vocabulary V at step t.

$$x_t = \underset{\substack{[x_{<t}:\hat{x}_t] \\ |X_{<t}|=B}}{\operatorname{argmax}} \left\{ s([x_{<t} : \hat{x}_t^{(i)}]) \mid i \in V, x_{<t} \in X_{<t} \right\} \tag{3}$$

Other sampling techniques maximizing human expectations and preferences were also developed for NLP. **Typical sampling** attempts to balance out the information content of the generated sequence and the expected information content (entropy) as shown in Equation (4) (Meister et al., 2022). The truncation set $C(x_{<t})$ represents the solution to the minimization problem, $\mathcal{P}$ is the power set operator, V is the vocabulary, H is the entropy, and τ is the amount of probability mass to be considered.

$$\min_{C(x_{<t})\in\mathcal{P}(V)} \sum_{x\in C(x_{<t})} \{H(\hat{x}_t^{(i)} \mid x_{<t}) + \log p(\hat{x}_t^{(i)} \mid x_{<t})\}$$

$$\text{s.t.} \sum_{x\in C(x_{<t})} p(\hat{x}_t^{(i)} \mid x_{<t}) \geq \tau \quad (4)$$

Basu et al. (2021) developed a fine-tuning-free method, called **mirostat sampling**, which self-adapts the perplexity of generated sequence at a predefined level based on human preference. This is done by truncating the token candidate list using top-k, while self-adaptation can be achieved by adjusting the error based on the observed and target perplexity difference. Algorithm S1 details this approach, where $\mathfrak{S}(x)$ is the observed perplexity, τ is the target perplexity, m is the number of most probable tokens, μ is the maximum perplexity, η is the learning rate, ϵ is the error term, and $\hat{s}$ refers to Zipf's exponent (Basu et al., 2021).

Language modeling can also be viewed as a sequence of decisions where the current generated tokens (state) will influence the next tokens (moves) being generated to achieve certain objectives. **Monte Carlo Tree Search (MCTS)** (Coulom, 2007; Silver et al., 2017) has been used in machine translation and, as a result, for protein design (Parker & Chen, 2020; Wang et al., 2023). Recently, Wang et al. (2023) leveraged a MCTS-based approach to carry out a 35 residue-restricted iterative protein design. The search mechanism is based on iterative rounds of simulations from the current state where the algorithm will play out subsequent moves. It leverages a UCT (Upper Confidence bounds applied to Trees) algorithm to decide between exploring or exploiting the current state. Exploration refers to playing moves on unexplored nodes whereas exploitation re-investigates nodes that have been previously played for higher rewards. The information on each node visits (potential mutations) and rewards (improvement in fitness) is backpropagated through the tree to improve subsequent simulation attempts. Equation (5) expresses the UCT algorithm for a particular node, where W_i is the total number of successful (positive) simulations, S_i is the total number of simulations, S_p is the total number of simulations of the parent node, and c is the exploration parameter.

$$UCT(\hat{x}_t^{(i)}) = \frac{W_i}{S_i} + c\sqrt{\frac{\ln S_p}{S_i}} \quad (5)$$

B.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 (Elnaggar et al., 2022) or FIM (Heinzinger et al., 2021) have also been explored. Unirep (Alley et al., 2019) was the first pLM trained autoregressively across unaligned protein sequences across protein families. With the improvements seen in Transformers (Vaswani et al., 2017) for NLP, masked-language models such as ProteinBERT, ESM-1b, TAPE-BERT, MSA Transformer, and SaProt (Brandes et al., 2022; Rives et al., 2021; Rao et al., 2019; 2021; Su et al., 2024), as well as autoregressive models, such as ProtGPT2, Progen, Tranception, RITA, ZymCTRL, PoET (Ferruz et al., 2022; Madani et al., 2023; Notin et al., 2022a; Hesslow et al., 2022; Munsamy et al., 2022; Jr & Bepler, 2023), have been proposed to further increase the quality of learned representations. These pLMs have been shown to achieve remarkable performance across diverse tasks, such as fitness prediction (Meier et al., 2021; Notin et al., 2022a) or structure prediction (Lin et al., 2023). For protein design, prior pLM works have mainly relied on autoregressive generation with simple sampling methods discussed in § B.1, such as Top-k, Top-p, random, or greedy sampling (Biswas et al., 2021; Sgarbossa et al., 2023; Nijkamp et al., 2022; Madani et al., 2023; Ferruz et al., 2022). Sgarbossa et al. (2023) developed a method that iteratively applies random masking of residues in the protein sequence and fills them greedily based on MSA Transformer (Rao et al., 2021) predictions. While these various approaches were successful in generating novel protein sequences, none of these prior works have systematically studied the effects and relative trade-offs of various sampling methods on the properties of generated sequences.

C Experimental Details

C.1 Experimental Protocol

As we need an autoregressive model that can support both ARS and IRS strategies, we chose Tranception (Notin et al., 2022a) to carry out our experiments as it achieves state-of-the-art fitness prediction performance compared with other autoregressive architectures (Notin et al., 2023a). For each sampling method, hyperparameter value, and protein family, we ultimately generated 100 novel designs after several rounds of in silico optimization. Evolution cycles for IRS are set at 4 while desired length for ARS are set equal to wild-type (this ensures all approaches have roughly the same compute budget). Details on datasets, protein families, and sampling hyperparameters are available in Appendix C.

C.2 Evaluation Framework

- **TM-score:** Assesses topological similarity between the predicted structure of the generated sequences and the structure of the template wild type sequence Zhang & Skolnick (2004). The final score is the average of the pairwise TM-scores. Structures were predicted with ESMFold Lin et al. (2023). Ideally, the structure of generated sequences is as close to that of the template sequence as possible to ensure functional relevance.
- **pLDDT:** Quantifies the confidence of a structure prediction model (here ESMFold (Lin et al., 2023)), averaged over the entire sequence. Higher pLDDT has been empirically observed to correlate with higher functional designs (Peñas-Utrilla & Marcos, 2022; Watson et al., 2023; Wang et al., 2022). Note that this may be limiting in certain cases (e.g. disordered proteins).
- **Overall Fitness:** Reports the proportion of designs with fitness greater than the 75th percentile of fitness scores for natural sequence in the corresponding MSAs to the total number of designs.
- **Maximum Fitness:** Evaluates the ability of the sampling procedure to generate sequences that significantly enhance the fitness of the starting wild-type sequence for the corresponding protein family. For the two aforementioned fitness oracles (ESM-1v and ProGen2) and for each sampled sequence, we compute the ratio of the predicted fitness over that of the wild type sequence. The maximum ratio across designs is then extracted for each predictor and averaged across them.
- **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 generative adversarial network (GAN)-generated images (Raghava & Barton, 2006; Heusel et al., 2018; Seitzer, 2020). It leverages the embeddings from the penultimate layer of ESM-1v Meier et al. (2021), recapitulating on broader protein features instead of scores, and computes the Fréchet distance between the embeddings of the generated sequences and the sequences in an MSA for that protein family. It also has a linear correlation to Sequence Dissimilarity (SD), as seen in Table S2-Table S8.
- **Uniqueness:** Reports the proportion of distinct sequences generated. Calculated by taking the ratio of unique sequences to the total number of sequences.
- **Overall Score:** Comprehensively summarises the evaluation by considering metrics from each objective. Computed by taking unique sequences with both a TM-score above 0.5 and fitness above 75th percentile relative to wild-type MSA, averaged across the two fitness oracles involved. We then divide this number by the total number of generated sequences to obtain a score between 0 (bad sampling) and 100 (good sampling).

C.3 Evaluation 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.

C.4 Data

Generation using IRS and ARS methods requires an initial protein sequence with a certain length. EVmutation library initiation requires wild-type DMS data from the protein of interest. Proteins that have extensive DMS reports are valid options, including the ones in the ProteinGym benchmarks (Notin et al., 2023a). Specifically for IRS generation, AA residue PoI is also required for masking and mutation targets.

C.5 Selected protein families

To test the generalizability of our results we conducted our experiments for three diverse protein families: *Aequorea victoria* Green Fluorescence Protein (avGFP) (§ F.1), *Escherichia coli* Beta-lactamase TEM-1 (BLAT) (§ F.2), and the Mpro (also known as 3CLpro) region of replicase protein 1ab (R1AB) from the SARS-CoV-2 virus (§ F.3). We performed all IRS experiments starting from the corresponding full-length wild-type sequences and used conditioning prompts of at least 5% and up to 85% of the full sequence length for the ARS experiments (Figure S5). Wild-type sequences and MSAs for the 3 protein families were obtained from ProteinGym benchmarks (Notin et al., 2023a).

C.6 Sampling and 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 (Basu et al., 2021), 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 2 and 3.

C.7 Prompt Lengths

To address the limitation of shorter prompt lengths, we note that one could alternatively condition on residues starting from the N-terminus and autoregressively generate towards the C-terminus, and then condition on the C-terminus residues and autoregressively generate residues towards the N-terminus. This is possible with models such as Tranception which have been trained to reconstruct sequences autoregressively from both directions.

D Code Availability

to facilitate further research and development, we open-source our codebase, providing convenient access to our unified *in silico* generation and evaluation framework, which includes all the sampling methods, at `https://anonymous.4open.science/r/sampling_plm-4B32`.

E Effects of different sampling hyperparameters

We first sought to investigate the impact of the conditioning prompt length used in ARS on design performance. To that end, we performed sampling experiments across the 3 protein families using prompt lengths based on 5 percentage values (5%, 20%, 40%, 65%, and 85%). Results are reported in Figure S5 and Figure S6. In general, metrics including TM-score, Uniqueness, FED, and Overall Score generally increase as prompt length increases. On the other hand, both pLDDT and maximum fitness exhibit the opposite trend, with 65% length showing the best balance for all the metrics. The impact of prompt length varies across protein families: avGFP and BLAT show moderate changes, while R1AB exhibits drastic increases (FED) or declines (uniqueness and maximum fitness). Longer prompts typically allow ARS methods to generate more family-relevant samples, at the cost of no diversity in the prompting region. Interestingly, diversity also appears to increase with prompt length (up to a limit) as short prompts often lead to degenerate sequences, which fail in relatively similar patterns (eg., excessive repeat of the same amino acid). We also note that for shorter prompt lengths, the model may generate novel sequences with high predicted fitness but corresponding to a different function than the target, as evidenced by the lower relevance score obtained for these designs. All our subsequent ARS experiments were done with a 65% prompt length for a balanced generation process.

We then aimed to analyze how different hyperparameter values for various sampling methods affect design performance. The hyperparameters we focused on were the values of *k* for Top-k sampling, *p* (or nucleus) for Top-p (nucleus) sampling, τ (tau) for typical sampling, and modifying the sampling temperature. For beam search, we tested the impact of the maximum length of subsequent mutations (number of steps) and the number of search rounds for MCTS. To analyze the impact of different hyperparameters on design outcomes, we used Top-k sampling from IRS-singles generation, across the three protein families. The corresponding results are reported in Figure S7. Maximum fitness remained high across all sampling values. However, by prioritizing high-fitness mutations with IRS (i.e. smaller k values), proteins generated were more likely to stay within the same family as limited mutations were introduced. Although it provided higher overall fitness, lowering the k hyperparameter value led to repetitive mutations in successive generations (i.e. fewer unique proteins generated). Conversely, by increasing the value of k, the distance (i.e. FED) between the generated proteins and the starting sequence increased as more diverse mutations were introduced.

F Glossary

F.1 avGFP Protein

The jellyfish *Aequorea victoria* has a particular protein that exhibits a green fluorescent when exposed to light (Tsien, 1998). Our experiments were conducted using the *Aequorea victoria* green fluorescent protein (avGFP) data by Sarkisyan et al. (2016). It is also one of the widely used biophysical property benchmarks and used in other protein generation attempts by Biswas et al. (2021). Several PoI (66, 65, 148, 203, 205, 145) were identified to be highly determinant for its function (Pakhomov & Martynov, 2008).

F.2 BLAT Protein

A fourth of the proteins in *Escherichia coli* are enzymes and beta-lactamase TEM-1 is an enzyme that confers resistance to the bacteria (Jacquier et al., 2013). Resistance is conferred by hydrolyzing β-lactam rings of penicillins and other common antibiotics found in healthcare facilities (Salverda et al., 2010). It also has a well-studied 3D structure and thermodynamic range that allow it to be suitable for medical investigations into drug resistance. The PoI at 36, 164, 179, 182, and 250 are critical for executing its function (Jacquier et al., 2013; Stiffler et al., 2015).

F.3 R1AB Protein

Belonging to the SARS-CoV-2 virus, the Mpro region of the R1AB protein (also known as the main protease Mpro) and its substrates play an essential functional role throughout every step of the viral life cycle. The protein is a peptidase that initiates autoproteolysis of the pp1a and pp1an polyprotein translated from the ORF1 gene which is essential for viral replication (Flynn et al., 2022). Residues at locations 145, 41, 299, 142, and 189 are PoI that were identified by Flynn et al. (2022) that significantly contribute to its proteolytic activity.

F.4 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.

F.5 Greedy Sampling

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

F.6 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.

F.7 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 by Holtzman et al. (2019). 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.

F.8 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 (Basu et al., 2021). 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).

F.9 Typical Sampling

Inspired by the psycholinguistic theory of human communication, Meister et al. (2022) 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.

F.10 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 (Fréchet, 1957; Heusel et al., 2018). This measure is also known as the Wasserstein-2 distance (Wasserstein., 1969). 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). Modifed into a protein-related analog of the Fréchet Inception Distance (FID), introduced to evaluate images generated by generative adversarial network (GAN)-generated images (Raghava & Barton, 2006; Heusel et al., 2018; Seitzer, 2020). Since we only modified the Inception model used in Heusel et al. (2018) into ESM-1v that is suitable for protein sequences, the "Fréchet Inception Distance" (FID) equation (Dowson & Landau, 1982) still applies:

$$d^2((m,C),(m_w,C_w)) = (||m-m_w||)_2^2 + Tr(C + C_w - 2(CC_w)^{\frac{1}{2}}) \tag{6}$$

G EVmutation Initialization

Initializing the EVmutation (Hopf et al., 2017) model requires two steps: (1) collecting deep multiple sequence alignments (MSA) and (2) generating the model parameters. MSAs were collected from the ProteinGym database for each protein family (Notin et al., 2023a). We used the recommended plmc (Hopf et al., 2017) commands to accomplish this with L2-regularization on the couplings set, applying their formula.

H FIGURES AND TABLES

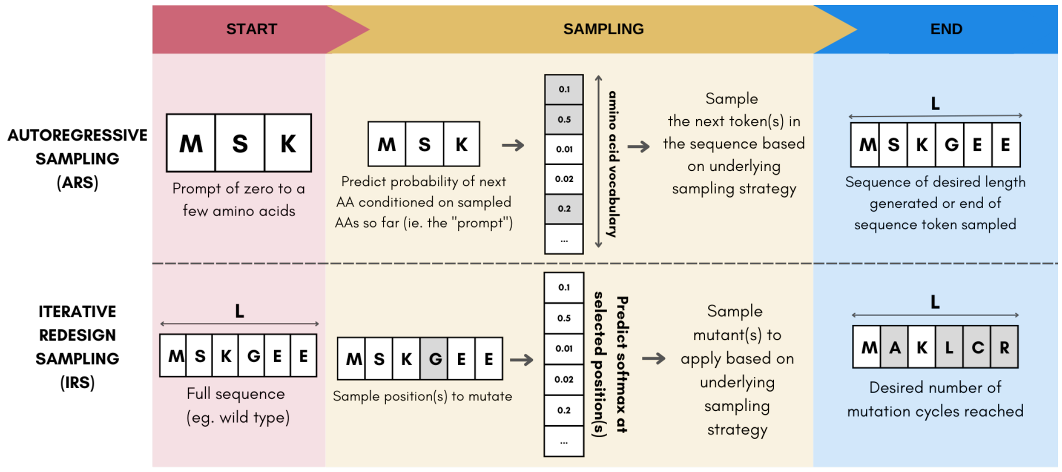

START
SAMPLING
END
AUTOREGRESSIVE SAMPLING (ARS)
M S K
Prompt of zero to a few amino acids
Predict probability of next AA conditioned on sampled AAs so far (ie. the "prompt")
amino acid vocabulary
Sample the next token(s) in the sequence based on underlying sampling strategy
L
M S K G E E
Sequence of desired length generated or end of sequence token sampled
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 underlying sampling strategy
M A K L C R
Desired number of mutation cycles reached

Figure S1: **Taxonomy of Protein Language Models sampling methods.** We discriminate between two strategies for sampling pLMs: Autoregressive Sampling and Iterative Redesign Sampling.

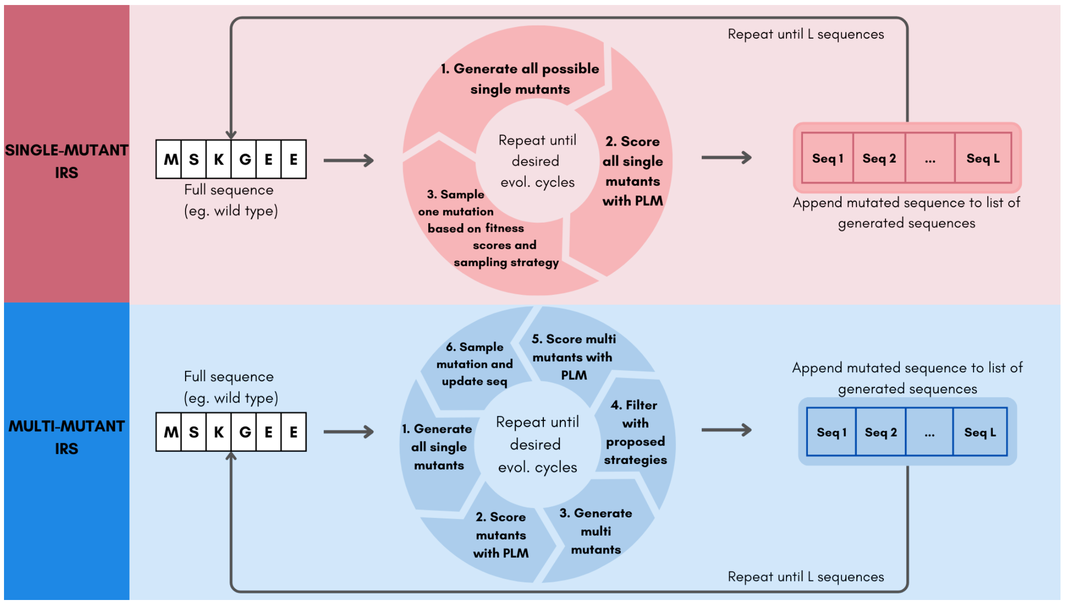

SINGLE-MUTANT IRS
M S K G E E
Full sequence (eg. wild type)
1. Generate all possible single mutants
2. Score all single mutants with PLM
3. Sample one mutation based on fitness scores and sampling strategy
Repeat until desired evol. cycles
Repeat until L sequences
Seq 1 Seq 2 ... Seq L
Append mutated sequence to list of generated sequences
MULTI-MUTANT IRS
1. Generate all single mutants
2. Score mutants with PLM
3. Generate multi mutants
4. Filter with proposed strategies
5. Score multi mutants with PLM
6. Sample mutation and update seq

Figure S2: **High-level single and multi-mutant IRS algorithms for generating novel protein sequences**. Detailed algorithms are provided in supplementary (Algorithm S7 and Algorithm S9).

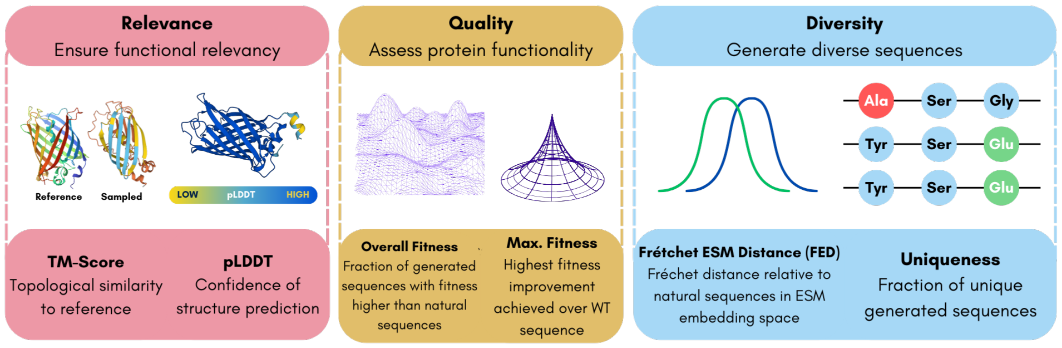
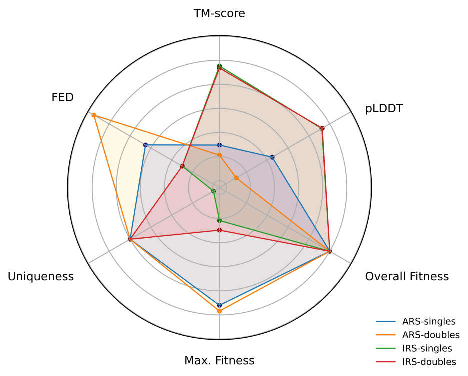

Relevance
Ensure functional relevancy
Reference
Sampled
LOW
pLDDT
HIGH
TM-Score
Topological similarity to reference
pLDDT
Confidence of structure prediction
Quality
Assess protein functionality
Overall Fitness
Fraction of generated sequences with fitness higher than natural sequences
Max. Fitness
Highest fitness improvement achieved over WT sequence
Diversity
Generate diverse sequences
Ala
Ser
Gly
Tyr
Ser
Glu
Tyr
Ser
Glu
Fréchet ESM Distance (FED)
Fréchet distance relative to natural sequences in ESM embedding space
Uniqueness
Fraction of unique generated sequences

Figure S3: **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.



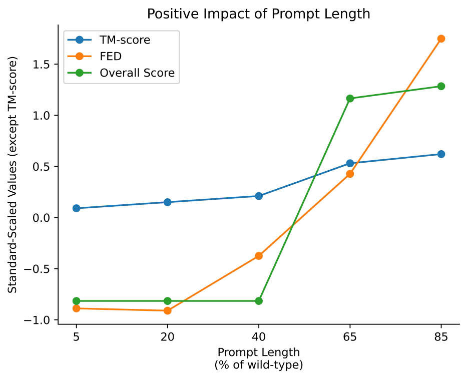
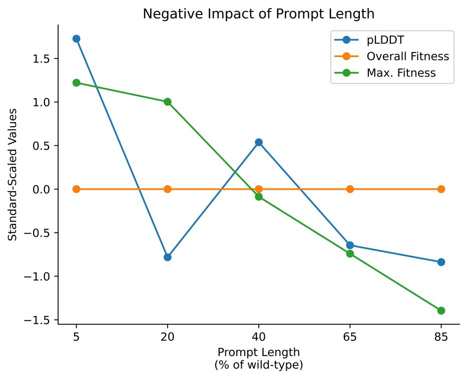


Figure S4: **Overview of IRS and ARS methods across evaluation framework metrics.** Best combination of sampling method and hyperparameter are reflected, averaged across protein families. Detailed results are in Table S1

Figure S5: **Impact of prompt lengths on sampling quality.** Results for a particular prompt length are obtained using the best method and hyperparameter, averaged across families.



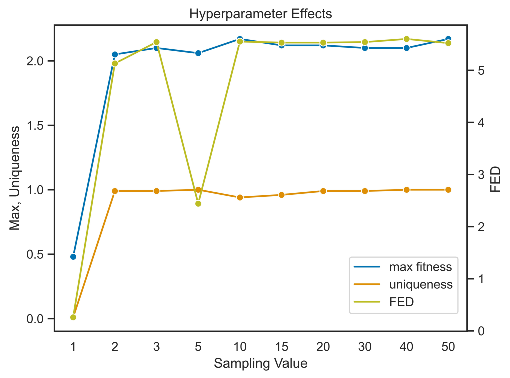


Figure S6: **Negative impact of prompt lengths based on percentage of wild-type prompted on our evaluation framework.** The values for a particular prompt length were obtained using the best combination across the experimented proteins and models. Standardization was performed to plot each of the effects.

Figure S7: **Comparing the effects of Top-k hyperparameters on Max. Fitness, Uniqueness, and FED metrics.**

Table S1: **Comparing across ARS and IRS (singles and doubles) methods on the different aspects of the evaluation framework.** Bold denotes best scores.

Method	Relevance		Quality		Diversity		Overall Score ↑
	TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	Uniqueness ↑	FED ↑	
ARS-singles	0.60	71.94	**97.50**	3.76	**1.00**	50.22	94.00
ARS-doubles	0.55	54.91	**97.50**	**3.87**	**1.00**	**114.00**	86.50
IRS-singles	**0.99**	95.57	92.10	2.07	0.84	5.45	77.00
IRS-doubles	0.98	**95.79**	97.17	2.26	**1.00**	5.54	**97.00**

Table S2: **Detailed comparison between the IRS protein generation classes for protein family: avGFP and BLAT.** Detailed best results for each method are available in Table S4 (IRS) and Table S5 and Table S6 (IRS-doubles). Bold denotes best scores. Results here are identical as with the inclusion of 3 protein families in Table 2

Depth	Sampling	Sampling Value	Relevance		Quality			Diversity		Overall Score ↑
			TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	SD ↑	Uniqueness ↑	FED ↑	
Singles	Top-k	20	**0.99**	95.57	92.10	2.07	22.45	0.84	5.45	77.00
Doubles	QFF-Random	-	0.96	**95.89**	**100.00**	**2.36**	**34.34**	**1.00**	**6.21**	**100.00**
	HPF-Greedy	-	**0.99**	95.76	97.00	2.20	7.00	**1.00**	5.27	97.00
	AMS-TopK	20	**0.99**	95.74	**100.00**	**2.36**	13.63	0.99	5.97	99.00
	RSF-Greedy	-	0.98	95.82	96.00	2.24	21.02	**1.00**	5.02	96.00
	MCTS-HPF	3	0.98	95.69	90.50	2.24	0.88	**1.00**	5.34	90.50
	Beam Search-HPF	3	**0.99**	95.84	99.50	2.18	1.20	**1.00**	5.42	99.50
	MCMC Biswas et al. (2021)		0.60	44.03	13.30	0.59	0.99	0.77	0	10.64

Table S3: **Detailed comparison between the ARS protein generation class.** Detailed best results for each method are available in Table S8. Bold denotes best scores.

Depth	Sampling	Sampling Value	Relevance		Quality			Diversity		Overall Score ↑
			TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	SD ↑	Uniqueness ↑	FED ↑	
Singles	Top-k	3	**0.65**	**83.74**	96.50	2.19	**41.87**	**1.00**	1.32	**96.50**
	Beam Search	2	0.54	60.13	**98.50**	**5.32**	33.87	**1.00**	99.11	91.50
Doubles	Top-k	20	0.55	56.14	**98.50**	3.91	33.93	**1.00**	109.44	89.50
	Beam Search	2	0.55	53.68	96.50	3.83	33.48	**1.00**	**118.55**	83.50

Table S4: Performance of the best sampling methods and hyperparameter values on IRS-singles protein generation across 3 proteins experimented.

	Sampling	Sampling Value	Relevance		Quality		Diversity			Overall Score ↑
			TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	SD ↑	Uniqueness ↑	FED ↑	
	avGFP									
	greedy	-	0.62	45.31	100.00	0.48	2.52	0.01	0.26	1.00
	mirostat	3.50	0.62	43.84	82.50	0.73	2.42	1.00	0.09	75.50
	random	-	0.60	42.76	51.00	0.68	2.37	1.00	0.12	47.00
	topk	2.00	0.60	45.18	90.00	0.60	2.52	1.00	0.06	81.50
	topp	0.70	0.61	44.11	80.00	0.70	2.46	1.00	0.06	75.50
	typical	0.70	0.58	42.92	84.00	0.70	2.43	1.00	0.08	76.00
	BLAT									
IRS-singles	greedy	-	1.00	95.92	100.00	2.16	30.77	0.01	6.02	1.00
	mirostat	4.50	0.99	95.55	90.00	2.21	4.34	1.00	5.40	90.00
	random	-	0.99	95.41	92.00	2.03	4.14	0.99	5.38	91.00
	topk	30.00	0.99	95.85	99.50	2.27	3.38	1.00	5.60	99.50
	topp	0.60	0.98	95.38	99.50	2.11	35.96	1.00	5.54	99.50
	typical	0.80	0.98	95.36	98.50	2.15	3.78	1.00	5.58	98.50
	R1AB									
	greedy	-	0.69	58.01	50.00	-0.30	1.63	0.01	2.49	0.50
	mirostat	4.50	0.71	60.24	49.00	0.56	1.49	1.00	0.69	49.00
	random	-	0.70	59.52	49.50	0.38	1.56	1.00	1.12	49.50
	topk	2.00	0.69	59.58	50.00	0.05	1.63	1.00	1.81	50.00
	topp	0.60	0.68	59.07	50.00	-0.02	1.60	1.00	1.13	49.50
	typical	0.80	0.69	59.28	50.00	0.36	1.60	1.00	1.03	49.50

Table S5: Performance of the best sampling methods and hyperparameter values on IRS-doubles QFF (left) and AMS (right) protein generation across 3 proteins experimented.

	Sampling	Sampling Value	Relevance		Quality		Diversity			Overall Score ↑
			TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	SD ↑	Uniqueness ↑	FED ↑	
	avGFP									
	greedy	-	0.60	45.75	50.00	0.32	1.26	0.01	0.18	0.50
	mirostat	3.00	0.63	46.22	71.00	0.71	1.57	0.69	0.09	50.00
	random	-	0.62	46.75	72.00	0.75	2.05	1.00	0.05	69.50
	topk	3	0.62	46.24	73.50	0.74	1.84	0.98	0.09	72.00
	topp	0.40	0.62	46.60	77.00	0.75	1.93	1.00	0.07	75.00
	typical	0.95	0.63	46.68	71.50	0.67	1.89	1.00	0.09	71.00
QFF96 (IRS-doubles)	**BLAT**									
	greedy	-	0.99	95.62	100.00	2.24	10.14	0.01	6.65	1.00
	mirostat	3.00	0.93	95.88	100.00	2.23	4.89	0.07	6.76	7.00
	random	-	0.96	95.89	100.00	2.36	33.20	1.00	6.21	100.00
	topk	30	0.97	95.84	100.00	2.38	26.99	1.00	6.15	100.00
	topp	0.90	0.96	95.86	100.00	2.38	4.65	1.00	6.22	100.00
	typical	0.90	0.96	95.87	100.00	2.35	4.66	1.00	6.22	100.00
	R1AB									
	greedy	-	0.98	62.43	50.00	0.23	2.61	0.01	0.33	0.50
	mirostat	3.00	0.68	63.51	11.00	-0.24	2.61	0.03	0.07	0.50
	random	-	0.73	63.34	41.00	1.88	2.61	1.00	0.08	40.50
	topk	20	0.80	62.62	50.00	0.76	2.60	1.00	0.12	50.00
	topp	0.10	0.73	63.16	42.50	1.74	2.60	1.00	0.08	41.50
	typical	0.10	0.75	63.65	44.50	0.80	2.61	0.99	0.04	44.00

	Sampling	Sampling Value	Relevance		Quality		Diversity			Overall Score ↑
			TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	SD ↑	Uniqueness ↑	FED ↑	
	avGFP									
	greedy	-	0.60	44.72	0.00	0.19	2.10	0.01	0.21	0
	mirostat	3.00	0.59	46.72	15.00	0.47	2.36	0.50	0.04	7.00
	random	-	0.62	45.08	13.00	0.47	2.74	1.00	0.06	13.00
	topk	10	0.62	45.29	33.00	0.44	2.26	1.00	0.07	32.00
	topp	0.40	0.59	44.08	22.50	0.52	2.68	1.00	0.06	20.50
	typical	0.90	0.61	44.64	17.50	0.53	2.68	1.00	0.05	17.00
AMS96 (IRS-doubles)	**BLAT**									
	greedy	-	1.00	96.06	100.00	2.11	31.47	0.01	5.88	1.00
	mirostat	3.00	0.95	95.39	100.00	2.16	43.71	0.02	6.50	2.00
	random	-	0.98	95.47	86.50	2.30	17.93	1.00	5.66	86.50
	topk	15	0.99	95.75	100.00	2.31	34.74	1.00	5.78	100.00
	topp	0.60	0.98	95.54	89.00	2.29	5.52	1.00	5.72	89.00
	typical	0.10	0.98	95.60	95.50	2.27	35.36	1.00	5.94	95.50
	R1AB									
	greedy	-	0.84	62.54	50.00	-0.05	2.61	0.01	0.26	0.50
	mirostat	3.00	0.78	62.73	36.50	0.71	2.58	1.00	0.07	36.50
	random	-	0.78	62.76	36.50	0.65	2.60	1.00	0.09	36.00
	topk	2	0.89	62.33	41.00	0.37	2.61	1.00	0.12	41.00
	topp	0.50	0.80	62.69	40.50	1.13	2.60	1.00	0.08	40.50
	typical	0.95	0.80	62.69	40.00	0.74	2.60	1.00	0.09	40.00

Table S6: Performance of the best sampling methods and hyperparameter values on IRS-doubles RSF (left) and HPF (right) protein generation across 3 proteins experimented.

	Sampling	Sampling Value	Relevance		Quality		Diversity			Overall Score ↑
			TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	SD ↑	Uniqueness ↑	FED ↑	
	avGFP									
	greedy	-	0.58	43.03	17.50	0.56	2.98	1.00	0.07	16.00
	mirostat	3.00	0.48	37.08	2.00	0.53	3.37	1.00	0.06	0.50
	random	-	0.55	40.02	10.50	0.57	3.31	1.00	0.02	10.00
	topk	2	0.60	43.02	19.50	0.70	3.05	1.00	0.07	18.00
	topp	0.80	0.53	39.70	12.00	0.55	3.32	1.00	0.06	10.00
	typical	0.80	0.54	40.49	13.50	0.65	3.35	1.00	0.05	10.50
RSF96 (IRS-doubles)	**BLAT**									
	greedy	-	0.98	95.82	96.00	2.24	5.03	1.00	5.02	96.00
	mirostat	3.00	0.96	94.87	50.00	1.78	4.63	1.00	4.08	50.00
	random	-	0.96	95.36	56.50	2.07	54.31	1.00	4.32	56.50
	topk	2	0.98	95.76	95.50	2.23	4.72	1.00	4.82	95.50
	topp	0.80	0.96	95.28	59.00	2.08	10.13	1.00	4.61	59.00
	typical	0.90	0.97	95.32	57.50	2.06	6.12	1.00	4.55	57.50
	R1AB									
	greedy	-	0.75	62.02	17.50	0.43	2.59	1.00	0.08	17.50
	mirostat	3.00	0.73	60.93	15.00	1.19	2.61	1.00	0.15	15.00
	random	-	0.72	61.42	19.50	0.74	2.60	1.00	0.11	19.50
	topk	40	0.73	61.64	21.50	0.88	2.59	1.00	0.10	21.50
	topp	0.30	0.72	61.18	21.00	1.07	2.61	1.00	0.13	21.00
	typical	0.60	0.74	61.47	19.50	0.34	2.60	1.00	0.13	19.50

	Sampling	Sampling Value	Relevance		Quality		Diversity			Overall Score ↑
			TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	SD ↑	Uniqueness ↑	FED ↑	
	avGFP									
	greedy	-	0.62	43.09	13.00	0.60	1.58	1.00	0.08	12.00
	mirostat	3.00	0.55	40.00	7.50	0.46	2.81	1.00	0.05	6.00
	random	-	0.55	40.46	10.00	0.60	2.82	1.00	0.06	5.50
	topk	20	0.61	42.26	14.50	0.58	2.58	1.00	0.05	14.00
	topp	0.10	0.54	39.87	10.50	0.61	2.90	1.00	0.06	9.00
	typical	0.10	0.56	40.17	15.00	0.86	2.92	1.00	0.06	14.00
HPF96 (IRS-doubles)	**BLAT**									
	greedy	-	0.99	95.76	97.00	2.20	6.64	1.00	5.27	97.00
	mirostat	3.00	0.97	95.44	60.00	2.02	36.75	1.00	4.45	60.00
	random	-	0.97	95.43	59.50	2.08	8.47	1.00	4.79	59.50
	topk	5	0.98	95.76	92.50	2.15	55.31	1.00	5.11	92.50
	topp	0.20	0.97	95.34	61.50	2.02	3.93	1.00	4.64	61.50
	typical	0.95	0.97	95.51	59.00	2.03	24.02	1.00	4.79	59.00
	R1AB									
	greedy	-	0.78	62.24	19.50	0.48	1.76	1.00	0.06	19.50
	mirostat	3.00	0.73	61.42	20.50	0.99	2.24	1.00	0.08	20.00
	random	-	0.74	61.38	18.00	1.12	2.28	1.00	0.09	18.00
	topk	40	0.75	61.88	25.00	0.75	2.19	1.00	0.06	25.00
	topp	1.00	0.73	61.59	20.50	0.67	2.24	1.00	0.14	20.50
	typical	0.60	0.75	61.59	21.00	0.80	2.31	1.00	0.09	21.00

Table S7: Performance of the best sampling methods and hyperparameter values on IRS-doubles Beam Search and MCTS protein generation across 3 proteins experimented.

	Sampling	Sampling Value	Relevance		Quality		Diversity			Overall Score ↑
			TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	SD ↑	Uniqueness ↑	FED ↑	
Beam Search (IRS-doubles)	**avGFP**									
	AMS	2	0.62	44.13	50.00	0.30	0.42	0.01	0.15	0.50
	HPF	2	0.66	43.40	15.00	0.55	0.87	1.00	0.09	14.00
	QFF	3	0.71	45.89	50.00	0.26	1.26	0.01	0.18	0.50
	BLAT									
	AMS	3	1.00	95.91	100.00	2.13	5.24	0.01	6.21	1.00
	HPF	3	0.99	95.84	99.50	2.18	33.66	1.00	5.42	99.50
	QFF	3	1.00	95.77	100.00	2.10	33.92	0.01	6.05	1.00
	R1AB									
	AMS	3	0.94	61.12	50.00	-0.03	1.96	0.01	0.07	0.50
	HPF	3	0.80	62.21	19.50	0.40	1.38	1.00	0.05	19.50
	QFF	3	0.96	61.85	50.00	0.25	1.96	0.01	0.52	0.50
MCTS (IRS-doubles)	**avGFP**									
	AMS	2	0.62	44.32	0.00	0.22	0.42	0.01	0.16	0
	HPF	2	0.59	42.30	13.00	0.46	1.17	1.00	0.07	12.00
	QFF	2	0.59	47.50	50.00	0.09	0.42	0.01	0.17	0.50
	BLAT									
	AMS	2	1.00	95.97	100.00	2.15	8.04	0.01	5.83	1.00
	HPF	3	0.98	95.69	90.50	2.24	16.99	1.00	5.34	90.50
	QFF	2	0.94	95.39	50.00	1.78	4.54	0.01	5.88	0.50
	R1AB									
	AMS	2	0.96	62.04	50.00	0.08	1.95	0.01	0.03	0.50
	HPF	3	0.79	61.99	21.00	0.75	0.90	1.00	0.05	21.00
	QFF	3	0.69	62.27	50.00	0.21	1.31	0.01	0.04	0.50

Table S8: Performance of the best sampling methods and hyperparameter values on ARS-singles and ARS-doubles protein generation across 3 proteins experimented.

	Sampling	Sampling Value	Relevance		Quality		Diversity			Overall Score ↑
			TM-score ↑	pLDDT ↑	Overall Fitness ↑	Max. Fitness ↑	SD ↑	Uniqueness ↑	FED ↑	
ARS-singles	**avGFP**									
	greedy	-	0.34	48.48	100.00	2.67	34.00	0.01	102.26	0
	mirostat	1.00	0.58	44.28	0.50	0.36	34.14	1.00	162.17	0.50
	random	-	0.59	43.92	2.00	0.42	34.40	1.00	160.68	1.50
	topk	3	0.54	59.74	100.00	5.60	34.20	1.00	81.80	91.00
	topp	0.10	0.58	44.58	9.50	0.80	32.76	1.00	152.73	9.50
	topp_t0.01	0.10	0.58	44.58	9.50	0.80	32.76	1.00	152.73	9.50
	typical	0	0.58	43.43	3.00	0.53	34.47	1.00	156.83	3.00
	beam_search	2	0.55	58.41	96.50	5.22	33.93	1.00	101.93	84.50
	beam_search_t0.01	2	0.54	60.13	98.50	5.32	33.87	1.00	99.11	91.50
	BLAT									
	greedy	-	0.70	84.24	100.00	1.41	32.65	0.01	4.54	1.00
	mirostat	2.50	0.67	89.83	93.50	2.42	34.75	1.00	0.86	79.50
	random	-	0.69	69.55	50.00	0.42	53.36	1.00	25.67	50.00
	topk	3	0.65	83.74	96.50	2.19	41.87	1.00	1.32	96.50
	topp	0.20	0.71	92.83	90.50	2.37	42.83	1.00	2.78	90.50
	topp_t0.01	0.40	0.71	71.12	50.00	0.71	30.67	1.00	21.40	50.00
	typical	1	0.67	90.14	60.50	1.65	35.82	1.00	1.72	58.00
	beam_search	2	0.68	81.61	77.00	2.02	32.68	1.00	3.07	77.00
	beam_search_t0.01	2	0.68	81.10	79.50	2.05	51.54	1.00	2.49	79.50
	R1AB									
	greedy	-	0.69	69.24	50.00	2.14	29.21	0.01	139.80	0.50
	mirostat	4.00	0.40	39.35	47.00	0.66	49.86	1.00	2.08	3.50
	random	-	0.31	45.87	48.50	10.32	50.71	1.00	4.12	1.50
	topk	2	0.65	67.54	45.00	8.04	31.75	1.00	156.10	44.50
	topp	0.10	0.68	59.86	3.50	-0.90	30.82	1.00	200.39	3.50
	topp_t0.01	0.90	0.68	59.59	4.50	-1.05	30.90	1.00	199.31	4.50
	typical	0	0.68	59.04	4.50	-0.16	31.76	1.00	198.46	4.50
	beam_search	3	0.67	62.64	38.00	3.49	30.97	1.00	178.66	38.00
	beam_search_t0.01	2	0.67	62.85	39.00	5.90	31.08	1.00	177.12	38.50
ARS-doubles	**avGFP**									
	greedy	-	0.65	49.86	100.00	0.53	22.18	0.01	141.04	1.00
	mirostat	1.50	0.59	44.39	1.50	0.36	34.17	1.00	161.31	1.50
	random	-	0.59	44.34	1.00	0.31	34.21	1.00	160.82	1.00
	topk	20	0.55	56.14	98.50	3.91	33.93	1.00	109.44	89.50
	topp	0.50	0.57	44.06	7.00	0.54	32.34	1.00	156.02	6.00
	topp_t0.01	0.50	0.57	44.06	7.00	0.54	32.34	1.00	156.02	6.00
	typical	0	0.58	43.83	1.50	0.49	34.29	1.00	158.86	1.50
	beam_search	2	0.53	53.47	94.50	3.40	33.10	1.00	119.06	80.50
	beam_search_t0.01	2	0.55	53.68	96.50	3.83	33.48	1.00	118.55	83.50
	BLAT									
	greedy	-	0.74	74.92	50.00	0.61	50.50	0.01	14.34	0.50
	mirostat	0.50	0.70	69.39	50.00	0.38	34.61	1.00	25.78	50.00
	random	-	0.69	69.83	50.00	0.42	35.08	1.00	25.93	50.00
	topk	20	0.67	77.73	82.00	1.73	51.59	1.00	2.98	82.00
	topp	0.60	0.70	69.63	50.00	0.34	31.87	1.00	25.53	50.00
	topp_t0.01	0.40	0.70	70.06	50.00	0.53	31.55	1.00	23.16	50.00
	typical	0	0.70	69.33	50.00	0.36	32.03	1.00	24.70	50.00
	beam_search	2	0.68	74.69	75.00	1.58	30.14	1.00	4.63	75.00
	beam_search_t0.01	2	0.68	74.63	74.00	1.58	30.27	1.00	4.76	74.00
	R1AB									
	greedy	-	0.66	57.13	0.00	-1.97	29.65	0.01	189.76	0
	mirostat	1.50	0.68	59.26	1.50	-1.57	31.94	1.00	205.09	1.50
	random	-	0.68	59.28	1.00	-1.33	31.66	1.00	204.11	1.00
	topk	10	0.67	62.48	37.00	4.77	30.97	1.00	178.65	37.00
	topp	0.40	0.68	59.61	2.00	-0.86	30.75	1.00	201.38	2.00
	topp_t0.01	0.60	0.68	59.83	5.00	-0.66	31.09	1.00	198.72	5.00
	typical	0	0.68	58.77	2.50	-1.50	31.71	1.00	200.90	2.50
	beam_search	3	0.67	61.64	33.00	1.56	31.18	1.00	184.64	33.00
	beam_search_t0.01	2	0.68	61.26	33.00	3.14	31.09	1.00	184.39	33.00

I Algorithms

Algorithm S1 Mirostat sampling (Basu et al., 2021)

```
Target perplexity $\tau$, maximum perplexity $\mu = 2\tau$, learning rate $\eta$, $m = 100$
while tokens to be generated do
    Compute $\hat{s} = \sum_{i=1}^{m-1} t_i b_i / \sum_{i=1}^{m-1} t_i^2$
    Compute $k = (\hat{\epsilon} 2^{\mu} / (1 - N^{-\hat{\epsilon}}))^{(1/\hat{s})}$
    Sample next token $X$ with top-$k$ sampling
    Compute error: $e = \mathfrak{S}(x) - \tau$
    Update $\mu$: $\mu = \mu - \eta e$
end while
```

Algorithm S2 Quantitative-Function Filter (QFF) pseudocode

```
1: Score all possible single amino acid substitutions $S$ for focus sequence $x$
2: Generate all possible double mutations $D$ from set $S$
3: Score double mutations $S_D$ using a lightweight fitness model (EVmutation)
4: Select the $N$ mutants from $S_D$ with the highest predicted scores
5: Sample the mutant pair to apply out of $N$ with one of the standard sampling strategies
```

Algorithm S3 High-Probability Filter (HPF) pseudocode

```
1: Score all possible single amino acid substitutions $S$ for focus sequence $x$
2: Select top-$K$ single mutations $S_K$ with highest predicted fitness by the pLM
3: Generate all possible double mutations $D$ from set $S_K$
4: Sample $N$ mutants at random from $D$
5: Sample the mutant pair to apply out of $N$ with one of the standard sampling strategies
```

Algorithm S4 Attention-Matrix Sampling (AMS) pseudocode

```
1: Score all possible single amino acid substitutions $S$ for focus sequence $x$
2: Select top-$K$ single mutations $S_K$ with highest predicted fitness by the pLM
3: Get important protein $PoI$ from $S_K$ with the highest average attention values in the underlying transformer
4: Generate all possible double mutants $D$ from set of $PoI$
5: Score double mutations $S_D$ using EVmutation
6: Sample $N$ mutants from $S_D$ with max predicted scores
7: Sample the mutant pair to apply out of $N$ with one of the standard sampling strategies
```

Algorithm S5 Random-Stratified Filter (RSF) pseudocode

```
1: Score all possible single amino acid substitutions $S$ for focus sequence $x$
2: Select top-$K$ single mutations $S_K$ with highest predicted fitness
3: Generate all possible double mutations $D$ from set $S_K$
4: Score double mutations $S_K$ using EVmutation
5: Stratify double mutations $S_K$ into 4 bins: $\{S_{\text{very low}}, S_{\text{low}}, S_{\text{high}}, S_{\text{very high}}\}$
6: Sample $N/4$ mutant pairs from each bin and combine to obtain $N$ double mutants
7: Sample the mutant pair to apply out of $N$ with one of the standard sampling strategies
```

Algorithm S6 MCTS Pseudocode

```
1: Consider each residue $x$ in the sequence as (parent) nodes $n_x$
2: for each round in a pre-defined number $R$ of search rounds do
3:     Decide exploration or exploitation of each node using UCT
4:     Generate subsequent mutation nodes $m$ for focus node $n_x$
5:     Select $N$ mutations $m_N$ from the set $m$ by applying one of the efficient multi-mutant strategies above (eg., QFF, HPF)
6:     Score mutations $m_N$ using pLM to get rewards $R^m$
7: end for
8: Select node $\max(R_m)$ with highest rewards
```

Algorithm S7 IRS-singles Protein Generation Pseudocode

```
Require: $initial\_sequence, num, cycles, output\_path$ {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}
  $final\_sampler \leftarrow sampler$ {Random, Greedy, Top-k, Top-p, Typical, or Mirostat}
  $final\_threshold \leftarrow sampling\_threshold$ {Threshold for final sampling}
  while $len(generated\_sequences) < SN$ do
    $iteration \leftarrow 0$
    $seq \leftarrow initial\_sequence$
    while $iteration < EC$ do
      $mutants \leftarrow generate\_all\_single\_mutants(seq)$
      $scores \leftarrow pLM(mutants)$
      $mutation \leftarrow final\_sampler(scores, final\_threshold)$
      $seq \leftarrow get\_mutated\_sequence(seq, mutation)$
      $iteration \mathrel{+}= 1$
    end while
    $generated\_sequences.append(seq)$
  end while
  $df \leftarrow pd.DataFrame(generated\_sequences)$
  $save\_as\_fasta(df, output\_path)$ {Output FASTA file}
```

Algorithm S8 Autoregressive (ARS-singles and -doubles) Protein Generation Pseudocode

```
Require: $initial\_sequence, num, length, output\_path$ {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}
  $final\_sampler \leftarrow sampler$ {Random, Greedy, Top-k, Top-p, Typical, or Mirostat}
  $final\_threshold \leftarrow sampling\_threshold$ {Threshold for final sampling}
  while $len(generated\_sequences) < SN$ do
    $iteration \leftarrow 0$
    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}
      $scores \leftarrow pLM(mutants)$
      $mutation \leftarrow final\_sampler(scores, final\_threshold)$
      $seq \leftarrow mutation$
      $seq\_length \leftarrow len(seq)$
    end while
    $generated\_sequences.append(seq)$
  end while
  $df \leftarrow pd.DataFrame(generated\_sequences)$
  $save\_as\_fasta(df, output\_path)$ {Output FASTA file}
```

Algorithm S9 IRS-doubles Protein Generation 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., EVmutation)}
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}
  if $use\_fitness\_oracle$ then
    $intsampling \leftarrow load\_model(model\_path)$ {Using QFF intermediate sampling}
  else
    $intsampling \leftarrow topk\_sampler()$ {Using HPF intermediate sampling}
  end if
  while $len(generated\_sequences) < SN$ do
    $iteration \leftarrow 0$
    $seq \leftarrow initial\_sequence$
    while $iteration < EC$ do
      $mutation\_count \leftarrow 0$
      while $mutation\_count < MM$ do
        $mutation\_count += 1$
      {First Mutation}
        if $mutation\_count = 1$ then
          $allmutants \leftarrow generate\_all\_single\_mutants(seq)$
          $scoredmutants \leftarrow pLM(allmutants)$
          $lastmutants = scoredmutants$
        end if
      {Subsequent Mutation}
        if $mutation\_count > 1$ and $mutation\_count < MM$ then
          $allmutants \leftarrow generate\_extra\_mutants(lastmutants)$
          $extramutants \leftarrow intsampling(allmutants, IST)$
          $scoredmutants \leftarrow pLM(extramutants)$
          $lastmutants = scoredmutants$
        end if
      {Final Mutation}
        if $mutation\_count = MM$ then
          $allmutants \leftarrow generate\_extra\_mutants(lastmutants)$
          $extramutants \leftarrow intsampling(allmutants, IST)$
          $scoredmutants \leftarrow pLM(extramutants)$
          $mutation \leftarrow final\_sampler(scoredmutants, final\_threshold)$
        end if
      end while
      $seq \leftarrow get\_mutated\_sequence(seq, mutation)$
      $iteration += 1$
    end while
    $generated\_sequences.append(seq)$
  end while
  $df \leftarrow pd.DataFrame(generated\_sequences)$
  $save\_as\_fasta(df, output\_path)$ {Output FASTA file}
```