Flow Factorization for Efficient Generative Flow Networks

Jiashun Liu[1]*, Chunhui Li[1]*, Cheng-Hao Liu[2], Dianbo Liu[3], Qingpeng Cai[4], Ling Pan[1]†

[1]Hong Kong University of Science and Technology
[2]Mila-Québec AI Institute, McGill University
[3]National University of Singapore
[4]Kuaishou Technology

Abstract

Generative Flow Networks (GFlowNets) is a new family of probabilistic samplers for generating objects under an unnormalized reward distribution. It has emerged as a promising framework for learning stochastic policies that generate high-quality and diverse discrete objects proportional to their rewards, surpassing traditional reward-maximizing reinforcement learning methods. However, existing GFlowNets often suffer with data efficiency due to the direct parameterization of edge flows or dependence on backward policies that are challenging to specify or optimize, especially in high-dimensional action spaces. While the recent development of GFlowNets has primarily focused on developing alternative loss functions, we introduce a novel approach by exploring enhanced flow representations from an architectural perspective. In this paper, we propose to factorize the conventional edge flows into separate state flow and edge-based allocation streams. By introducing an effective method to synergistically combine these two streams to estimate the flows, we develop Bifurcated Generative Flow Networks (**BN**), a practical implementation to improve learning efficiency. We conduct extensive experiments on various standard benchmarks, and results show that BN significantly improves learning efficiency and effectiveness compared to state-of-the-art baselines.

Extended version — https://arxiv.org/abs/2406.01901

1 Introduction

Deep reinforcement learning (RL) has made remarkable progress in recent years, particularly in the domain of games (Mnih et al. 2015; Vinyals et al. 2019). While RL typically aims to maximize the reward function and learn the optimal policy, this singular focus may not always yield the most desirable outcomes in many practical applications where the diversity of generated states is crucial, e.g., molecule discovery (Bengio et al. 2021), combinatorial optimization (Zhang et al. 2023b, 2024), and large language models (Li et al. 2023; Hu et al. 2023). In contrast, Generative Flow Networks (GFlowNets) (Bengio et al. 2021) offer a novel approach as probabilistic samplers that learn a stochastic policy for generating compositional objects $x \in \mathcal{X}$ with probability proportionally to the return $R(x)$. Unlike RL, GFlowNets model all the modes of the reward function, enabling the discovery of high-quality and diverse candidates.

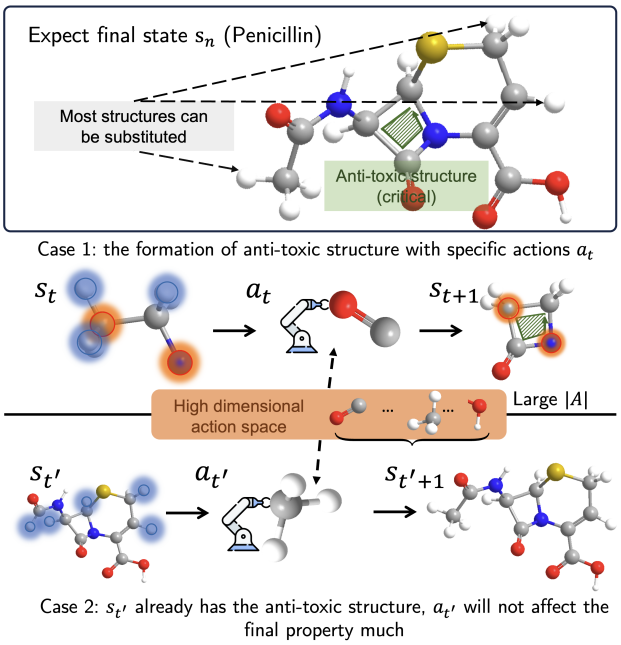


Figure 1: Upon factorizing the conventional edge flows in Flow Matching (Bengio et al. 2021) into the state flow and the edge-based allocation streams, the disparities in attention between these streams across various states are illustrated via the Penicillin discovery task. *The red circle with a red halo* signifies the attention of the edge-based allocation at the current state, while *the blue circle with a blue halo* denotes the focus of the state flow stream.

GFlowNets training can be conceptualized as satisfying the flow consistency constraint in a water flow network (Bengio et al. 2023), ensuring that incoming flows from parent states $F(s_{parent} \to s)$ match the state flow $F(s)$, and equal to outgoing flows to children states $F(s \to s_{child})$, with terminal flows set to returns $R(x)$. Flow Matching (Bengio et al. 2021) pioneered the conversion of this constraint into a practical loss function for training neural networks that directly

*Equal contribution
†Corresponding author, email: lingpan@ust.hk

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

represent edge flows $F(s \to s')$, estimating the state-next state flow function. Recent advancements, including Detailed Balance (Bengio et al. 2023), Trajectory Balance (Malkin et al. 2022), and Sub-Trajectory Balance (Madan et al. 2023), have introduced backward policies $P_B(s|s')$, extending flow matching to various levels based on the TD(λ) principle (Sutton 1988) in the RL literature. However, these methods still face challenges when dealing with high-dimensional tasks involving large-scale action spaces, primarily due to the complexity in specifying and optimizing backward policies.

In high-dimensional and complex real-world tasks such as molecule generation (Bengio et al. 2021), GFlowNets face the challenge of traversing vast state and action spaces. Indeed, the impact of different states and actions on the final properties of generated molecules varies significantly. Figure 1 illustrates this concept through an example of antibiotic generation. At critical states, such as the formation of key functional groups that determine therapeutic properties, certain actions play a crucial role in shaping the outcome. Therefore, an effective flow representation should exhibit high sensitivity to the available actions (as illustrated in Figure 1, reflecting their importance in determining the molecule's final properties. Conversely, during less critical phases, such as the addition of peripheral side chains, action choices may have only marginal effects on the molecule's core functionality. In these cases, the flow representation need not exhibit high sensitivity to specific action choices.

The key insight is that the learning of GFlowNets may not need to model every possible state-action transition with equal precision. While recent GFlowNet research has largely focused on designing alternative loss functions, we propose a complementary approach by exploring better flow representations from the network architectural perspective. In this paper, we propose a novel training perspective for GFlowNets, Bifurcated GFlowNets (BN), that factorizes the edge flows into separate representations for state flows (which estimate the flow through each state) and edge-based allocations (which determines how the flow is distributed among the outgoing edges). Our idea draws inspiration from the dueling network architecture (Wang et al. 2016) in RL, where the Q-value is computed via a state-dependent value function $V(s)$ and an advantage function $A(s, a)$ (i.e., $Q(s, a) = V(s) + A(s, a)$). However, following a direct decomposition as in (Wang et al. 2016) will lead to negative flows that are invalid in the GFlowNets formulation. To address the challenge of extending the advantage function to GFlowNets while maintaining non-negative flow constraints, we redefine the learning objective to incorporate edge-based allocations. This factorization enables BN to learn more efficiently from experiences, which improves efficiency from the collected data, and scales up well to handle large state and action spaces. Finally, extensive experiments on the various standard evaluation benchmarks demonstrate the remarkable effectiveness of BN.

The main contributions are summarized as follows:

- We propose a novel training perspective for GFlowNets, termed Bifurcated GFlowNets (BN), that factories edge flows into separate representations for state flows and edge-based flow allocations.
- We develop a new learning objective that incorporates edge-based allocations, effectively maintaining non-negative flow constraints.
- We conduct extensive experiments on standard evaluation benchmarks in the GFlowNets literature, including HyperGrid (Bengio et al. 2021), RNA sequence generation (Kim et al. 2023), and molecule generation (Bengio et al. 2021), where the results demonstrate that BN significantly improves learning efficiency and task performance compared to previous methods and can be better scale up to real-world high-dimensional tasks.

2 Background

2.1 GFlowNets Preliminaries

Given a directed acyclic graph (DAG) as $G = (\mathcal{S}, \mathcal{A})$, where $\mathcal{S}$ denotes the state space and $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ represents the action set, we denote s_0 in $\mathcal{S}$ as the unique initial state with no incoming edges, and terminal states as those in $\mathcal{X} \subseteq \mathcal{S}$ with no outgoing edges. The objective of GFlowNets is to learn a stochastic policy P_F to objects s_n from trajectories $\tau = (s_0 \to s_1 \to \ldots \to s_n)$ where $s_n \in \mathcal{X}$ and $(s_i \to s_{i+1}) \in \mathcal{A}$, with the sampling probability directly proportional to a reward function $R : \mathcal{X} \to \mathbb{R}_0^+$, such that $P_F^\top(x) \propto R(x)$. As detailed by (Bengio et al. 2021), the flow characterizes the transitions within the DAG. We define the trajectory flow $F : T \to \mathbb{R}^+$. The flow for any state s is given by $F(s) = \sum_{\tau \in T_s} F(\tau)$ where T_s is the collection of all trajectories ends with state s. The flow for any edge $s \to s'$ is given by $F(s \to s') = \sum_{\tau_{s \to s'}} F(\tau)$, which induces a probability measure $P(\tau) = \frac{F(\tau)}{Z}$, Z denotes the total flow, calculated as $Z = \sum_{\tau \in T} F(\tau)$. The forward policy $P_F(s'|s) = \frac{F(s \to s')}{F(s)}$ and the backward policy $P_B(s|s') = \frac{F(s \to s')}{F(s')}$ determine the transitions between states.

2.2 Training Criteria for GFlowNets

Flow Matching (FM). The FM (Bengio et al. 2021) objective realizes the flow consistency constraint (Bengio et al. 2021), i.e., the incoming flows for a state match the outgoing flows, at the state level. FM directly parameterizes the edge flow function as $F_\theta(s \to s')$, where θ represents the learnable parameters. The optimization objective is to minimize the loss function $\mathcal{L}_{\text{FM}}(s)$ defined as

$$\Big(\log \sum_{s'' \in \text{Parent}(s)} F_\theta(s'' \to s) - \log \sum_{s' \in \text{Child(s)}} F_\theta(s \to s')\Big)^2 \tag{1}$$

for non-terminal states, where the log-scale is used for optimization to address stability concerns (Bengio et al. 2021), while the objective for terminal states x is to encourage the incoming flow to match the corresponding reward $R(x)$. To optimize the FM objective, trajectories are sampled from a training policy π that ensures full support over the state space. This policy can be constructed as a tempered version of the forward policy P_{F_θ} or as a mixture of P_{F_θ} and a uniform policy U (i.e., $\pi_\theta = (1 - \epsilon)P_{F_\theta} + \epsilon U$), which can encourage exploration and prevent premature convergence

to sub-optimal solutions. It is proven in (Bengio et al. 2021) that if the expected loss function (Eq. (1)) reaches a global minimum and the training policy π has full support, then FM can generate samples from the desired target distribution.

Detailed Balance (DB). Bengio et al. (2023) propose an alternative objective, DB, to achieve consistent flows in the edge level. The training involves learning three models: a state flow model $F_\theta(s)$, a forward policy model $P_F(\cdot|s)$, and a backward policy model $P_B(\cdot|s)$. The training objective is to minimize the loss defined as $\mathcal{L}_{\text{DB}}(s, s') = (\log(F_\theta(s)P_F(s'|s)) - \log(F_\theta(s')P_B(s|s')))^2$ for the non-terminal states, and a similar object is used to ensure $F_\theta(x)$ match the corresponding rewards $R(x)$ for terminal states.

Trajectory Balance (TB). TB extends the learning objective of DB based on a telescoping calculation to directly train on full trajectories(Malkin et al. 2022), whose learning objective is to minimize the loss function defined as $\mathcal{L}_{\text{TB}}(\tau) = (\log(Z_\theta \prod_{t=0}^{n-1} P_F(s_{t+1}|s_t)) - \log(R(x)\prod_{t=0}^{n-1} P_B(s_t|s_{t+1})))^2$, which parameterizes the forward and backward policies and also the total flow Z_θ.

Sub-Trajectory Balance (SubTB). SubTB considers all possible $O(n^2)$ sub-trajectories $\tau_{i:j} = \{s_i, \cdots, s_j\}$, and the optimization objective is defined as in Eq. (2), where $w_{ij} = \frac{\lambda^{j-i}}{\sum_{0\le i<j\le n}\lambda^{j-i}}$ denotes the weight for $\tau_{i:j}$, and λ represents the weighting hyperparameter.

$$\mathcal{L}_{\text{SubTB}}(\tau) = \sum_{\tau_{i:j}\in\tau} w_{ij}\Bigg(\log F(s_i)\prod_{t=i}^{j-1} P_F(s_{t+1}|s_t) - \log F(s_j)\prod_{t=i}^{j-1} P_B(s_t|s_{t+1})\Bigg)^2 \tag{2}$$

3 Proposed Method

We begin by presenting a motivating example that illustrates the benefits of factorizing edge flows in the flow matching objective (Bengio et al. 2021) into state flows and edge-based allocations. Then, we provide BN for introducing the above idea into a practical training policy, followed by a theoretical analysis and discussion of BN with previous approaches.

3.1 Motivation Example

As discussed above, an expected edge flow should not focus extensively on learning the relative merits of each action. Instead, it should concentrate on states where the choice of action is crucial. However, the direct parametrization of edge flows as in Flow Matching (FM) (Bengio et al. 2021) can not achieve such an effect, therefore decreasing data efficiency, which we illustrate through a tabular example below.

Consider a simple directed acyclic graph (DAG) as shown in Figure 2, which illustrates a hard exploration problem with sparse rewards, as only s_5 has a reward of 1 while other terminal states have very small terminal rewards. Suppose that the FM agent has sampled a trajectory $\tau = \{s_0, s_1, s_4\}$ and obtained a reward of $R(s_4) = 10^{-3}$. According to the learning criterion of FM, the update for state s_4 (illustrated in Figure 2(a)) only directly affects the edge flows $F(s_1 \to s_4)$ and $F(s_2 \to s_4)$ according to $R(s_4)$. However, it does not effectively propagate meaningful information to other transitions that are not sampled, and can leave them underexplored. In contrast, with an effective decomposition of the edge flow into separate representations for state flows and edge-based allocations (to be introduced in Section 3.2), the update for state s_4 will not only affect edge-based allocations, but will also update the value of the corresponding state flows $F(s_1)$ and $F(s_2)$ based on the reward. Thus, it will also implicitly change the value for other unsampled edges including $s_1 \to s_3$ and $s_2 \to s_5$, which informs all outgoing edges from the parent states, thereby enhancing exploration by providing valuable information for potential trajectories like $s_0 \to s_2 \to s_5$. Therefore, it can address FM's data inefficiency problem by ensuring that the training signal is more broadly disseminated throughout the underlying DAG, and realize better exploration and discovery of modes.

It is also worth noting that although there have been recent methods (Bengio et al. 2023; Malkin et al. 2022; Madan et al. 2023) that do not explicitly model edge flows, they rely on a backward policy $P_B(s|s)$ that can be hard to learn or specify in large scale problems with huge action spaces (Zhang et al. 2022). We validate this on a didactic environment with controllable state and action spaces with more complicated transition dynamics than Figure 2. We consider two environment specifications, small and large, with varying state and action space sizes, where the full description of the environment can be found in Appendix B.1. We evaluate the empirical L_1 error (Bengio et al. 2021) (defined as $\mathbf{E}[|p(x) - \pi(x)|]$, where $p(x) = R(x)/Z$ denotes the underlying true reward distribution, and π is approximated by repeated sampling and calculating the visitation frequencies of each x) of the flow factorization method (BN) and compare it with previous GFlowNets objectives including FM, DB, TB, and SubTB. As shown in Figure 3, when the problem size is large, all methods that rely on learning the backward policy (including DB, TB, and SubTB) face challenges in scaling effectively, which highlights the difficulty of specifying or learning the backward policy in large-scale problems (Bengio et al. 2021).

3.2 Bifurcated Generative Flow Networks (BN)

As discussed in the previous sections, it can pose a significant challenge for the agent to accurately capture the state-next state flows for every possible transition and to efficiently learn from experiences. We aim to naturally separate the edge flow function into a state-only dependent state flow and an edge-related component responsible for assigning importance to different actions within a single state. In RL, there is a natural decomposition of the Q-value for a state-action pair $Q(s, a)$ based on the summation of the state-value $V(s)$ and advantage function $A(s, a)$, i.e., $Q(s, a) = V(s) + A(s, a)$. The advantage function-based Q-value decomposition (Wang et al. 2016) can greatly improve sample efficiency by more efficient information propagation.

However, extending the notion of the advantage function to GFlowNets is non-trivial, and a direct extension (i.e., $F(s \to s') = F(s) + A(s \to s')$) is invalid. This is be-

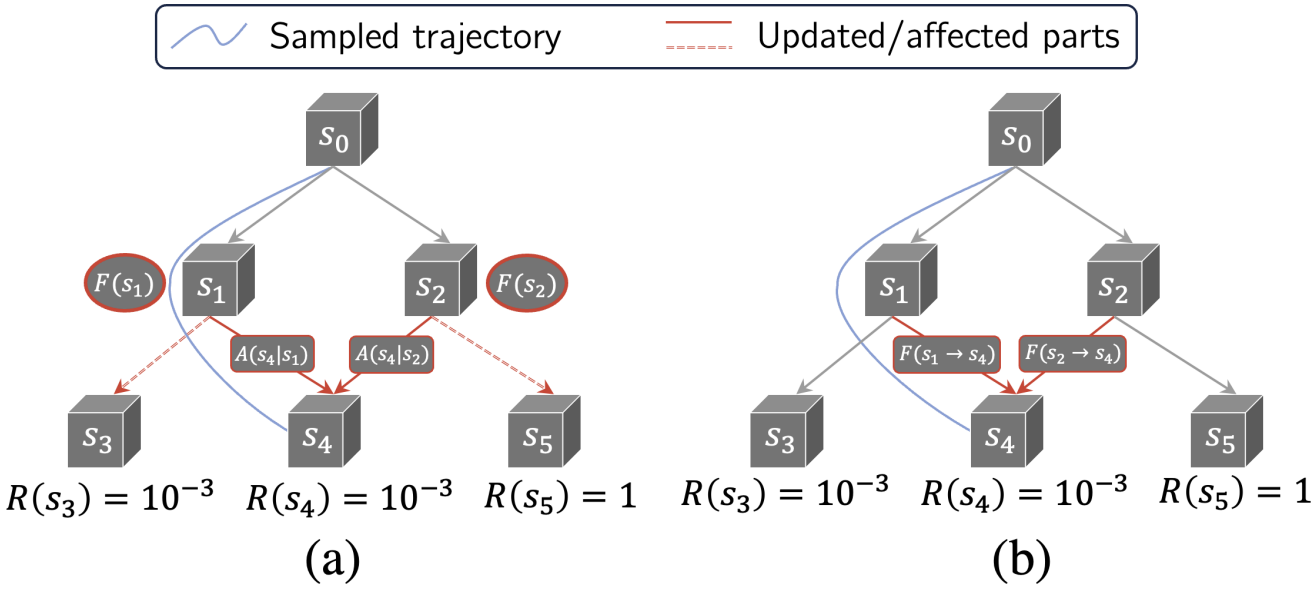


Figure 2: The data inefficiency problem in a simple scenario.

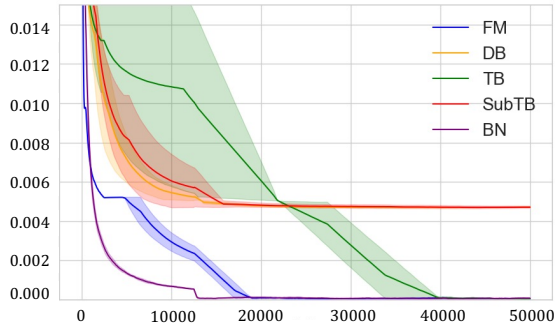


Figure 3: Results in large didactic DAG tasks.

cause the state flow for a given state s' is defined as the summation of incoming edge flows from its parent states s, i.e., $F(s) = \sum_{s'' \in \text{Parent}(s)} F(s'' \to s)$, and the naive extension of the advantage function can lead to *negative* advantage flows, which does not make sense and contradicts the fundamental non-negative flow constraints inherent to flow network systems. To address this challenge, we carefully analyze the flow definition in GFlowNets, and introduce a novel notion of edge-based allocation based on the relationships of value functions and flow functions (Tiapkin et al. 2024), defined as $\log F(s \to s') = \log F(s) + \log A(s'|s)$, where $F(s)$ represents the state flow and $A(s'|s)$ denotes the edge-based allocation (as a probability distribution). This allows for a more interpretable and expressive representation of the flow.

Based on this notion of edge-based allocations, we derive the learning objective of Bifurcated Generative Flow Networks (BN) as in Eq. (3) by extending Flow Matching to allow for efficient decomposition of the edge flow.

$$\sum_{s} F(s)A(s'|s) = F(s') = \sum_{s''} F(s')A(s''|s') \quad (3)$$

Notably, the right-hand-side can be further simplified as $\sum_{s''} A(s''|s') = 1$, and we obtain the final loss function for learning BN as in Eq. (4) for non-terminal states, and a similar objective encouraging the inflow at terminal states x equal the reward $R(x)$. We train BN in the log-scale following (Bengio et al. 2021) to stabilize the learning process.

$$\mathcal{L}_{\text{BN}}(s') = \left(\log \sum_{s \to s' \in \mathcal{A}} F(s)A(s'|s) - \log F(s') \right)^2 \quad (4)$$

Theoretical Justification. In Theorem 1, we provide a theoretical justification for the BN learning objective, demonstrating its convergence and correctness. The detailed proof can be found in Appendix A.

Theorem 1. *If $\mathcal{L}_{BN}(s') = 0$, for $\forall s$, then the edge advantage policy $A(s'|s)$ samples proportionally to the reward function.*

Practical Implementation. The idea can indeed be implemented in a simple yet effective manner, where Figure 4 illustrates the learning structure of our proposed Bifurcated GFlowNets. It consists of a carefully designed bifurcated architecture with a shared state encoder parameterized by θ, which decomposes the edge flow $F(s \to s')$ into a state flow network stream $F(s)$ parameterized by μ and an edge advantage network stream $A(s'|s)$ parameterized by η.

Discussion. Our Bifurcated Generative Flow Networks (BN) introduce a novel decomposition of the edge flow in GFlowNets (Bengio et al. 2021), inspired by the dueling deep

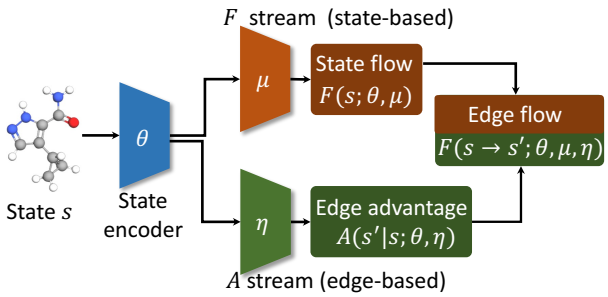


Figure 4: The network structure of BN.

Q-network (DQN) (Wang et al. 2016) architecture. However, it is non-trivial to directly extend such decomposition to GFlowNets, as it can lead to impractical negative edge-based flows. Through an in-depth analysis of the flow function in GFlowNets, we propose an innovative edge flow decomposition that effectively separates the representations into state-related and edge-related components, which significantly enhances data efficiency. Our method is also related to the actor-critic framework (Sutton et al. 1999), featuring a state-dependent component that estimates the flow value for each state and a policy-like component for edge-based allocations. In contrast, FM can be considered as a pure value-based approach, while methods like DB, TB, and SubTB involve learning a state flow function, a forward policy, and an additional backward policy. BN offers a new perspective by circumventing the complexities of the introduction of the backward policy, which can be challenging to specify or learn in high-dimensional problems. Interestingly, our final learning objective also exhibits a close relationship with the concept of Detailed Balance when considering the full spectrum of possible next states (Bengio et al. 2023), which can also be derived from another analytical perspective of DB (Please refer to Appendix D for the detailed analysis).

4 Experiments

In this section, we conduct extensive experiments to understand the effectiveness of our method and investigate the following key questions: i) How does BN's performance compare to state-of-the-art baselines? ii) To what extent does BN improve learning efficiency and promote diversity in solution generation? iii) How effectively does BN scale to tackle larger-scale and more complex tasks?

4.1 HyperGrid

Experimental Setup We first evaluate our method on a commonly used toy HyperGrid task to validate its effectiveness as in (Bengio et al. 2021). The agent navigates in the d-dimensional grid-based world by selecting actions at each timestep. The objective is to model the distribution of the target rewards and successfully capture all the valuable modes.

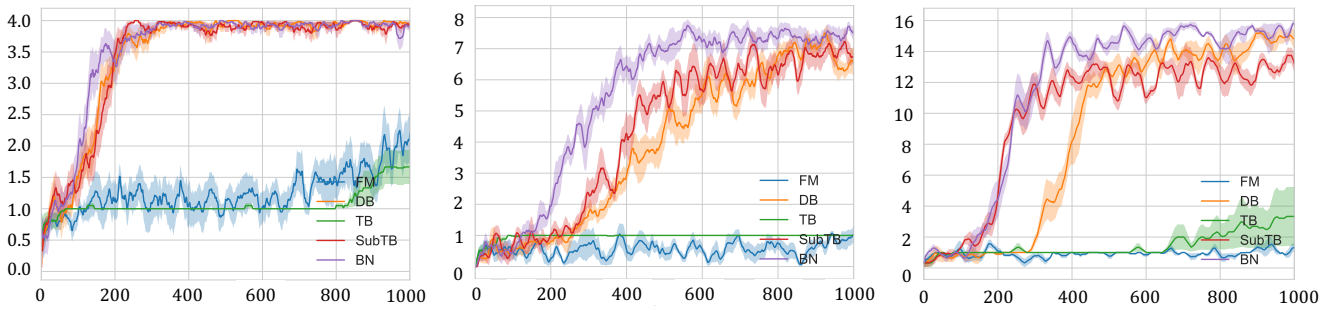
A two-dimensional illustration of the task with horizon H is shown in Figure 6, where the intensity of the color shading represents the magnitude of the rewards, with darker colors indicating higher reward values. The objective is to model the distribution of the target rewards and successfully capture all the valuable modes.

We evaluate the performance of each algorithm using five different seeds (0-4), and the results are presented as the mean performance across these runs, along with the corresponding standard deviation. The implementation of all baseline methods is based on the publicly available open-source code following default hyperparameters as used in (Bengio et al. 2021; Malkin et al. 2022; Madan et al. 2023). Please refer to Appendix B.2 for a detailed description of the reward function $R(x)$ and hyperparameter settings due to space limitation.

Performance Comparison We evaluate our method and compare it against strong baselines including Flow Matching (FM) (Bengio et al. 2021), Detailed Balance (DB) (Bengio et al. 2023), Trajectory Balance (Malkin et al. 2022), and Sub-Trajectory Balance (Madan et al. 2023), following the evaluation scheme in (Bengio et al. 2021) based on the empirical L_1 error and the number of discovered modes. Specifically, the L_1 error is computed as $\mathbb{E}[|p(x) - \pi(x)|]$, where $p(x) = R(x)/Z$ represents the underlying true reward distribution. To estimate π, we repeatedly sample and summarize the visitation frequencies for each possible state x. In this toy environment with horizon $H = 16$ and increasing dimensions $d \in \{2, 3, 4\}$, the state space is relatively small, so the true reward distribution can be directly calculated since it allows for enumerating all possible states. We measure the number of modes discovered during the last 1024 samples. The purpose of evaluating this toy environment is to verify that our method can converge to sampling proportionally to the rewards, and comparisons in more complex and practical environments are analyzed in Sections 4.2 and 4.3.

The comparison results in HyperGrid in terms of the number of updates (Bengio et al. 2021) with increasing dimensions of the action space n (from small, medium to large) are summarized in Figures 5(a)-(c). As shown, our approach significantly improves the learning efficiency of FM, which validates our hypothesis that the well-designed learning structure improves data efficiency. Furthermore, BN outperforms strong baselines including DB and TB by a large margin.

In this toy environment, the most competitive baseline with our approach is SubTB, where our BN method performs comparably to this strong baseline and outperforms it as the problem scale increases with larger action spaces. However, it is important to note that the computational cost of SubTB grows quadratically with the trajectory length n, as it considers all possible $O(n^2)$ sub-trajectories when computing the loss for a single trajectory. In contrast, our method only grows linearly with the trajectory length. We further compare each method in terms of wall-clock time as studied in (Falet et al. 2023). As shown in our arxiv version, BN significantly outperforms SubTB, which is more computationally efficient and leads to faster wall-clock time convergence. In addition, we further demonstrate in Appendix C.1 that our method outperforms RL methods in diverse generation scenarios and analyze the efficiency of encoder in Appendix C.2.



(a) Small. (b) Medium. (c) Large.

Figure 5: Comparison of updates number on HyperGrid. (a)-(c): Number of modes.

Method	DB	TB	SubTB	FM	BN
# of mols	6.67 ± 4.11	4.33 ± 2.05	4.67 ± 2.62	1447.00 ± 267.55	$\mathbf{2345.67 \pm 92.07}$

Table 1: The number of new molecules discovered by each method with a score > 8 that are not in the training dataset.

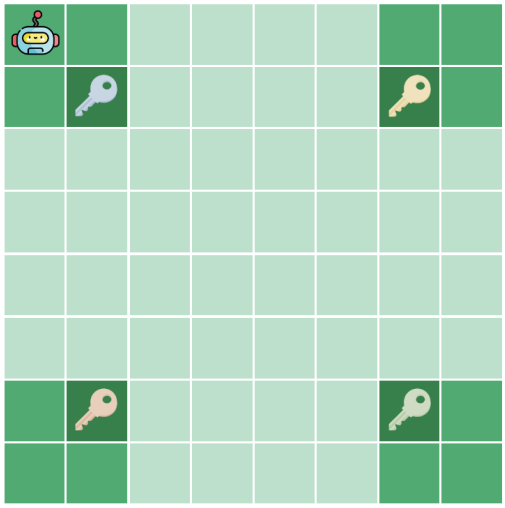


Figure 6: The HyperGrid task.

4.2 Biological Sequence Design

Experimental Setup We now study a practical task of generating RNA sequences consisting of 14 nucleobases which aims to achieve high binding affinity to the target transcription factor. At each timestep, the agent chooses to either prepend or append a token to the current state, following the setup in (Shen et al. 2023) that results in a directed acyclic graph instead of a simple tree. We consider four different target transcriptions introduced by (Lorenz et al. 2011). Details for the experimental setup can be found in Appendix B.3.

Results We follow the evaluation scheme in (Shen et al. 2023; Kim et al. 2023) and evaluate our method and baselines in terms of accuracy, which measures how well it matches the target reward distribution. Specifically, it is computed by using a relative error between the sample mean of $R(x)$ under the learned policy distribution $P_F(x)$ and the expectation of $R(x)$ given the target distribution $R(x)/Z$, i.e., $\frac{\mathbb{E}_{P_F(x)}[R(x)]}{\mathbb{E}_{p*(x)}[R(x)]}$ (and clipped with a maximum value of 1). In addition, we also measure the number of modes discovered by each method during the course of learning following (Shen et al. 2023; Kim et al. 2023), to investigate the diversity-seeking ability.

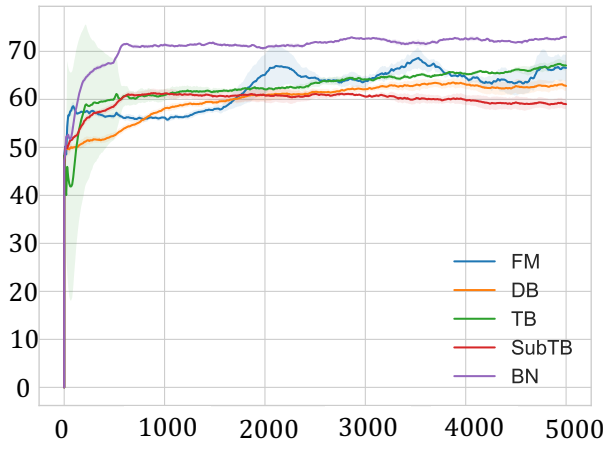


Figure 7: Comparison in terms of accuracy for different methods in RNA generation task.

As demonstrated in Figure 7, our method achieves higher accuracy and learns more efficiently than other baselines in different reward setups. Figure 8 demonstrates the number of modes discovered by each method, which shows that our proposed approach discovers more modes in a more efficiently manner, which validates its effectiveness in practical tasks.

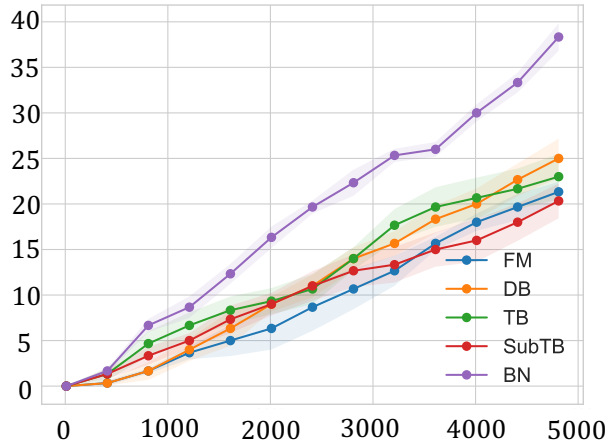


Figure 8: Comparison in terms of the number of modes discovered for different methods in RNA generation task.

4.3 Molecule Generation

Experimental Setup We investigate the efficacy of BN on the more complex and larger-scale molecule generation task (Bengio et al. 2021), In this task, molecules are represented as graphs, composed of a set of building blocks.

The agent iteratively constructs the molecule by deciding the attachment point and the specific block to attach at each step, while adhering to chemical validity constraints. The action space also includes an exit action allowing the agent to terminate the generation process. This task presents significant challenges due to the large state (approximately 10^6) and action (can contain about 2000 actions) spaces. The objective of the agent is to discover a diverse set of molecules with high rewards, corresponding to low binding energy to the soluble epoxide hydrolase (sEH) protein (Bengio et al. 2021). Following (Bengio et al. 2021), we employ a pre-trained proxy model to estimate this binding energy. Please refer to Appendix B.4 for further details of experimental setup.

Results We follow the same evaluation scheme as in (Bengio et al. 2021) by comparing each method in terms of top-K rewards and the number of modes discovered during the training course. As shown in Figure 9, our method achieves the highest top-K rewards, where the most competitive baseline in this large-scale problem is FM (consistent to previous studies (Pan et al. 2022; Zhang et al. 2023a)). Other baselines that involve learning or specifying a backward policy fail to perform well. In addition, our method also discovers many more modes than FM, with a significant gap compared to DB, TB, and SubTB. Table 1 summarizes the number of new unique molecules discovered by each method with a score above 8 that are not contained in the training dataset. As demonstrated, our method has the potential to discover new candidates with high quality that have not been seen before. We further calculate the average pairwise Tanimoto similarity (Bengio et al. 2021) for the top-K molecules produced by BN and a leading FM method. BN yields a result of 0.472 ± 0.015, while FM yields 0.497 ± 0.037. This suggests that the top-performing samples from our method are more varied than those from FM. The results demonstrate the effectiveness of BN in large-scale practical problems.

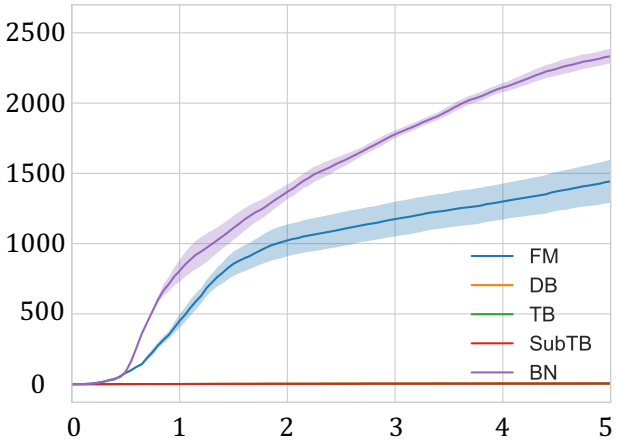


Figure 9: Results in molecule generation. Number of modes with $R > 8.0$. X-axis denotes the states visited number ($\times 10^5$)

5 Conclusion and Future Work

In this paper, we introduce Bifurcated GFlowNets (BN), a novel approach to address the challenges of data efficiency and scaling up to larger-scale problems. BN factorizes edge flows into separate representations for state flows and edge-based allocations, which leads to more efficient learning. Extensive experiments demonstrated that BN outperforms existing GFlowNets variants, achieving superior performance in terms of both learning efficiency and overall effectiveness. While BN involves summation over parent states, the computational overhead is minimal, as shown by wall-clock time comparisons (Please refer to our arXiv version). Our architectural method opens new avenues for GFlowNets research. Future work could focus on enhancing efficiency and scalability in larger state spaces.

Acknowledgments

The work of Jiashun Liu, Chunhui Li, and Ling Pan is supported by the MTR Research Funding 2024 (UST-24007). Dianbo Liu acknowledges the support of NUS-University of Toronto Joint funding.

References

Bengio, E.; Jain, M.; Korablyov, M.; Precup, D.; and Bengio, Y. 2021. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34: 27381–27394.

Bengio, Y.; Lahlou, S.; Deleu, T.; Hu, E. J.; Tiwari, M.; and Bengio, E. 2023. Gflownet foundations. *Journal of Machine Learning Research*, 24(210): 1–55.

Falet, J.-P. R.; Lee, H. B.; Malkin, N.; Sun, C.; Secrieru, D.; Zhang, D.; Lajoie, G.; and Bengio, Y. 2023. Delta-AI: Local objectives for amortized inference in sparse graphical

models. In *The Twelfth International Conference on Learning Representations*.

Hu, E. J.; Jain, M.; Elmoznino, E.; Kaddar, Y.; Lajoie, G.; Bengio, Y.; and Malkin, N. 2023. Amortizing intractable inference in large language models. *arXiv preprint arXiv:2310.04363*.

Kim, M.; Yun, T.; Bengio, E.; Zhang, D.; Bengio, Y.; Ahn, S.; and Park, J. 2023. Local Search GFlowNets. In *The Twelfth International Conference on Learning Representations*.

Li, Y.; Luo, S.; Shao, Y.; and Hao, J. 2023. Gflownets with human feedback. *arXiv preprint arXiv:2305.07036*.

Lorenz, R.; Bernhart, S. H.; Höner zu Siederdissen, C.; Tafer, H.; Flamm, C.; Stadler, P. F.; and Hofacker, I. L. 2011. ViennaRNA Package 2.0. *Algorithms for molecular biology*, 6: 1–14.

Madan, K.; Rector-Brooks, J.; Korablyov, M.; Bengio, E.; Jain, M.; Nica, A. C.; Bosc, T.; Bengio, Y.; and Malkin, N. 2023. Learning GFlowNets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, 23467–23483. PMLR.

Malkin, N.; Jain, M.; Bengio, E.; Sun, C.; and Bengio, Y. 2022. Trajectory balance: Improved credit assignment in GFlowNets. *Advances in Neural Information Processing Systems*, 35: 5955–5967.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Pan, L.; Zhang, D.; Courville, A.; Huang, L.; and Bengio, Y. 2022. Generative Augmented Flow Networks. In *The Eleventh International Conference on Learning Representations*.

Shen, M. W.; Bengio, E.; Hajiramezanali, E.; Loukas, A.; Cho, K.; and Biancalani, T. 2023. Towards understanding and improving gflownet training. In *International Conference on Machine Learning*, 30956–30975. PMLR.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3: 9–44.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Tiapkin, D.; Morozov, N.; Naumov, A.; and Vetrov, D. P. 2024. Generative flow networks as entropy-regularized rl. In *International Conference on Artificial Intelligence and Statistics*, 4213–4221. PMLR.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.

Zhang, D.; Dai, H.; Malkin, N.; Courville, A. C.; Bengio, Y.; and Pan, L. 2024. Let the flows tell: Solving graph combinatorial problems with GFlowNets. *Advances in Neural Information Processing Systems*, 36.

Zhang, D.; Malkin, N.; Liu, Z.; Volokhova, A.; Courville, A.; and Bengio, Y. 2022. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, 26412–26428. PMLR.

Zhang, D.; Pan, L.; Chen, R. T.; Courville, A.; and Bengio, Y. 2023a. Distributional gflownets with quantile flows. *arXiv preprint arXiv:2302.05793*.

Zhang, D. W.; Rainone, C.; Peschl, M.; and Bondesan, R. 2023b. Robust scheduling with GFlowNets. *arXiv preprint arXiv:2302.05446*.