Curriculum-Augmented GFlowNets For mRNA Sequence Generation

Aya Laajil[*] **Abduragim Shtanchaev, Sajan Muhammad, Eric Moulines, Salem Lahlou**

Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE

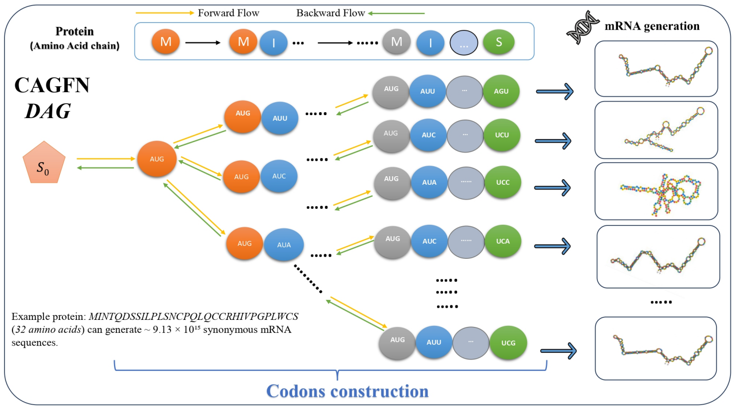
Abstract

Designing mRNA sequences is a major challenge in developing next-generation therapeutics, since it involves exploring a vast space of possible nucleotide combinations while optimizing sequence properties like stability, translation efficiency, and protein expression. While Generative Flow Networks are promising for this task, their training is hindered by sparse, long-horizon rewards and multi-objective trade-offs. We propose **Curriculum-Augmented GFlowNets (CAGFN)**, which integrate curriculum learning with multi-objective GFlowNets to generate *de novo* mRNA sequences. CAGFN integrates a length-based curriculum that progressively adapts the maximum sequence length guiding exploration from easier to harder subproblems. We also provide a new **mRNA design environment** for GFlowNets which, given a target protein sequence and a combination of biological objectives, generates plausible mRNA candidates encoding the same protein. This provides a biologically motivated setting for applying and advancing GFlowNets in mRNA sequence design. On different mRNA tasks, CAGFN improves Pareto performance and biological plausibility, while maintaining sequence diversity. Moreover, CAGFN reaches higher-quality solutions faster than a GFlowNet trained with random sequence sampling (no curriculum), and enables generalization to out-of-distribution sequences.

1 Introduction

Imagine a molecule that can be designed to instruct human cells to produce a protein of interest. Such is the promise of messenger RNA (mRNA), which has become a cornerstone of modern biotechnology [Pardi et al., 2018, Sahin et al., 2014]. Designing *de novo* mRNA sequences, that encode a target protein and achieve optimality on particular properties of interest [Gustafsson et al., 2004, Kane, 1995, Mauger et al., 2019], is therefore of growing practical importance. This task can be framed as generating sequences under multiple, often competing objectives, which makes search and optimization challenging [Keeney and Raiffa, 1993, Zhang et al., 2023, Angermueller et al., 2020]. Because biological targets are diverse and downstream outcomes are difficult to predict, diversity is a central design criterion [Mullis et al., 2019]. This need is amplified by the limited predictive power of inexpensive screening methods, such as *in-silico* simulations or *in vitro* assays. To maximize the likelihood that at least one candidate proves successful, it is important that the proposed sequences broadly cover the different modes of the underlying "goodness" function that estimates future success. Due to codon redundancy (see Appendix A.1), the number of synonymous nucleotide sequences that encode a protein grows exponentially with its length, and they differ dramatically in biological properties. For example, the SARS-CoV-2 spike protein ($\sim$1,273 amino acids) has on the order of 10^{632} possible synonymous mRNA sequences, most of which are suboptimal for practical use [Zhang et al., 2023]. Therefore, generative methods that can efficiently

[*]Corresponding author. Email : `ayalaajil132002@gmail.com`



Example protein: *MINTQDSSILPLSNCPQLQCCRHIVPGPLWCS* (*32 amino acids*) can generate ~ 9.13×10^{15} synonymous mRNA sequences.

Figure 1: **Directed acyclic graph (DAG)** representation of the mRNA sequence design process. Each node corresponds to a codon selection step, and edges represent possible transitions, illustrating the sequential construction of mRNA sequences. This DAG (tree in this case) forms the basis for the GFlowNet to explore diverse and high-quality sequences.

explore this combinatorial landscape, propose diverse high-quality candidates, and expose trade-offs between objectives are therefore of significant scientific and practical value.

Generative Flow Networks (GFlowNets) learn policies that sample complete objects with probability proportional to a non-negative reward $R(x)$, making them well-suited for producing diverse, high-reward candidates instead of collapsing to a single mode [Bengio et al., 2021, Jain et al., 2022a, Roy et al., 2023]. Recent applications to molecules and biological sequences demonstrate their promise [Jain et al., 2022b, Zhu et al., 2023, Cretu et al., 2024]. A more detailed review of related work is provided in Appendix A.3.

Despite these advantages, training GFlowNets on mRNA sequences reveals two fundamental challenges. First, sequences are long: rewards are only observed at the terminal step, leading to extremely sparse credit assignment. Second, the design problem is inherently multi-objective: optimizing a fixed combination of objectives typically drives the search into a narrow region of the design space, obscuring alternative solutions that are valuable in practice.

Curriculum Learning [CL; Bengio et al., 2009] offers a natural strategy to mitigate these challenges by structuring training so the learner first masters simpler tasks before progressing to harder ones. Crucially, modern curricula do more than sort tasks by difficulty: they preferentially present tasks on which the model is making the fastest progress, i.e., where the slope of the learning curve is highest. For GFlowNets, this perspective addresses long-horizon difficulty by using an *adaptive, length-based curricula* that allocate tasks dynamically based on recent learning progress to focus training where it is most informative [Matiisen et al., 2019].

Contributions. We introduce **Curriculum-Augmented GFlowNets (CAGFN)** for a principled integration of CL with multi-objective GFlowNets for mRNA sequences. In the next sections:

- We present a new **mRNA-design environment** (compatible with *torchgfn* [Lahlou et al., 2023]) for mRNA sequence generation and illustrate the design process in Figure 1.
- We propose adaptive, length-based curricula that preferentially present tasks with high learning progress to accelerate training and stabilize credit assignment.
- We show that CAGFN improves training and generalizes across different protein inputs, demonstrating a scalable path toward generative modeling in such complex biological task.

Novelty. To our knowledge, this is the first work that combines CL with GFlowNets for mRNA sequence design. Our approach demonstrates generalization across different objective weights and protein sequences, suggesting its potential applicability to generate mRNA sequences for out-of-distribution protein inputs.

2 Background

mRNA sequence design. Messenger RNA (mRNA) is a linear polymer built from nucleotide alphabet $\Sigma = \{\mathrm{A}, \mathrm{U}, \mathrm{G}, \mathrm{C}\}$. Proteins are sequences of amino acids and each amino acid is encoded by a *codon*, a contiguous triplet of nucleotides. The standard genetic code maps the set of 64 codons onto 20 amino acids plus stop signals. Because several codons can encode the same amino acid (synonymous codons), the mapping is degenerate (see definition and examples in Appendices A.1). This redundancy creates a large combinatorial set of possible mRNA sequences that all encode the same protein, and motivates algorithmic search and generative approaches for this design task.

Design objectives. We aim to generate diverse high-quality mRNA sequences that preserve a target protein while exposing trade-offs between competing biological objectives (see Appendix A.2 for motivation). mRNA design is inherently multi-objective, we focus on three standard proxies: **CAI** (codon adaptation index), **MFE** (minimum free energy), and **GC content**, which together capture translation efficiency, stability, and expression-related properties [Sharp and Li, 1987, Zuker and Stiegler, 1981, Courel et al., 2019, Pardi et al., 2018, Sahin et al., 2014, Zhang et al., 2023].

3 Methodology

3.1 GFlowNet-based Generation of mRNA Sequences

We consider the problem of searching the space of codon sequences $x = (c_1, \ldots, c_L)$ that encode a target protein $a = (a_1, \ldots, a_L)$ to maximize a multi-objective reward $R(x)$ (details in Appendix A.5). Each codon $c_i \in \mathcal{C}(a_i)$ belongs to the set of synonymous codons for amino acid a_i, so the feasible space is $\mathcal{X} = \prod_{i=1}^{L} \mathcal{C}(a_i)$. Formal GFlowNet details appear in Appendix A.4.

To design mRNA sequences with GFlowNets, we introduce the **CodonDesignEnv** (compatible with *torchgfn*), see Appendix A.5 for details. Details of the two training regimes considered, unconditional training with a fixed objective weighting and conditional training over weight vectors, are deferred to the Appendix A.7. Algorithmic and implementation details are given in Appendix A.6.

3.2 Curriculum Learning

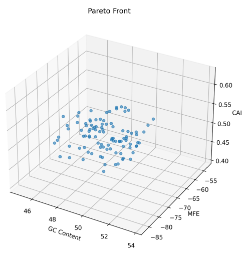
We introduce **Curriculum-Augmented GFlowNets (CAGFN)**, which combine Curriculum Learning with multi-objective GFlowNets to improve generalization across protein lengths. Practically, we partition training into length intervals and advance the curriculum according to a strategy that systematically adapts learning tasks of different difficulty based on GFlowNets learning. It allows the same parameterized policy to refine and extend learned representations as task complexity grows (see Algorithm 2 for more details).We use the following task set of amino-acid (AA) length intervals in our experiments:

$$\text{Tasks} \;=\; \{[l_1, l_2],\; [l_3, l_4],\; [l_5, l_6],\; [l_7, l_8],\; [l_9, l_{10}]\}\ \text{(AA)}, \tag{1}$$

Each pair(l_i, l_{i+1}) defines an interval of protein lengths from which we sample during training : given a sampled length, the corresponding protein sequence is then drawn from our pool of available proteins. See the formal recipe in Appendix A.8.1.

4 Experiments

Although our primary objective is conditional mRNA generation, we first conducted exploratory experiments in an unconditional setting to validate the core GFlowNet architecture. Using a MOGFN [Jain et al., 2023] with fixed weights, we generated sequences for a small protein task. The model produced diverse candidates with strong biological scores. For instance, the best generated sequence outperformed the natural counterpart from our dataset (Appendix A.9) across all metrics: GC content 53.78% vs. 45.33%, MFE -79.23 vs. -68.20, and CAI 0.63 vs. 0.54. The Pareto front and reward distributions are shown in Figures 2a and 2b. While unconditional generation demonstrates the validity of GFlowNets, real-world design requires adapting to varying objective weights. We therefore focus on conditional GFlowNets for the remainder of this work. In a small-scale comparison, we evaluated an unconditional model against a conditional GFlowNet where weights were sampled from *Dirichlet*$(1, 1, 1)$ during training. With fixed evaluation weights, the conditional model achieved superior results (Figure 3).


(a)

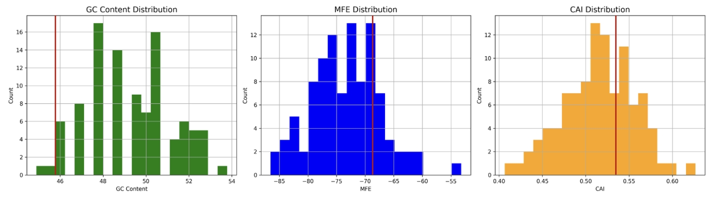

(b)

Figure 2: Unconditional mRNA sequence generation under fixed objective weights $[0.3, 0.3, 0.4]$. **(a)** Pareto front. **(b)** Distribution of reward metrics: The spread demonstrates that the model explores diverse regions of the design space rather than collapsing to a single mode. Vertical lines represent the natural sequence scores.

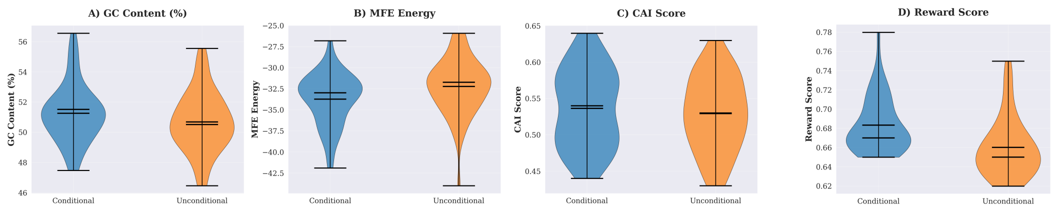


Figure 3: **Conditional Vs Unconditional mRNA generation results.** Metrics distribution across mRNA sequences of a small protein of interest (∼35AA). More details in Figure 7 of Appendix A.8.2.

Comparison with Reinforcement Learning. We compare against RL baselines MOReinforce [Lin et al., 2022] and PPO [Schulman et al., 2017]. We do not include MARS or traditional MCMC: MARS is tailored to small-molecule local-mutation optimization and does not map well to our multi-objective mRNA generation, while MCMC methods scale poorly in astronomically large combinatorial spaces and tend to produce low-diversity, locally concentrated solutions [Terayama et al., 2021, Pyzer-Knapp, 2018].

Our GFlowNet approach outperformed RL baselines in both diversity and Pareto-quality (Table 5). We found that MOReinforce and PPO suffered mode collapse and produced highly repetitive and trivial sequences, whereas GFlowNet-based models produce high-quality and diverse sequences [Lin et al., 2022, Schulman et al., 2017, Jain et al., 2023].

CL further improved training: loss and sampling dynamics (Figures 9, 8 in the Appendix) across training regimes (presented in Appendix A.8.2) indicate that CL increases sample efficiency and stabilizes optimization for longer, more complex proteins. On unseen proteins, CAGFN is competitive with, and often superior to, the Short-Only specialist on small proteins, and it matches or exceeds all baselines on medium-length proteins (Table 1). Comparison with the Random-Order baseline shows that a structured, length-based curriculum improves Pareto efficiency by enabling hierarchical, generalizable codon-selection representations.

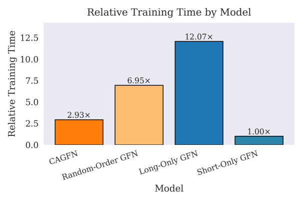


Figure 4: Training differences. CAGFN trains 4× faster than LGFN and 2.4× faster than ROFN.

Method	Reward (↑)	Uniqueness (↑)	Diversity (↑)
MOReinforce	0.7	1	19
PPO	0.51 (±0.02)	12	30
MOGFN (w/o CL)	0.59 (±0.064)	**100**	**31**
CAGFN	0.59 (±0.066)	**100**	**31**

Figure 5: Averaged metrics across 100 generated sequences for a small protein task.

Results for a medium-length protein task, shown in Figure 10 of Appendix A.8.2, further confirm that CAGFN shifts distributions toward more preferable values.

Table 1: TopK (K = 50) Reward and Diversity across generated sequences for different types of protein sequences. We use the Top-K Diversity and Top-K Reward metrics [Bengio et al., 2021]. Standard deviations of sample scores are shwon between parentheses.

	TopK Reward (↑)					TopK Diversity (↑)				
Model	0	1	2	3	4	0	1	2	3	4
(25–60 AA)										
SGFN	0.58 (±0.01)	0.52 (±0.05)	0.55 (±0.02)	0.64 (±0.02)	0.67 (±0.02)	24 (±3.2)	32 (±3.1)	37 (±3.0)	40 (±3.5)	36 (±3.3)
CAGFN	0.61 (±0.06)	**0.58** (±0.05)	**0.62** (±0.05)	0.69 (±0.06)	**0.73** (±0.01)	25 (±3.3)	32 (±3.2)	38 (±3.8)	**41** (±3.6)	37 (±4.0)
ROGFN	0.58 (±0.02)	0.53 (±0.02)	0.55 (±0.02)	0.67 (±0.02)	0.70 (±0.02)	**25** (±3.3)	**32** (±3.4)	**38** (±3.2)	40 (±3.5)	37 (±3.3)
LGFN	**0.62** (±0.03)	0.57 (±0.03)	0.61 (±0.02)	**0.70** (±0.03)	**0.73** (±0.03)	24 (±3.1)	30 (±3.2)	36 (±3.3)	38 (±3.5)	35 (±3.3)
(85–120 AA)										
SGFN	0.55 (±0.03)	0.64 (±0.02)	0.69 (±0.02)	0.71 (±0.02)	0.65 (±0.03)	37 (±3.0)	**40** (±3.2)	**36** (±3.7)	34 (±3.4)	39 (±3.1)
CAGFN	0.55 (±0.06)	0.65 (±0.06)	0.70 (±0.05)	0.71 (±0.08)	**0.72** (±0.01)	37 (±3.2)	**40** (±3.3)	**36** (±3.9)	34 (±3.5)	**40** (±3.2)
ROGFN	0.56 (±0.02)	0.67 (±0.02)	0.70 (±0.02)	**0.72** (±0.02)	0.66 (±0.02)	38 (±3.4)	**40** (±3.3)	**36** (±3.2)	**36** (±3.7)	39 (±3.0)
LOGFN	**0.60** (±0.03)	**0.70** (±0.03)	**0.73** (±0.02)	0.69 (±0.03)	**0.72** (±0.03)	37 (±3.1)	38 (±3.2)	34 (±3.5)	35 (±3.4)	38 (±3.3)

Indexes 0–4 denote protein task indices.

Table 2: Pareto Performance over 50 generated sequences across models.

	Pareto Performance (↑)				
Model	0	1	2	3	4
(25–60 AA)					
SGFN	0.08	0.06	0.09	0.16	0.19
Random-Order GFN	**0.13**	**0.15**	0.17	0.17	0.21
CAGFN	0.07	0.12	**0.21**	**0.21**	**0.22**
Long-Only GFN	0.05	0.13	0.21	0.21	0.20
(85–120 AA)					
SGFN	0.05	0.21	0.13	0.15	0.16
Random-Order GFN	0.09	0.11	**0.28**	0.19	**0.22**
CAGFN	**0.15**	**0.21**	0.19	**0.21**	0.19
LGFN	0.05	0.13	0.13	0.21	0.20

Indexes 0–4 denote protein task indices.

CAGFN, in table 2 shows a good coverage of the Pareto front (up to 22 sequences) across proteins.

OAZ2 protein. We also studied the performance on one particular protein, *Ornithine decarboxylase Antizyme 2 (OAZ2)*. OAZ2 is a protein that helps regulate the levels of certain small molecules called *polyamines* inside human cells. It ensures that the cell keeps the right balance of them and doesn't make too much.
We present in Figure 11 in the Appendix the distribution of reward objectives across the four models. The results reveal that our CAGFN consistently matches or exceeds baselines, and generates high-quality mRNA sequences.

5 Conclusion

Motivated by the growing need for efficient and reliable mRNA therapeutics and vaccines, we addressed the problem of mRNA sequence design. The task focuses on optimizing codon sequences that encode the same protein but differ in biological properties. We introduced **Curriculum-Augmented Generative Flow Networks (CAGFN)**, a multi-objective GFlowNet enhanced with a teacher-driven curriculum. Building on a MOGFN formulation, we designed a TSCL-style teacher that schedules tasks according to learning progress, enabling the model to generate diverse and high-quality mRNA sequences across proteins of varying lengths. Our experiments showed that CAGFN consistently improves upon non-curriculum GFlowNet baselines by achieving higher rewards, better objective trade-offs, faster convergence, and broader Pareto-front coverage.

Reproducibility statement

To ensure complete reproducibility of our results, we provide implementation details and make all resources publicly available in `https://github.com/ayalaajil/GFN_for_mRNA_design`. The complete source code includes the different training modes and the dataset we used for protein/mRNA task generation. The algorithms used are outlined in the appendices. The final set of hyperparameters used to reproduce the reported results are discussed in Appendix A.9, and provided as default values in our code. Additionally, we provide evaluation scripts for the comparison validation and comprehensive testing procedures that allow researchers to reproduce all experimental results reported in Section 4.

References

Norbert Pardi, Matthew J Hogan, Frederick W Porter, and Drew Weissman. mrna vaccines — a new era in vaccinology. *Nature Reviews Drug Discovery*, 17(4):261–279, 2018.

Ugur Sahin, Katalin Karikó, and Ozlem Türeci. mrna-based therapeutics — developing a new class of drugs. *Nature Reviews Drug Discovery*, 13(10):759–780, 2014.

Claes Gustafsson, Sridhar Govindarajan, and Jeremy Minshull. Codon bias and heterologous protein expression. *Trends in biotechnology*, 22(7):346–353, 2004.

James F Kane. Effects of rare codon clusters on high-level expression of heterologous proteins in escherichia coli. *Current opinion in biotechnology*, 6(5):494–500, 1995.

David M Mauger, Benjamin J Cabral, Valeria Presnyak, Shengdong Su, Daniel W Reid, Ben Goodman, Kevin Link, Nikhil Khatwani, Jean-Paul Reynders, Daniel Esposito, et al. mrna structure regulates protein expression through stability and translation efficiency. *Nature Structural & Molecular Biology*, 26(10):1089–1097, 2019.

Ralph L Keeney and Howard Raiffa. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press, 1993.

He Zhang, Liang Zhang, Ang Lin, Congcong Xu, Ziyu Li, Kaibo Liu, Boxiang Liu, Xiaopin Ma, Fanfan Zhao, Huiling Jiang, et al. Algorithm for optimized mrna design improves stability and immunogenicity. *Nature*, 621(7978):396–403, 2023.

Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Population-based black-box optimization for biological sequence design. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 52–62, 2020.

Megan M Mullis, Ian M Rambo, Brett J Baker, and Brandi Kiel Reese. Diversity, ecology, and prevalence of antimicrobials in nature. *Frontiers in microbiology*, 10:2518, 2019.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

Moksh Jain, Emmanuel Bengio, Alex Hernández-García, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with gflownets. *Proceedings of Machine Learning Research (PMLR), ICML 2022*, 162:9786–9801, 2022a. URL `https://proceedings.mlr.press/v162/jain22a.html`.

Julien Roy, Pierre-Luc Bacon, Christopher Pal, and Emmanuel Bengio. Goal-conditioned gflownets for controllable multi-objective molecular design. *arXiv preprint*, 2023. URL `https://arxiv.org/abs/2306.04620`.

Meher Jain, Eric Bengio, Alejandro Hernandez-Garcia, Joshua Rector-Brooks, Benjamin F. Dossou, Chaitanya A. Ekbote, ..., and Yoshua Bengio. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, June 2022b.

Yiheng Zhu, Jialu Wu, Chaowen Hu, Jiahuan Yan, Tingjun Hou, Jian Wu, et al. Sample-efficient multi-objective molecular optimization with gflownets. *Advances in Neural Information Processing Systems*, 36:79667–79684, 2023.

Miruna Cretu, Charles Harris, Julien Roy, Emmanuel Bengio, and Pietro Liò. Synflownet: Towards molecule design with guaranteed synthesis pathways. In *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2024.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3732–3740, 2019.

S. Lahlou, J. D. Viviano, V. Schmidt, and Y. Bengio. torchgfn: A pytorch gflownet library. *arXiv preprint arXiv:2305.14594*, 2023.

P. M. Sharp and W. H. Li. The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic acids research*, 15(3):1281–1295, 1987.

M. Zuker and P. Stiegler. Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1):133–148, 1981.

Marie Courel, Yohann Clément, Camille Bossevain, Dominik Foretek, Olivier Vidal Cruchez, Zheng Yi, Marion Bénard, Marie-Noëlle Benassy, Michaël Kress, Chloé Vindry, Michèle Ernoult-Lange, Christophe Antoniewski, Antonin Morillon, Patrick Brest, Alexis Hubstenberger, Hugues Roest Crollius, Nancy Standart, and Dominique Weil. Gc content shapes mrna storage and decay in human cells. *eLife*, 8, 2019. doi: 10.7554/eLife.49708. URL `https://doi.org/10.7554/eLife.49708`.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-García, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. In *International Conference on Machine Learning*, pages 14631–14653. PMLR, July 2023.

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. *arXiv preprint arXiv:2203.15386*, 2022.

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

Kei Terayama, Masato Sumita, Ryo Tamura, and Koji Tsuda. Black-box optimization for automated discovery. *Accounts of Chemical Research*, 54(6):1334–1346, 2021.

Edward O Pyzer-Knapp. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development*, 62(6):2–1, 2018.

Yoshua Bengio, Sami Lahlou, Thomas Deleu, Eugene J. Hu, Mohit Tiwari, and Eric Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

Michał Koziarski, Andrei Rekesh, Dmytro Shevchuk, Almer van der Sloot, Piotr Gaiński, Yoshua Bengio, Chenghao Liu, Mike Tyers, and Robert Batey. Rgfn: Synthesizable molecular generation using gflownets. *Advances in Neural Information Processing Systems*, 37:46908–46955, 2024.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*, 2022. URL `https://arxiv.org/abs/2201.13259`.

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pages 23467–23483. PMLR, 2023.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

A. Fallahpour, V. Gureghian, G. J. Filion, A. B. Lindner, and A. Pandi. Codontransformer: a multispecies codon optimizer using context-aware neural networks. *Nature Communications*, 16(1): 3205, 2025. doi: 10.1038/s41467-025-XXXXX.

Benjamin D. Lee. Python implementation of codon adaptation index. *Journal of Open Source Software*, 3(30):905, oct 2018. doi: 10.21105/joss.00905. URL `https://doi.org/10.21105/joss.00905`.

A Appendix

A.1 mRNA Design Problem

Codon Redundancy. The genetic code is redundant: multiple codons can encode the same amino acid. For example, the amino acid **Leucine (Leu)** is represented by six codons (UUA, UUG, CUU, CUC, CUA, CUG), while **glycine** has four (GGU, GGC, GGA, GGG). This allows different mRNA sequences to yield the same protein. As an illustration, a short peptide Met–Leu–Gly may be encoded by the mRNA sequence `AUG--UUA--GGU` or alternatively by `AUG--CUG--GGC`, both producing the identical protein despite differing at the nucleotide level. Exploiting codon redundancy is central to mRNA sequence design, as it provides a vast combinatorial space in which sequences can be optimized for multiple objectives while preserving the encoded protein.

To clarify the nature of the mRNA design problem, we provide a simple example that highlights the role of codon redundancy and the search space of feasible mRNA sequences.

Example : A short protein sequence. Consider designing an mRNA sequence for a short protein consisting of three amino acids: Methionine–Phenylalanine–Lysine **(Met–Phe–Lys)**.

- Met (M) can only be encoded by the codon `AUG`.
- Phe (F) can be encoded by `UUU` or `UUC`.
- Lys (K) can be encoded by `AAA` or `AAG`.

The feasible design space for this protein is therefore:

$$\{\texttt{AUG UUU AAA}, \texttt{AUG UUU AAG}, \texttt{AUG UUC AAA}, \texttt{AUG UUC AAG}\}.$$

All four codons sequences produce the same protein, but differ in codon usage patterns. Choosing among them involves evaluating biological properties such as GC content, minimum free energy (MFE), and codon adaptation index (CAI). This simple case illustrates the exponential growth of the design space as protein length increases, underscoring the need for generative models like GFlowNets to explore it effectively.

A.2 Motivation for diversity

Diversity is essential in biological sequence design because targets, mechanisms of action, and structural contexts are highly heterogeneous and often change over time [Mullis et al., 2019]. Cheap preclinical screens (in silico or in vitro) are imperfect proxies for eventual performance in wet-lab experiments, so concentrating effort on a single 'best' sequence risks failure if that sequence aligns with biases or blind spots of the surrogate assays. By contrast, proposing a diverse set of high-quality candidates increases the chance that at least one design occupies a different mode of the true, and unknown, success landscape, providing robustness to model misspecification and experimental noise. Practically, this is critical when searching astronomical, combinatorial spaces (e.g., on the order of 10^{60} possible sequences) under tight experimental budgets: diversity-aware methods trade breadth for targeted exploration of multiple promising modes, improving the probability of downstream success and accelerating the overall design cycle [Pyzer-Knapp, 2018, Terayama et al., 2021]. Generative models that can exploit the combinatorial structure in these spaces have the potential to speed up the design process for such sequences.

A.3 Related work

GFlowNets are probabilistic generative models for sampling complex objects via sequential decisions. Unlike conventional RL that seeks a single optimal solution, GFlowNets train a policy to sample complete objects (e.g., sequences or graphs) with a probability proportional to a reward function. This proportional sampling intrinsically promotes diversity and allows GFlowNets to amortize the cost of sampling in large combinatorial spaces while representing distributions over composite structures [Bengio et al., 2021, 2023, Jain et al., 2022b, 2023]. Empirically, GFlowNets have been applied to a range of scientific design tasks. For example, Jain et al. [2022b] embed a GFlowNet in an active learning loop to generate diverse candidate peptides and find more novel high-scoring sequences than previous methods. In drug discovery, several works extend GFlowNets to enforce domain constraints: Reaction-GFN [Koziarski et al., 2024] constrains generation to chemical reaction steps so that every molecule has a valid synthetic route, and SynFlowNet [Cretu et al., 2024] restricts actions to documented reactions and purchasable starting materials to produce synthetically accessible molecules while preserving diversity. These applications demonstrate that GFlowNets can flexibly incorporate domain-specific action spaces to generate complex, structured objects. Many design problems involve multiple, often conflicting objectives. Jain et al. [2023] have extended GFlowNets to handle multi-objective settings: they propose Multi-Objective GFlowNets (MOGFNs) to directly sample diverse Pareto-optimal solutions, and show in diverse experiments that MOGFNs achieve improved Pareto-front coverage and candidate diversity compared to standard RL baselines. Similarly, Zhu et al. [2023] integrate GFlowNets into a multi-objective Bayesian optimization framework via a hypernetwork-based GFlowNet (HN-GFN) that generates batches of molecules across different trade-offs, effectively sampling an approximate Pareto front. These works highlight that GFlowNets can be conditioned on objective preference vectors to explore trade-off spaces instead of collapsing to a single greedy objective.

GFlowNets have shown promise in molecular and peptide design, but their application to mRNA sequence design has not yet been explored and introduces distinct domain constraints. Although mRNA is a biological sequence, its design introduces unique objectives compared to typical molecular-generation tasks: codon choice (synonymous encodings), secondary-structure, and expression-related metrics must all be balanced. These domain-specific requirements make mRNA design a fundamentally different combinatorial problem from classical small-molecule generation, and they motivate adapting multi-objective GFlowNet methods to this setting.

The considerable length and structural complexity of mRNA sequences pose additional challenges for training GFlowNets, particularly with respect to credit assignment and sparse rewards. Curriculum learning addresses this by presenting tasks of increasing difficulty so that the model can master simpler subtasks first. Bengio et al. [2009] originally advocated for curriculum learning by training neural models on easier examples before harder ones. A prominent formalism in RL is Teacher–Student Curriculum Learning [TSCL; Matiisen et al., 2019], where a teacher dynamically selects tasks on which the student shows the fastest learning progress. In TSCL, the teacher monitors per-task learning curves and focuses on tasks with the steepest improvement (highest slope), which lets training rapidly move up in difficulty as the agent improves and has solved environments that were intractable under uniform sampling. The hypothesis we validate in this work is that applying an adaptive, learning-progress-driven curriculum to GFlowNets stabilizes credit assignment and accelerates convergence for long, sparse-reward sequence-design problems.

Taken together, these strands motivate our approach. We adopt the MOGFN formulation as the base model for our curriculum-augmented GFlowNet (CAGFN) to (i) generate diverse, Pareto-aware mRNA candidates and (ii) exploit a teacher-driven curriculum that progressively adapts sequence length during training based on measured learning progress. This combination addresses both the multi-objective nature of mRNA design and the training challenges posed by the length of these sequences.

A.4 Background on GFlowNets

GFlowNets (GFNs) are a family of probabilistic models that learn a stochastic policy to generate compositional objects, such as a graph describing an mRNA sequence, through a sequence of steps, with probability proportional to their reward $R(x) \geq 0$ [Bengio et al., 2021].

We consider a directed acyclic graph $(\mathcal{S}, \mathcal{A})$, where nodes $\mathcal{S}$ represent states and edges $\mathcal{A}$ represent actions. If $(s, s') \in \mathcal{A}$, we say that s is a parent of s' and that s' is a child of s. There is a unique state s_0 with no parents that we name *source state* and a unique state s_f with no children that we name *sink state*. We refer to the parents of the *sink state* as terminating states $\mathcal{X} = \{s \in \mathcal{S} \mid (s, s_f) \in \mathcal{A})\}$. A complete trajectory $\tau = (s_0, s_1, ..., s_n, s_{n+1} = s_f)$ is a sequence of states that start with the *source state* and ends with the *sink state*.

We also consider a reward function $R : \mathcal{X} \to \mathbf{R}$ such as $\forall x \in \mathcal{X}, R(x) > 0$. By convention, we can also consider that the reward for non terminating states is zero $\forall s \in \mathcal{S} \setminus \mathcal{X}, R(s) = 0$. The goal of the GFlowNet is to learn to sample terminating states proportionally to the reward function $P_T(x) \propto R(x)$. To achieve this, GFlowNets start from the initial state (for example, empty sequence or molecule in biological setting, initial coordinates in hyper-grid environment...) and iteratively sample actions to move to the following states.

Formally, we consider a non-negative flow function $F : \mathcal{A} \to \mathbf{R}$. The goal of the GFlowNet framework is to learn such a flow function that satisfies the following flow matching (FM) and reward matching constraints, for all $s' \neq s_0, s_f$ and for all $x \in \mathcal{X}$, respectively:

$$\sum_{(s,s')\in\mathcal{A}} F(s \to s') = \sum_{(s',s'')\in\mathcal{A}} F(s' \to s''), \tag{2}$$

$$F(x \to s_f) = R(x). \tag{3}$$

We extend the definition of the flow function to states:

$$\forall s \in \mathcal{S}, \quad F(s) = \sum_{(s,s')\in\mathcal{A}} F(s \to s'). \tag{4}$$

We also define the forward and backward policies as follow:

$$P_F(s' \mid s) = \frac{F(s \to s')}{F(s)}, \;\; P_B(s \mid s') = \frac{F(s \to s')}{F(s')}. \tag{5}$$

Trajectory Balance (TB). Given a complete trajectory/episode $\tau = (s_0 \to s_1 \to \ldots \to s_n = x)$ that terminates in x, the trajectory balance objective enforces a global balance using a learned normalizing constant $Z > 0$ [Malkin et al., 2022]:

$$\mathcal{L}_{\text{TB}}(\tau) \;=\; \left(\log \frac{Z \prod_{t=0}^{n-1} P_F(s_{t+1} \,|\, s_t)}{R(x) \prod_{t=0}^{n-1} P_B(s_t \,|\, s_{t+1})}\right)^2. \tag{6}$$

Minimizing this loss across sampled trajectories ensures that $Z \prod_t P_F(\cdot) \approx R(x) \prod_t P_B(\cdot)$, which leads to sampling proportional to $R(x)$ under appropriate parameterizations.

Sub-trajectory TB (SubTB). To enable learning from partial episodes, SubTB [Madan et al., 2023] applies a TB-like constraint to partial trajectories $\tau_{0:k} = (s_0, \ldots, s_k)$:

$$\mathcal{L}_{\text{SubTB}}(\tau_{0:k}) \;=\; \left(\log \frac{F(s_0) \prod_{t=0}^{k-1} P_F(s_{t+1} \,|\, s_t)}{F(s_k) \prod_{t=0}^{k-1} P_B(s_t \,|\, s_{t+1})}\right)^2, \tag{7}$$

where $F(s)$ is the learned flow at state s.

In this work, we conducted our experiments using TB and SubTB losses for training GFlowNets. SubTB is particularly well-suited for our curriculum learning approach, as it naturally accommodates training on sequences of varying lengths and improves convergence for long mRNA sequences compared to standard TB loss.

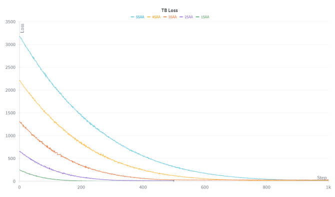
Loss Curves During Training We found that the SubTB loss function is critical for maintaining numerical stability for sequences longer than approximately 55 amino acids, as the standard TB loss consistently failed. All subsequent models are built upon these two findings.

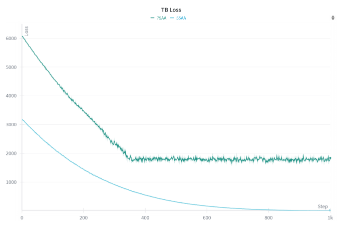
We provide additional loss curves in Figures 6a–6c that compare the standard TB loss with the SubTB loss on sequences of length up to 75 amino acids (AA). The results clearly illustrate that TB fails to converge, with the loss plateauing at high values and exhibiting unstable behavior. In contrast, SubTB consistently converges to a stable solution, demonstrating its robustness for longer sequences.

Furthermore, we investigated whether the convergence issues with TB could be alleviated by changing model architectures or tuning hyperparameters. Specifically, we trained the model using MLP, LSTM, and Transformer architectures across a wide range of learning rates, batch sizes, and hidden dimensions. Across all settings, the TB loss exhibited the same instability, confirming that the problem is inherent to the TB formulation rather than to the architectural choice. These findings motivated our decision to adopt SubTB for all subsequent experiments.

A.5 mRNA Codon Design Environment

To exploit codon redundancy in protein coding, we introduce the **CodonDesignEnv**, a discrete environment for codon-level mRNA design given a protein sequence. The environment models mRNA construction as a sequential decision process: states represent partially constructed sequences, actions correspond to choosing codons, and trajectories terminate once a full sequence encoding the target protein is completed.


(a) Trajectory Balance (TB) loss on small sequences (<75AA).


(b) Trajectory Balance (TB) loss on a 75AA sequence.

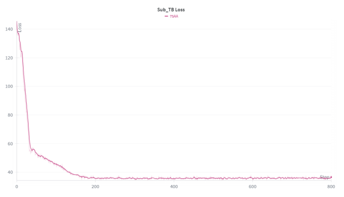

(c) SubTB loss (75AA). Converges stably.

Figure 6: Sub-Trajectory Balance (SubTB) loss on 75AA sequences. Unlike TB, SubTB converges smoothly and remains numerically stable.

State Space ($\mathcal{S}$). Each state $s \in \mathcal{S}$ is represented as a fixed-length integer vector of codon indices of size L (protein length). Unassigned positions are denoted by -1, and the initial state is

$$s_0 = (-1, -1, \ldots, -1).$$

A state becomes terminal once all L positions are filled with valid codons, yielding a complete mRNA sequence x.

Action Space. The action set consists of all 64 codons plus a special "exit" action a_{exit}, i.e. $|\mathcal{A}| = 65$. At step t, the model selects a codon for the t-th amino acid in the target protein. Upon completion ($t = L$), the only valid action is a_{exit}, which transitions the environment to a terminal sink state s_f.

Dynamic Masking Strategy. To ensure biological validity, the environment uses dynamic action masking:

- **Synonymous codon masking:** At step $t < L$, the forward action set $\mathcal{A}_f(s)$ is restricted to codons encoding the $(t+1)$-th amino acid.
- **Termination masking:** At step $t = L$, only a_{exit} is permitted, forcing proper termination.
- **Backward masking:** For backward sampling, only the most recently added codon can be removed, preserving sequential construction.

These constraints define a directed acyclic graph (DAG) where each path from s_0 to s_f corresponds to a valid mRNA sequence.

Reward Function. Rewards are assigned only to terminal states and capture biologically motivated objectives (§2): GC content, minimum free energy (MFE), and codon adaptation index (CAI). Beyond our chosen objectives (GC, MFE and CAI), the framework is flexible and can incorporate alternative or additional design objectives.

For a complete mRNA sequence x, the reward is

$$R(x) = w^{\top}\phi(x),$$

where $\phi(x)$ is a vector of normalized objectives and $w \in R^3$ are user-defined weights. By adjusting w, one can bias the GFlowNet toward different Pareto-optimal trade-offs, promoting diversity and mode coverage across the design space.

While we adopt the weighted-sum formulation in this work for simplicity and interpretability, the same framework is compatible with alternative scalarization schemes. More generally, our approach follows the idea of *preference-conditional GFlowNets* [MOGFN-PC; Jain et al., 2023], in which a single reward-conditional GFlowNet models the entire family of multi-objective optimization subproblems simultaneously. Here, preferences $\omega \in \Delta^d$ over the set of objectives $\{R_1(x), \ldots, R_d(x)\}$ act as conditioning variables, and the reward function is defined via a scalarization $R(x \mid \omega)$. This general formulation accommodates any scalarization function, whether classical weighted sums, max–min formulations, or novel functions designed for specific biological contexts. For example, alternatives such as the weighted-log-sum scalarization have been proposed to mitigate cases where one objective dominates the reward signal.

A.6 GFlowNet Training Loop Algorithm

This section provides the detailed pseudocode for the GFlowNet Training Loop.

Algorithm 1 GFlowNet Training Loop

```
Require: $\mathcal{E}$ (env), $\pi_\theta$ (policy), $R(\cdot)$ (reward), $B$ (batch), $I$ (iters), $\mathcal{O}$ (opt), $\boldsymbol{\alpha}$
1: for $i \in \{1, \ldots, I_{\text{total}}\}$ do
2:     $\boldsymbol{w} \leftarrow$ NONE
3:     if conditional then
4:         $\boldsymbol{w} \sim \text{Dirichlet}(\boldsymbol{\alpha})$
5:     $\mathcal{T}_{1:B} \sim \pi_\theta(\cdot \mid \mathcal{E}, \boldsymbol{w})$                    ▷ Sample $B$ paths
6:     $r_{1:B} \leftarrow R(\mathcal{T}_{1:B})$
7:     $\mathcal{L} \leftarrow \text{Loss}(\mathcal{E}, \mathcal{T}_{1:B}, r_{1:B})$                    ▷ Eq. 7
8:     $\theta \leftarrow \theta - \mathcal{O}(\nabla_\theta \mathcal{L})$
9: return $\pi_\theta$
```

A.7 Conditional vs Unconditional Generation

The environment models mRNA sequence construction as a trajectory through states $s_t = (c_1, \ldots, c_t)$, s_0 is the empty sequence and s_t a partially constructed mRNA sequence. A terminal state $x = s_L$ corresponds to a complete mRNA sequence encoding the target protein. The action space consists of the 64 codons plus a special exit action, yielding $|\mathcal{A}| = 65$. Transitions are deterministic and constrained by the genetic code. At step $t < L$, the forward action set is dynamically masked to only allow synonymous codons for the $(t+1)$-th amino acid. At $t = L$, only the exit action is valid, forcing termination. These dynamic masks ensure that every trajectory from s_0 to the terminal sink state s_f corresponds to a valid mRNA sequence, and preserve the encoding of the original protein sequence. The resulting state-action graph is a directed acyclic graph (DAG) 1 aligned with the codon redundancy of the genetic code, providing a natural fit for flow-based generative modeling with GFlowNets. Rewards are assigned only to terminal states and capture biologically motivated objectives (§2). For a complete mRNA sequence x, the reward is $R(x) = w^\top \phi(x)$, where $\phi(x)$ is a vector of normalized objectives and $w \in R^3$ are objective weights. By adjusting w, one can bias the GFlowNet toward different Pareto-optimal trade-offs, promoting diversity and mode coverage across the design space. Our framework supports both (i) unconditional training, where the weights of the reward are fixed and define a single trade-off between objectives, and (ii) conditional training, where the desired objective weights are provided at training and inference, enabling controlled exploration of the Pareto front.

Unconditional Generation. In the unconditional setting, we fix the objective weight vector w during training so the GFlowNet learns a fixed reward function $R(x \mid w) = w^\top \phi(x)$. The model is trained (see Algorithm 1 in Appendix A.6) to assign flows over complete trajectories such that the sampling probability of the trajectory is proportional to its reward $\pi(x) \propto R_w(x)$, encouraging the sampler to visit high-reward regions while preserving mode coverage. At inference, we sample complete trajectories from the learned forward policy, producing a diverse pool of candidate mRNA sequences whose sampling probability reflects their composite reward under the chosen weights. This contrasts with optimization-only methods [Williams, 1992, Schulman et al., 2017, Holland, 1992, Snoek et al., 2012] that collapse to a single optimum.

Conditional Generation. As the biological objectives are often imperfect proxies for mRNA properties of interest, we aim to generate diverse Pareto-optimal solutions for each sub-problem $\max_{x \in \mathcal{X}} R(x \mid w)$. Also, fixing a single set of weights for the GFlowNet at a time prevents the model from exploiting the shared structure present between these related sub-problems. Bengio et al. [2021] extend standard GFlowNets to learn a single conditional policy that simultaneously models a family of distributions indexed by a conditioning variable. Each weight vector $w \in \mathcal{W}$ induces a different reward function $R(x \mid w)$ over terminal states $x \in \mathcal{X}$.The conditioning information w is then available at every state $s \in \mathcal{S}_w$. Exploiting the shared structure across w-conditioned reward functions allows a single conditional policy to model the entire family of sub-problems and, importantly, to generalize to different weight vectors w.

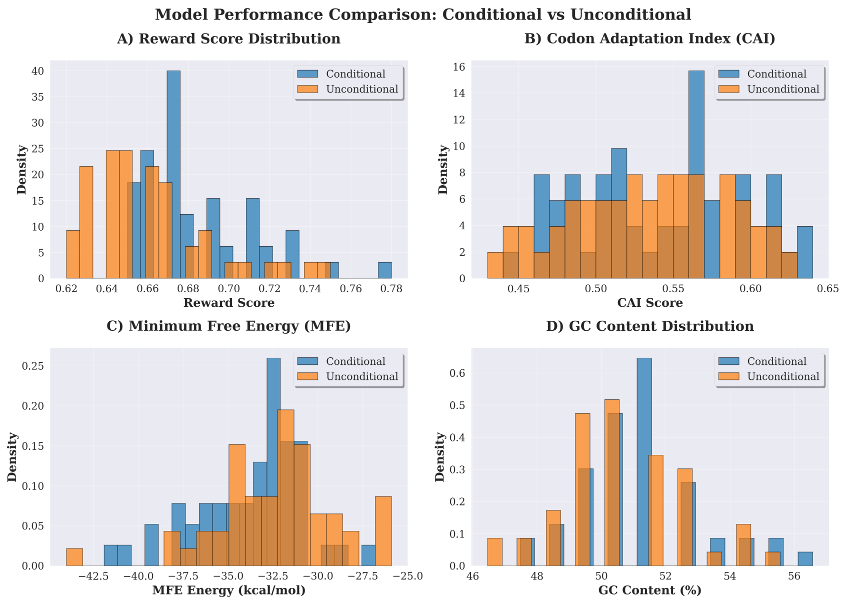


Figure 7: Metrics Distributions

A.8 Curriculum Learning Algorithm

A.8.1 Teacher-Student CL: a formal recipe

We follow the teacher-student (TSCL) paradigm [Matiisen et al., 2019] in which the Teacher maintains a per-task performance signal and uses *learning progress* to bias sampling of tasks. For each task index $i \in \{1, \ldots, K\}$ and discrete training step t define a per-task metric $m_{i,t}$. The Teacher forms a smoothed learning-progress estimate $\mathrm{LP}_{i,t}$ from the discrete changes $\Delta m_{i,t} = m_{i,t} - m_{i,t-1}$. A common and stable choice for the learning-progress estimate $\mathrm{LP}_{i,t}$ is an exponential moving average (EMA):

$$\mathcal{LP}_{i,t} \;=\; (1-\beta)\,\mathcal{LP}_{i,t-1} \;+\; \beta\,(m_{i,t} - m_{i,t-1}), \tag{8}$$

with smoothing hyperparameter $\beta \in (0, 1]$ (e.g. $\beta \approx 0.01$–0.1). The Teacher then converts LP to a sampling distribution over tasks. A robust, positivity-enforcing rule is:

$$P(i \mid t) \;=\; \frac{\max\big(0, \mathcal{LP}_{i,t}\big) + \varepsilon}{\sum_{j=1}^{K} \big(\max\big(0, \mathcal{LP}_{j,t}\big) + \varepsilon\big)}, \tag{9}$$

where $\varepsilon > 0$ is a small floor to ensure exploration. We further describe and compare different curriculum configurations in Appendix A.8.3.

Choice of per-task metric. The metric $m_{i,t}$ should reflect the Student's progress on task i. The design is robust to the specific metric as long as it (i) changes as the Student learns, and (ii) is reasonably smooth or smoothed by EMA to reduce noise. We choose for this work the average reward over generated sequences, as a per-task metric, since it reflects the performance of our GFlowNets (the Student model).

$\mathcal{LP}$-based selection. Selecting tasks with highest positive $\mathcal{LP}$ focuses training on regions where the Student still makes rapid gains, accelerating sample efficiency and directing compute to the frontier of competence. Negative $\mathcal{LP}$ (decline in performance) can be used to trigger replay and reduce forgetting [Matiisen et al., 2019]. This approach necessitates a dynamic training setup where the environment's constraints change with each task. We initialize a single, shared GFlowNet model. The central principle of our methodology is parameter sharing. The model does not learn a separate policy for each protein sequence, instead, it learns a single, conditional policy that can generalize across the entire task distribution. This forces the model to learn the underlying principles of codon selection that lead to high rewards, rather than memorizing solutions for specific proteins.

The complete procedure is detailed in the following Algorithm 2.

Algorithm 2 Curriculum Learning Algorithm

```
Require:
    $T = \{t_1, \dots, t_N\}$                    ▷ Pool of tasks sets where $t_i$ is a set tasks
    $S = \{S_1, \dots, S_N\}$                    ▷ Pool of protein sequence sets for each $t_i$
    $\pi_\theta$                                 ▷ GFlowNets Policy (Student)
    $\boldsymbol{w}_{\text{eval}}$               ▷ Fixed weights for eval
    $I_{\text{total}}, I_{\text{task}}, I_{\text{eval}}, \beta, \varepsilon$   ▷ Hyperparameters
 1: $H_k \leftarrow \emptyset$ for each task $t_k \in T$     ▷ Initialize history for Learning Progress
 2: $\mathcal{LP}_k \leftarrow 0$ for each task $t_k \in T$     ▷ Initialize Learning Progress for $t_i$
 3: $P \leftarrow \text{Uniform}(1/N)$
 4: for $i \in \{1, \dots, I_{\text{total}}\}$ do
 5:     $k \sim P$, $p_{\text{seq}} \sim S_k$     ▷ Sample task index $k$ and a protein
 6:     $\mathcal{E} \leftarrow \text{InitializeEnv}(p_{\text{seq}})$
 7:     for $s \in \{1, \dots, I_{\text{task}}\}$ do
 8:         $\boldsymbol{w} \sim \text{Dirichlet}(\boldsymbol{\alpha})$
 9:         $\mathcal{T} \sim \pi_\theta(\cdot|\mathcal{E}, \boldsymbol{w})$     ▷ Sample trajectory
10:         $\mathcal{L}_{\text{SubTB}} \leftarrow \text{CalculateLoss}(\mathcal{T})$
11:         $\theta \leftarrow \text{OptimizerStep}(\theta, \nabla_\theta \mathcal{L}_{\text{SubTB}})$
12:     if $i \in \{I_{\text{eval}}, 2I_{\text{eval}}, \dots\}$ then
13:         for each task $t_j \in T$ do
14:             $p_{\text{seq}} \sim S_j$
15:             $\mathcal{E}_{\text{eval}} \leftarrow \text{InitializeEnv}(p_{\text{seq}})$
16:             $\mathcal{T}_{eval} \sim \pi_\theta(\cdot|\mathcal{E}_{\text{eval}}, \boldsymbol{w}_{\text{eval}})$     ▷ Generate a batch of trajectories
17:             $r_j^{\text{new}} \leftarrow \sum_{\mathcal{T} \in \mathcal{T}_{eval}} R(\mathcal{T})/|\mathcal{T}_{eval}|$     ▷ Compute normalized reward
18:             $r_j^{\text{prev}} \leftarrow \begin{cases} H_{j=|H_j|}, & |H_j| > 0 \\ 0 & |H_j| = 0 \end{cases}$     ▷ Assign reward or set to 0 if history is empty
19:             $\Delta m_j \leftarrow r_j^{\text{new}} - r_j^{\text{prev}}$
20:             $\mathcal{LP}_j \leftarrow (1-\beta)\,\mathcal{LP}_j + \beta\,\Delta m_j$     ▷ Update moving average of the LP $t_i$
21:             $H_j \leftarrow H_j \cup \{r_j^{\text{new}}\}$     ▷ Update the history
22:         for $j \in \{1, \dots, N\}$ do
23:             $P(j) \leftarrow \max(0, \mathcal{LP}_j) + \varepsilon$
24:         $P \leftarrow P/\sum_{k=1}^{N} P(k)$
25: return $\pi_\theta$
```

In summary, it proceeds as follows:

- **Initialization:** We define the set of curriculum tasks T (cf. Eq.1), each associated with a pool of protein sequences S_i [2]. The GFlowNet policy π_θ, is initialized, along with the task-sampling distribution P, which initially assigns equal probability to all tasks.
- **Task Sampling:** At each training step (Alg. 2, lines 3–5), the Teacher samples a task index $k \sim P$. A protein sequence p_{seq} is then drawn from the corresponding pool S_k, and a task-specific environment $\mathcal{E}$ is instantiated for that sequence.
- **GFlowNet Training:** The shared policy π_θ is trained for I_{task} iterations on this environment (Alg. 2,lines 6–11). At each iteration, a weight vector $\boldsymbol{w}$ is sampled, trajectories are generated, and the parameters θ are updated via the SubTB loss. Afterward, the environment is discarded, and a new task is sampled.
- **Evaluation and Adaptation:** At fixed intervals I_{eval} (Alg. 2, lines 12–28), the Student is evaluated on all tasks using a fixed evaluation weight vector $\boldsymbol{w}_{\text{eval}}$. For each task t_j, we compute the mean reward across generated sequences and compare it to the previous

[2]We created the pool of protein sequences sets for each tasks from the CodonTransformer dataset [Fallahpour et al., 2025]

evaluation. The difference defines a learning progress signal Δm_j, which updates $\mathcal{LP}_j$. Tasks with higher $\mathcal{LP}_j$ are assigned greater sampling probability in the following taring step, updating P to favor tasks where the Student shows improvement.

A.8.2 Curriculum Learning Benefits

To isolate the benefits of the CL strategy, we compare four distinct training strategies using the same conditional multi-objective GFlowNets (MOGFN) [Jain et al., 2023] architecture:

- **Short-Only GFN (SGFN)**: A baseline trained exclusively on short protein sequences (30–60 amino acids), testing specialization within its domain.
- **Long-Only GFN (LGFN)**: A baseline trained only on long protein sequences (125–180 amino acids), assessing the difficulty of learning long-range dependencies from scratch.
- **Random-Order GFN (ROFN)**: A control trained on the full dataset but with proteins presented in random order, to check whether gains come from data diversity rather than structured curriculum.
- **Curriculum-Augmented GFN (CAGFN) (Ours)**: The proposed model, trained on the full dataset with tasks organized by increasing sequence length, encouraging hierarchical pattern learning and generalization.

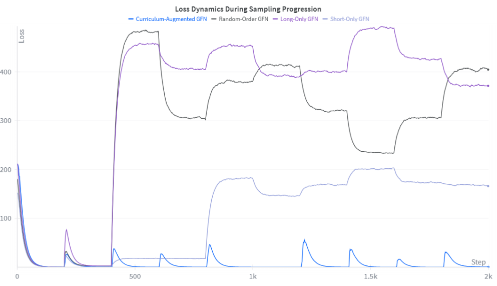
Metrics We evaluated and compared baselines using three complementary metrics: Top-K reward, Top-K diversity, and Pareto performance.

- **Top-K Reward.** The mean reward among the top-K generated candidates:

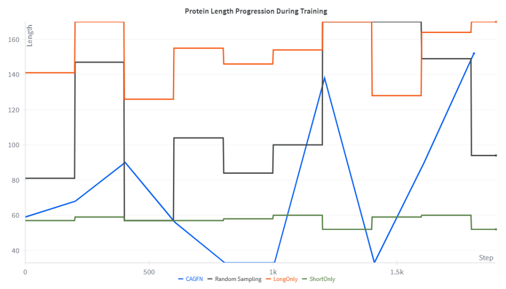
$$\text{TopKReward}_K = \frac{1}{K}\sum_{i=1}^{K} R\big(x_{(i)}\big),$$

  where $x_{(1)}, \ldots, x_{(K)}$ are the K sequences with highest reward.
- **Top-K Diversity.** Measures how diverse the top-K candidates are. We report two compact summaries:
  - *Uniqueness*: fraction of unique sequences in the top-K generated sequences.
  - *Pairwise diversity*: average normalized Levenshtein distance across top-K generated sequences.
- **Pareto performance.** Quantifies the quality of the empirical Pareto front produced by a generative model. We report the fraction of generated candidates that are non-dominated,

$$\text{ParetoPerf} = \frac{|\{x \in S: \ x \text{ is non-dominated in } S\}|}{|S|},$$

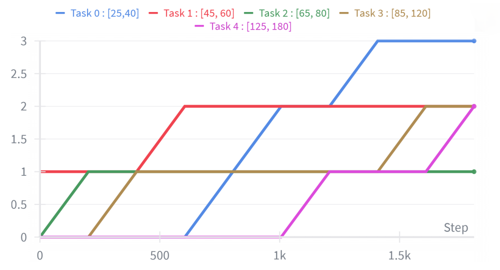


(a) Training loss across the four regimes.

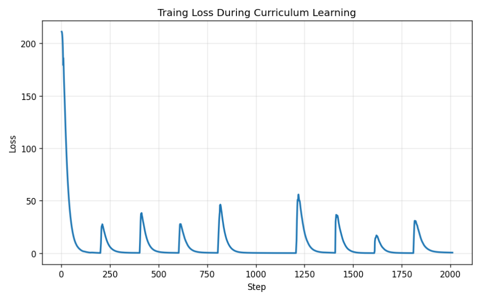


(b) Protein sequence length distribution.

Figure 8: Loss dynamics and curriculum progression. (a) Training loss for *Short-Only*, *Long-Only*, *Random-Order*, and *Curriculum-Augmented GFN (ours)*. (b) Corresponding progression of protein sequence lengths shown to the model during training. The comparison shows how the curriculum adapts the sequence length during training compared to alternative strategies.



(a)



(b)

Figure 9: **Curriculum learning dynamics.** (a) Indicates how often tasks of different lengths are sampled during curriculum, it shows the balance between exploration across easy and hard tasks, ensuring that the training process remains stable while covering the full range of sequence lengths. (b) CAGFN Loss. Brief spikes or oscillations may occur around curriculum tasks transitions.

A.8.3 Curriculum Learning Configurations

To systematically evaluate the impact of different curriculum learning (CL) strategies on the multi-objective mRNA design task, we designed three distinct experimental configurations. These configurations are tailored to probe the trade-offs between learning stability, adaptiveness, and the exploration-exploitation balance. Each configuration uses a unique combination of components for Learning Progress Estimation (LPE), Attention Computation (ACP), and Attention-to-Distribution mapping (A2D), allowing us to isolate and analyze their effects on model performance.

Configuration 1: Conservative EMA-based Curriculum. This configuration is designed to prioritize stable, gradual learning, making it potentially more robust to the high variance often present in biological reward landscapes.

- Learning Progress Estimation (LPE): It employs an Online Exponential Moving Average (EMA) (lpe='Online') with a very small smoothing factor (α=0.05). This ensures that the progress estimate is robust to noisy rewards and changes slowly, preventing drastic shifts in the curriculum.
- Attention Computation (ACP): Task selection relies on a simpler Learning Progress (LP) attention mechanism (acp='LP'), which directly uses the smoothed learning rate to prioritize tasks.
- Distribution Mapping (A2D): It uses an ϵ-greedy proportional strategy (a2d='GreedyProp') with a moderate exploration rate of ϵ=0.15. This forces the model to dedicate 15% of its time to uniform exploration, ensuring it does not prematurely converge on a suboptimal subset of tasks.

Configuration 2: Aggressive Sampling-based Curriculum. In contrast, this setup is engineered for rapid adaptation and aggressive exploration. It is intended to quickly discover and exploit promising areas of the vast mRNA design space.

- Learning Progress Estimation (LPE): It utilizes a Sampling-based progress estimator (lpe='Sampling') over a small, recent window of performance (K=10). This makes the curriculum highly responsive to immediate learning signals.
- Attention Computation (ACP): It leverages a more sophisticated Mastering Rate (MR) attention mechanism (acp='MR') with a high power parameter (power=8) and a strong emphasis on task potential (pot_prop=0.8). This aggressively prioritizes tasks that show the most promise for rapid improvement.

Configuration 3: Balanced Proportional Curriculum. This configuration seeks a robust balance between stability and adaptability, making it hypothetically well-suited for the complex trade-offs inherent in multi-objective optimization.

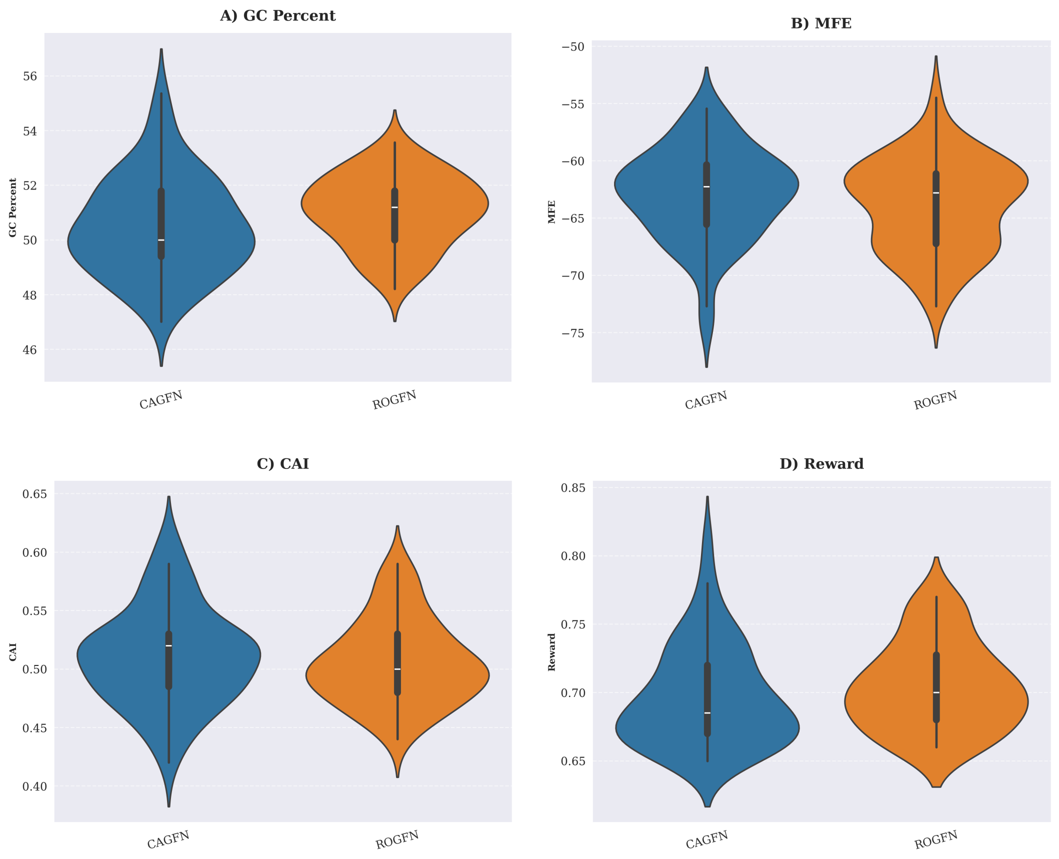


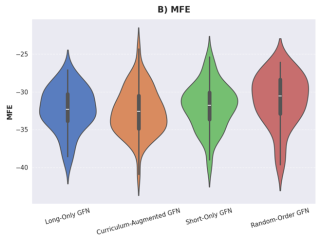
Figure 10: Distribution of reward and objectives for the medium-protein task, contrasting the curriculum-augmented GFlowNet (CAGFN) with a baseline GFlowNet trained without curriculum (ROGFN). Panels show empirical distributions of (a) Reward, (b) CAI, (c) MFE; lower values indicate more stable folding, and (d) GC content. Across all panels, CAGFN-shifted distributions indicate improved sampling: higher rewards and CAI, lower MFE, and GC content closer to the desired range [0.65, 0.35], suggesting that curriculum learning yields higher-quality and more biologically favourable mRNA candidates.

- Learning Progress Estimation (LPE): Learning progress is estimated using Linear Regression (lpe='Linreg') over a larger window (K=25). This provides a stable yet responsive measure of the learning trend, which is effective for handling noisy, multi-objective rewards by capturing the underlying slope of improvement.
- Attention Computation (ACP): It also uses the Mastering Rate (MR) attention mechanism but with moderate parameters (power=4, pot_prop=0.6) to balance focus between currently improving tasks and potentially valuable new ones.
- Distribution Mapping (A2D): Task distribution is purely proportional (a2d='Prop') to the computed attention weights. This represents a pure exploitation strategy based on the curriculum's policy, with no additional random exploration (ϵ=0.0).

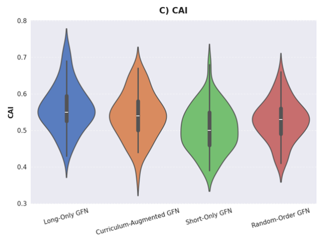
A.9 Hyperparameters and implementation details

Reward Details

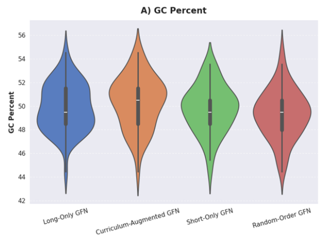
In our implementation, we made use of previously published repositories, making small adaptations where necessary. We used the torchgfn library [Lahlou et al., 2023] to implement the GFlowNet model. To compute reward, we used 3 objectives : the **Codon Adaptation Index (CAI)**, a measure of how well the sequence conforms to species-specific codon usage preferences. Higher CAI



(a) MFE (Low is better)



(b) CAI (High is better)



(c) GC % (High is better)

Figure 11: Metrics distribution for the OAZ2 protein

typically leads to more efficient translation [Sharp and Li, 1987]. To compute the CAI of mRNA sequences, we used the CAI library [Lee, 2018]. The second objective is the **Minimum Free Energy (MFE)**, a (negative) folding free energy of the mRNA's secondary structure. A lower MFE (more negative) indicates a more stable folded structure. Then, third one is the **GC content,** which is the fraction of G or C nucleotides in the sequence. High GC-content generally correlates with increased mRNA stability and translation efficiency [Courel et al., 2019], in particular, optimizing GC-content has a similar effect to optimizing codon usage [Zhang et al., 2023].

Dataset

We base our experiments on the dataset provided by the CodonTransformer paper [Fallahpour et al., 2025], which contains natural mRNA sequences corresponding to a diverse set of proteins. Each entry includes a DNA sequence, the corresponding protein sequence, and gene and organism information. Since our task focuses on mRNA design, we converted the DNA sequences into their corresponding mRNA sequences before using them in our experiments.

Hyperparameters

This appendix lists the hyperparameters and implementation choices used for curriculum learning and the policy function (PF) in our GFlowNet experiments. Values shown are the defaults used in all experiments reported in the paper. We also briefly summarise the alternative ranges explored during tuning and the selection rationale.

Table 3: Curriculum learning hyperparameters

Hyperparameter	Value
`curriculum_tasks`	[25,40], [45,60], [65,80], [85,120], [125,180]
`n_iterations`	100
`eval_every`	5
`train_steps_per_task`	200

Policy function (PF). We choose for the PF module in the GFlowNets a Transformer architecture with the following hyperparameters: $\texttt{embedding\_dim} = 32$, $\texttt{hidden\_dim} = 256$, $\texttt{n\_hidden} = 4$ Transformer layers, $\texttt{n\_head} = 8$.

Values explored during tuning. During preliminary tuning, we explored multiple ranges for each major hyperparameter (final values are reported in the tables). For model capacity we tested $\texttt{embedding\_dim} \in \{16, 32, 64\}$, $\texttt{hidden\_dim} \in \{128, 256, 512\}$, and $\texttt{n\_hidden} \in \{2, 4, 6\}$. For optimization we swept learning rates with $\texttt{lr} \in \{5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}\}$ and $\texttt{lr\_logz} \in \{1 \times 10^{-2}, 1 \times 10^{-1}\}$. For the CL algorithm, we varied teacher parameters such as $\texttt{lpe\_alpha} \in \{0.01, 0.05, 0.1\}$ and $\texttt{acp\_MR\_K} \in \{10, 25, 50\}$.

For each hyperparameter class we ran short preliminary experiments (a small number of curriculum iterations) to evaluate stability, wall-clock cost, and average learning progress. The reported defaults were selected as a trade-off between stable learning dynamics, empirical performance in these exploratory runs, and computational efficiency, when multiple configurations performed similarly we prioritized smaller/cheaper configurations. All experiments in the main text use the values listed in Tables 3–5.

Table 4: Optimization and sampling hyperparameters

Hyperparameter	Value
lr_logz	1e-1
lr	5e-3
lr_patience	10
n_samples	100
top_n	50
batch_size	64
epsilon	0.25

Table 5: Teacher / curriculum selection hyperparameters

Hyperparameter	Value
lpe (learning-progress estimator)	Online
acp (progress metric)	LP
a2d (action-to-difficulty)	GreedyProp
a2d_eps	0.15
lpe_alpha	0.05
acp_MR_K	25
acp_MR_power	2
acp_MR_pot_prop	0.4
acp_MR_att_pred	0.1
acp_MR_att_succ	0.05