EXPLORING MULTIPLE HIGH-SCORING SUBSPACES IN GENERATIVE FLOW NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Flow Networks (GFlowNets) construct combinatorial objects through sequences of elementary actions, but exploration in large spaces can spend much of a finite sampling budget on low-reward regions. We propose CBFlowNet, an adaptive subspace exploration framework that periodically selects subsets of action types to guide trajectory collection. Its controller learns a set-level reward predictor from the mean terminal rewards collected under selected subsets. Policy-conditioned arm statistics, obtained from unrestricted evaluation, serve as auxiliary features rather than sufficient descriptions of subspace value. A Deep Sets scorer combines these features with action identities to model interactions, while bandit-inspired exploration encourages alternative selections as the GFlowNet evolves. Our analysis establishes trajectory-support and zero-loss target-consistency properties and decomposes the errors affecting current-policy set selection. Experiments on molecular, RNA, and bit-sequence design demonstrate improvements in high-reward candidate discovery. Implementations are available at `https://anonymous.4open.science/r/ICLR-41377`.

1 INTRODUCTION

Generative Flow Networks (GFlowNets) (Bengio et al., 2021; Zhang et al., 2022; Cretu et al., 2024) generate diverse, high-scoring candidates in a range of domains, including combinatorial object generation (Zhang et al., 2023c;b). They combine sequential decisions in an MDP with flow-based probability matching to sample candidates in proportion to their rewards.

Despite this appealing objective, practical GFlowNet training often suffers from inefficient exploration in large combinatorial spaces (Kim et al., 2023; Shen et al., 2023). In many realistic tasks, the vast majority of states yield negligible reward, while high-reward candidates occupy a vanishingly small fraction of the space. As a result, early and mid-stage training frequently resembles near-uniform exploration over actions, causing most sampled trajectories to terminate in low-value regions. Under limited sampling budgets, this leads to slow discovery of high-reward modes and empirical terminal distributions that underrepresent the most valuable solutions, even when the theoretical objective is reward-proportional sampling.

Several approaches attempt to increase the *greediness* of GFlowNets, thus biasing towards high-reward regions. Temperature scaling modifies the target distribution to $R(x)^\beta$ with $\beta \gg 1$, but choosing β introduces instability and risks mode collapse (Malkin et al., 2022a; Lau et al., 2024). Other methods incorporate Q-values (Lau et al., 2024), local search (Kim et al., 2023), or evolutionary strategies (Kim et al., 2024a), which bias sampling toward promising regions.

In structured generative domains, combinations of action types can define regions with very different reward profiles. For example, a subset of molecular building blocks may induce a large space containing many high-scoring candidates. This motivates exploration over *action-induced subspaces* (Fig. 1): periodically select a subset of the shared action vocabulary and favor its actions during trajectory collection. The value of a selection depends on interactions among its actions and on the evolving GFlowNet. We therefore treat subspace selection as an online prediction and exploration problem with policy-dependent feedback.

CBFlowNet uses two complementary feedback channels. Unrestricted evaluation produces policy-conditioned arm statistics that describe reward associations of individual action types. Samples

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

collected while training under a selected subset provide a set-level reward label. A Deep Sets scorer learns from these labels using action identities and arm statistics as features. It does not assume that subspace values are determined by individual conditional means. Bandit-inspired scores and set exploration guide subsequent selections. A uniform action component preserves trajectory support, while the effect of changing subspaces on finite-budget discovery and diversity is evaluated empirically. Appendix D compares this approach with QGFN (Lau et al., 2024).

We evaluate CBFlowNet on molecule design (Bengio et al., 2021), three RNA design tasks (Sinai et al., 2020), and bit-sequence generation (Malkin et al., 2022a). The results show improved high-reward candidate discovery during training. Our main contributions are:

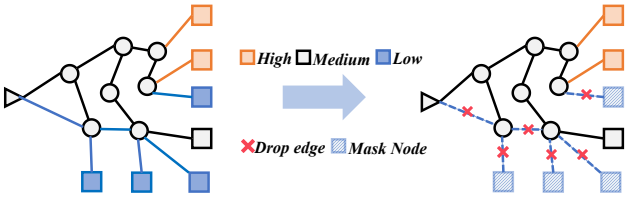


Figure 1: **Illustration of an action-induced subspace.** The triangle is the initial state, circles are intermediate states, and squares are terminals. Retaining selected action types can concentrate sampling on a region containing high-reward candidates (orange). The diagram shows hard gating. Uniform action exploration preserves access outside the selected subspace.

❶ We formulate adaptive GFlowNet exploration as selection over action-induced subspaces whose sampling quality changes with the learned policy.

❷ We develop a controller that learns from set-level rewards using policy-conditioned arm features, and analyze support preservation, zero-loss TB consistency, and current-policy selection errors.

❸ We show that CBFlowNet improves high-reward mode discovery in molecular, RNA, and synthetic sequence design experiments.

2 Related Work

GFlowNets: Since their introduction by Bengio et al. (2021), GFlowNets have advanced rapidly in theory and applications, with recent works establishing connections to variational inference (Malkin et al., 2022b), distributional analysis (Silva et al., 2025), proxy-free training in offline settings (Zhang et al., 2025; Chen et al., 2025), and alternative loss designs (Hu et al., 2024). They have also been applied to combinatorial optimization tasks, including general problems (Zhang et al., 2023b), computation graphs (Zhang et al., 2023a), hierarchical exploration with evolutionary search (Kim et al., 2024b), and multi-objective optimization (Zhu et al., 2023). In contrast to these approaches, which mainly employ GFlowNets to solve combinatorial problems, our method leverages combinatorial optimization techniques to improve GFlowNets themselves. Complementary efforts have enhanced GFlowNet training through Q-value integration (Lau et al., 2024), local search (Kim et al., 2023), Thompson sampling (Rector-Brooks et al., 2023), replay strategies (Vemgal et al., 2023; Shen et al., 2023), genetic and evolutionary algorithms (Kim et al., 2024a; Ikram et al., 2024), and adaptive teacher mechanisms (Kim et al., 2024c).

Combinatorial multi-armed bandit: The combinatorial multi-armed bandit (CMAB) framework was first introduced by (Chen et al., 2013). Chen et al. (2016) later extended it to nonlinear reward functions dependent on variable distributions. Subsequent work includes the cost-aware auction-based CMAB by Gao et al. (2021) and full-bandit algorithms by Agarwal et al. (2021); Fourati et al. (2024b), where no individual arm feedback is available.

3 Preliminary

In the combinatorial multi-armed bandit (CMAB) setting (Chen et al., 2013; 2016), a learner selects a super arm $S_t \subseteq [N]$ with $|S_t| = K$ and observes reward feedback. Full-bandit feedback reveals the aggregate reward, while semi-bandit feedback additionally reveals arm-level outcomes (Chen et al., 2013; 2021; Fourati et al., 2024a). Regret measures performance relative to a specified comparator (Slivkins et al., 2019). We borrow the combinatorial selection perspective and exploration

Table 1: Action decomposition in the three task families.

Task	State-dependent Component (a_d)	State-independent Component (a_i, Base Arm)	$\mathcal{A}_i$
Bit Sequence Generation	Position to edit	Binary value to assign	$\{0, 1\}$
Molecule Design	Stem to extend	Building block to attach	Vocabulary of 105 blocks
RNA Sequence Design	Prepend or Append	Nucleotide to add	$\{A, C, G, U\}$

scores, but our auxiliary arm statistics are conditional averages of terminal rewards, not separately observed reward components whose aggregation defines the set reward.

Our controller receives set-level rewards from subspace-guided sampling and arm features from unrestricted evaluation. Both depend on the evolving policy. This is a CMAB-inspired online subspace selection problem. Standard stationary semi-bandit guarantees do not directly apply.

4 METHODOLOGY

CBFlowNet periodically selects a subset of action types to guide GFlowNet trajectory collection. We first define these subspaces and their set-level feedback, then construct auxiliary arm features and a learned set selector. Gating changes the training behavior policy, while the GFlowNet objective continues to use unrestricted model probabilities.

4.1 FRAMEWORK DESIGN

4.1.1 ACTION VOCABULARY AND INDUCED SUBSPACES

We use the terms *base arm* for a shared action type and *super arm* for a selected subset of types. Treating every state transition $s \to s'$ as a separate arm would create a large, state-specific vocabulary with little sharing. Instead, we exploit the structure of generative actions.

In many sequential generation tasks, actions have two components. The **state-dependent** component a_d specifies where to act, such as which sequence position to fill or which molecular stem to extend. The **state-independent** component a_i specifies the primitive choice, such as the value to assign or the building block to attach.

$$\mathcal{A}_i = \bigcup_{s \in \mathcal{S} \setminus \mathcal{X}} \{a_i : \exists a_d,\ (a_d, a_i) \in \mathcal{A}_s\}. \tag{1}$$

Here $\mathcal{X}$ denotes terminals. The vocabulary labels shared primitive choices, but their validity still depends on $\mathcal{A}_s$. A super arm $\mathbb{S} \subseteq \mathcal{A}_i$ selects vocabulary entries. Hard gating retains valid selected-type actions and stopping actions, falling back to the unrestricted policy for an empty mask. Soft gating reweights actions without masking. Table 1 lists primitive choices. Bit-sequence experiments use macro-actions (Section 4.1.5).

4.1.2 SOFT SUBSPACE GATING

As an alternative to hard gating, we introduce *soft subspace gating*, which favors the selected subspace without masking other actions. Let $P_F(a \mid s; \theta)$ be the unrestricted forward policy and $g_{\mathbb{S}}(a) = \mathbf{1}\{a_i \in \mathbb{S}\}$, with $g_{\mathbb{S}}(a) = 0$ for actions without a base-arm component. We reweight valid actions as

$$\widetilde{P}_F^{\mathbb{S},\eta}(a \mid s) = \frac{P_F(a \mid s; \theta) e^{\eta g_{\mathbb{S}}(a)}}{\sum_{b \in \mathcal{A}_s} P_F(b \mid s; \theta) e^{\eta g_{\mathbb{S}}(b)}}, \qquad \eta \geq 0. \tag{2}$$

This adds η to selected-action logits: $\eta = 0$ recovers the unrestricted policy, while larger values increase the preference for selected arms. With uniform exploration, the training behavior policy becomes

$$\pi_{\text{soft}}(a \mid s, \mathbb{S}) = (1 - \epsilon)\widetilde{P}_F^{\mathbb{S},\eta}(a \mid s) + \frac{\epsilon}{|\mathcal{A}_s|}, \qquad 0 < \epsilon < 1. \tag{3}$$

For finite η, unselected actions retain their learned-policy contribution whenever $P_F(a \mid s; \theta) > 0$, alongside the exploration floor $\epsilon/|\mathcal{A}_s|$. Every terminal reachable through a finite valid trajectory therefore has nonzero sampling probability. Gated training samples supply set-level feedback, while separate unrestricted samples supply arm features.

4.1.3 SET-LEVEL FEEDBACK

At controller interval k, we hold the selected set $\mathbb{S}_k$ for I GFlowNet training steps. Let $x_{k,1}, \ldots, x_{k,M_k}$ be the terminal samples collected under this set's behavior policy, including its exploration component. Their mean terminal reward provides the set-level label

$$Y_k = \frac{1}{M_k} \sum_{j=1}^{M_k} R(x_{k,j}). \tag{4}$$

The scorer is trained to predict this interval sampling yield from the selected set and its selection-time features. GFlowNet parameters continue to change within the interval, so Y_k combines rewards from evolving policies. It need not be an unbiased observation of a single frozen-policy utility and does not measure long-term learning gain. This set-level supervision is distinct from the arm features defined next.

4.1.4 POLICY-CONDITIONED ARM FEATURES

Unrestricted evaluation supplies auxiliary features describing reward associations of each action type under the current model. At evaluation checkpoint t, let q_t be the unrestricted terminal distribution and r_t a reference reward normalization. For arms with positive incidence, define

$$\mu_{i,t} := \mathbb{E}_{x \sim q_t}[r_t(x) \mid i \in x], \tag{5}$$

which changes with q_t and r_t. These statistics describe conditional associations, not intrinsic action contributions. We use them as features alongside action identities. We do not sum them to define set rewards or assume that they uniquely determine subspace values.

For each selected arm with nonempty evaluation feedback, we record

$$X_i^t = \frac{1}{|C_i^t|} \sum_{x \in C_i^t} r_t^{\text{obs}}(x), \tag{6}$$

where C_i^t contains all unrestricted samples with arm i, including repeated draws. Empty batches produce no update. The observed normalization is $r_t^{\text{obs}}(x) = \text{normalize}_{\text{obs}}(R(x))$. Using extrema available before checkpoint t, the reference normalization is

$$r_t(x) = \text{clip}\left(\frac{R(x) - R_{\min,t}}{R_{\max,t} - R_{\min,t} + \varepsilon_{\text{norm}}}, 0, 1 \right), \tag{7}$$

with fixed $\varepsilon_{\text{norm}} > 0$. Freezing normalization within a batch gives $r_t^{\text{obs}} = r_t$, whereas updating extrema from that batch can introduce bias (Lemma B.2). Rewards already in $[0, 1]$ need no additional normalization.

The two sampling phases supply different controller inputs:

❶ **Subspace-guided training and set feedback.** Hard gating mixes the retained-action policy with uniform valid-action exploration of probability $\epsilon > 0$. Soft gating uses Eq. 3. Training samples also supply Y_k for the set scorer.

❷ **Unrestricted evaluation for arm features.** We sample without the selected-set gate and record X_i^t only for selected arms with matching samples. These draws follow the current learned policy, not a uniform distribution over terminals, and are separate from the gated samples used for Y_k.

Unrestricted evaluation requires additional samples and reward calls. Training the set scorer also adds computation. Comparisons at matched GFlowNet training rounds therefore include this extra controller work. Table 8 reports measured time and memory costs.

4.1.5 ADJUSTING THE NETWORK STRUCTURE

When the independent action space is small and trajectories are long, the flow network becomes narrow and deep. We adjust the action granularity to make subspace selection useful in this setting.

For example, in bit-sequence generation, the independent action space contains only two actions, $\{0, 1\}$, whereas generating a complete sequence may require more than 100 steps. This leads to a deep yet fragile network, where pruning even a single action may eliminate the entire solution space. We define each base arm as a short macro-action consisting of t consecutive primitive actions:

$$a_1 \to a_2 \to \cdots \to a_t.$$

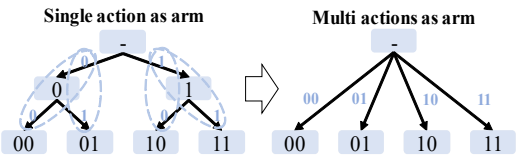


Figure 2: Using short action sequences as base arms reshapes a narrow-and-deep network into a more balanced, wider architecture.

A super arm then consists of a set of such macro-actions. A sub-trajectory is considered valid if its independent subsequence belongs to the selected super arm. This strategy is commonly used in bit-sequence tasks (Zhang et al., 2023c) to reshape the flow network, and we adopt the same idea here in the construction of base arms.

4.1.6 Exploration Features and Learned Set Selection

T_i is arm i's cumulative valid-feedback count, including initialization. Writing $X_i^{(j)}$ for its jth valid feedback record, we use the cumulative mean $\hat{\mu}_i = T_i^{-1} \sum_{j=1}^{T_i} X_i^{(j)}$ ($T_i > 0$). Each new record x increments T_i and updates $\hat{\mu}_i \leftarrow \hat{\mu}_i + (x - \hat{\mu}_i)/T_i$. We augment this feature with a UCB-style exploration score:

$$\overline{\mu}_i = \hat{\mu}_i + \sqrt{\frac{3 \ln t}{2T_i}}, \tag{8}$$

where $t \geq 1$ counts training steps after initialization. The cumulative mean combines feedback from evolving policies, so this score is an exploration feature, not a calibrated bound on current arm or set utility. We also evaluate scoring variants based on CTS (Wang & Chen, 2018) and ESCB (Kveton et al., 2015).

The **Deep Sets scorer** predicts interval yield from action identities and the feature vector $v = (\overline{\mu}_i)_i$:

$$\hat{y}_\psi(\mathbb{S}; v) = \rho_\psi \left(\sum_{a \in \mathbb{S}} \phi_\psi(\text{onehot}(a), v_a) \right). \tag{9}$$

It is trained online on $(\mathbb{S}_k, v_k, Y_k)$, retaining the features v_k used when $\mathbb{S}_k$ was selected. Set-level supervision allows the scorer to learn interactions beyond an additive arm score, though its accuracy on new sets and policies remains an empirical question. To construct a set, we seed it with the highest-scoring arm and greedily add arms maximizing

$$\text{score}(a \mid \mathbb{S}) = \hat{y}_\psi(\mathbb{S} \cup \{a\}; v) + \hat{\sigma}_\psi(\mathbb{S} \cup \{a\}; v), \tag{10}$$

where $\hat{\sigma}_\psi$ is a heuristic Monte Carlo dropout bonus. Neither this bonus nor the greedy search provides a certified set-level confidence bound or approximation ratio. Appendix F details training and partial-set queries.

Algorithm 1 maintains cumulative arm statistics separately from the set scorer's training examples. Empty arm feedback is skipped. The analysis begins after initialization, whose completion is not ensured by positive finite-time arm probabilities alone.

4.1.7 Properties and Scope of the Analysis

Appendix B establishes trajectory support and zero-loss TB target consistency (Proposition B.1), bounds cumulative arm-feature estimation error (Lemma B.2), and decomposes current-policy selection error (Theorem B.3). The selection benchmark freezes the same GFlowNet and reference normalization for every candidate set. Interval labels Y_k reflect evolving policies, so the prediction residual retains temporal and reward-scale mismatches. The analysis explains these error sources without establishing coupled-training convergence or unconditional sublinear regret.

Algorithm 1 Adaptive Subspace Selection with Arm Features and Set Feedback

Maintain: Cumulative mean $\hat{\mu}_i$ and valid-feedback count T_i
Inputs: m arms, set size K, T training steps, update interval I

Define base-arm universe $\mathcal{A} \leftarrow \{1, \ldots, m\}$
// Initialization:
Initialize $T_i \leftarrow 0$, $\hat{\mu}_i \leftarrow 0$ for all i and initialize the Deep Sets scorer
while $\exists i \in \mathcal{A}$ with $T_i = 0$ **do**
    GFLOWNET.TRAIN($\mathcal{A}$), then receive feedback from GFLOWNET.GEN($\mathcal{A}$)
    For each new valid record x of arm i, increment T_i and update $\hat{\mu}_i \leftarrow \hat{\mu}_i + (x - \hat{\mu}_i)/T_i$
end while
// Main Loop:
Set $k \leftarrow 1$, $v_k \leftarrow (\hat{\mu}_i)_{i \in \mathcal{A}}$, and $\mathbb{S}_k \leftarrow$ DEEPSETSGREEDYSELECT(v_k, K)
for $t = 1$ to T **do**
    Train under $\mathbb{S}_k$ and accumulate gated rewards and separate unrestricted arm feedback
    if $t \bmod I = 0$ **then**
        For each valid record x for $i \in \mathbb{S}_k$, increment T_i and update $\hat{\mu}_i \leftarrow \hat{\mu}_i + (x - \hat{\mu}_i)/T_i$
        Compute Y_k from gated rewards (Eq. 4) and store $(\mathbb{S}_k, v_k, Y_k)$
        Update the Deep Sets scorer from stored examples and clear interval accumulators
        Update exploration features: $\overline{\mu}_i \leftarrow \hat{\mu}_i + \sqrt{\frac{3 \ln t}{2T_i}},\ \forall i \in \mathcal{A}$
        $k \leftarrow k + 1$; $v_k \leftarrow (\overline{\mu}_i)_{i \in \mathcal{A}}$; $\mathbb{S}_k \leftarrow$ DEEPSETSGREEDYSELECT(v_k, K)
    end if
end for

5 EXPERIMENT

We evaluate the method on five standard tasks. The baselines are Trajectory Balance (TB) (Malkin et al., 2022a), Sub-Trajectory Balance (SUBTB) (Madan et al., 2023), Detailed Balance (DB) (Jain et al., 2022; Malkin et al., 2022a), LSGFN (Kim et al., 2023), Teacher (Kim et al., 2024c) and QGFN (Lau et al., 2024). All experiments use NVIDIA Tesla A100 80GB GPUs. We examine three research questions (**RQs**): ✎ **RQ1. What is the quality of candidates generated by CBFlowNet?** ✎ **RQ2. How diverse are the generated candidates?** ✎ **RQ3. How do arm scoring and set-selection strategies affect performance?** For RQ3, we report empirical arm-feature trajectories in Bit Sequence Generation and compare bandit-inspired scoring variants and set-selection strategies, including Deep Sets, Top-K, Hard Pruning, and Proportional Pruning. **RandGFN** replaces learned selection with uniformly random subsets. All experiments use 10 seeds.

Unless otherwise stated, subspace-guided sampling uses *hard subspace gating* (hard gating) with uniform action exploration. Appendix G.2 compares this default with soft gating.

5.1 BIT SEQUENCE GENERATION

5.1.1 TASK DEFINITION

The task is to generate binary bit sequences using the set $\{0, 1\}$ with a fixed length $n = 120$ with a terminal state space of $2^{120} \approx 10^{36}$ and more intermediate states. The reward of a terminal x is defined as $R(x) = \exp(-\min_{m \in M} \mathrm{dist}(x, m))$, where $\mathrm{dist}(x, m)$ is the Levenshtein Distance of two sequences following (Malkin et al., 2022a; Zhang et al., 2023c). M is a predefined sequence set to be discovered as modes. The mode $m \in M$ is regarded as found if there exists a sample x satisfying $\mathrm{dist}(x, m) < \delta$, where δ is a predefined parameter.

In this task, we consider a more complex version with many more intermediate states. Malkin et al. (2022a) considers the process as a left-to-right generation where the state space is only a simple tree. Lau et al. (2024); Shen et al. (2023) use a prepend-append MDP to induce a DAG. In our setting, instead of prepending or appending to the existing sequence, we first divide the sequence into $\lfloor \frac{n}{k} \rfloor$ positions. We can insert a generated k-bit into any unfilled position, resulting in a more complex DAG.

5.1.2 RESULT

Figure 3 compares the methods on the bit sequence task. GFlowNets trained with TB outperform those trained with the other objectives, and CBFlowNet finds all target modes. The modes are constructed from combinations of 00000000, 11111111, 11110000, 00001111, 00111100, which helps explain why the base arm $\{0000\}$ performs better than arms such as $\{1011\}$. Figure 3-a shows the empirical feature estimates $\hat{\mu}_{i,t}$ separating during training. The actions $\{0000, 1111\}$ obtain the highest proxy scores, consistent with their frequent occurrence in the construction of M. These curves describe policy-conditioned reward associations rather than intrinsic action values or a convergence guarantee.

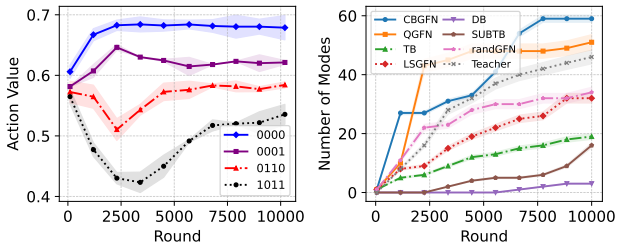


Figure 3: **Bit Sequence results.** Left: empirical policy-conditioned arm features $\hat{\mu}_{i,t}$ used in Eq. 8. Right: modes discovered by each method.

5.2 MOLECULE DESIGN

5.2.1 TASK DEFINITION

Table 2: Comparison on Molecule Design. *Modes R>7.5/8* means the number of modes with a reward bigger than 7.5/8. *perf* denotes the average reward of the top 1000 candidates. *similarity* denotes the tanimoto similarity of the top 1000 candidates. Best result marks exclude results with less than 1000 discovered high-scoring modes.

Model	Modes R>7.5 ↑	Modes R>8 ↑	Top1000	
			Perf↑	Similarity↓
CBFlowNet	**14915.3**±1512.4	**3006.8**±254.2	**8.436**±0.041	0.49±0.01
QGFN	8559.5±438.7	1424.6±96.5	8.387±0.014	0.54±0.01
LSGFN	4528.8±301.4	559.4±47.9	8.321±0.019	0.53±0.01
TB	2525.0±188.6	308.3±29.4	8.216±0.024	0.47±0.02
SUBTB	19.5±6.8	1.7±1.0	7.597±0.061	0.50±0.03
DB	26.3±8.1	3.6±1.4	7.624±0.058	0.46±0.02
Teacher	7897.2±402.5	1311.7±88.2	8.393±0.013	0.49±0.01
RandGFN	2231.8±174.3	286.1±26.7	8.255±0.021	**0.46**±0.02

We consider the most common scenario for GFlowNets, the fragment-based molecule generation task. The objective is to design a variety of molecules with a high reward, where the reward is given by a proxy model predicting the binding affinity to the sEH protein based on a docking prediction(Trott & Olson, 2010). We use the proxy model provided by (Bengio et al., 2021).

In this task, the states are represented as molecule graphs. The action space consists of two components: selecting which molecular stems to extend and choosing which building blocks to add. The maximum number of allowed blocks controls the size of the state space. The vocabulary of building blocks consists of 105 distinct elements, where a block has several possible attachment points(stems). We generate a molecule graph of up to 8 fragments.

We define each block as a base arm and choose K blocks as a super arm. There are C_{105}^{K} different super arms. We use Tanimoto similarity (Bender & Glen, 2004) to distinguish different modes, with a threshold of 0.7. We also examine how the choice of K affects performance and the greediness of sampling.

5.2.2 RESULT

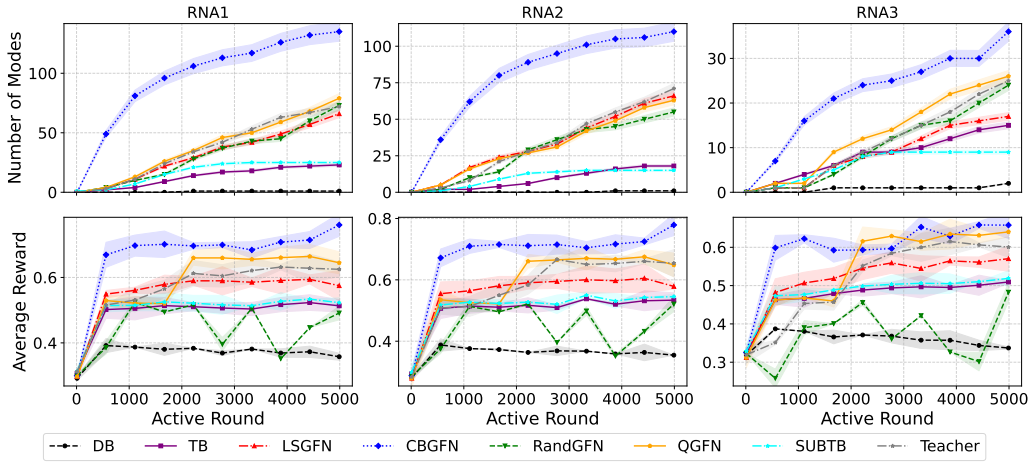


Figure 5: **Performance comparison on RNA design tasks.** Rows correspond to RNA-1, RNA-2, and RNA-3, respectively.

Table 2 and Figure 4 compare the models. SUBTB and DB consistently underperform in both mode discovery and the generation of high-reward candidates. Lau et al. (2024) reported a similar trend in molecule design. Our proposed CBFlowNet improves high-scoring mode discovery, identifying over 10,000 modes with $R > 7.5$ within 100,000 training rounds. It also achieves the highest average reward for the top 1000 candidates among all methods compared.

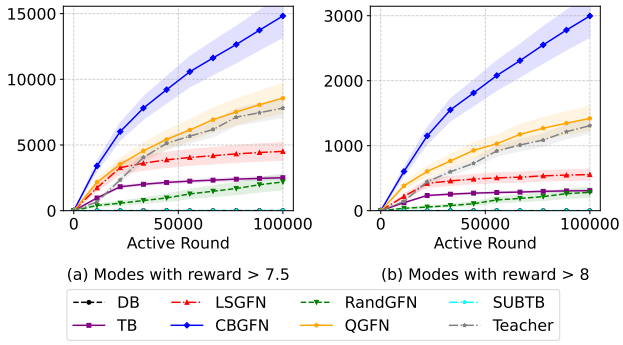


Figure 4: **Modes discovered over training rounds** (10^3) **on Molecule Design.** The left panel counts modes with reward $R > 7.5$, and the right panel counts those with $R > 8$.

Table 2 reports the Tanimoto similarity of the Top-1000 candidates. The learned controller periodically reselects subsets (Alg. 1), allowing collection across different subspaces. Empirically, CBFlowNet combines the best Top-1000 reward with a similarity of 0.49, compared with 0.54 for QGFN. These results indicate that higher reward is compatible with maintaining candidate diversity in this setting.

5.3 RNA-Binding

5.3.1 Task Definition

The task is to generate a string of 14 nucleobases. We use a prepend-append MDP to keep adding tokens to a string until it reaches the maximum length, following Kim et al. (2023). There are 4 tokens: adenine (A), cytosine (C), guanine (G), and uracil (U). We conducted experiments on three different target transcriptions: L14-RNA1, L14-RNA2, and L14-RNA3 proposed by Sinai et al. (2020). We treat each token as a base arm, and K is set to $2/3$, denoting that we can either choose 2 base arms or 3 base arms as a super arm.

5.3.2 Result

Figure 5 reports the number of discovered modes and average reward on the three RNA tasks. CBFlowNet discovers nearly twice as many modes as the strongest competitor and achieves higher average rewards on all three tasks. Section H.1 examines the effect of the reward exponent β (inverse temperature).

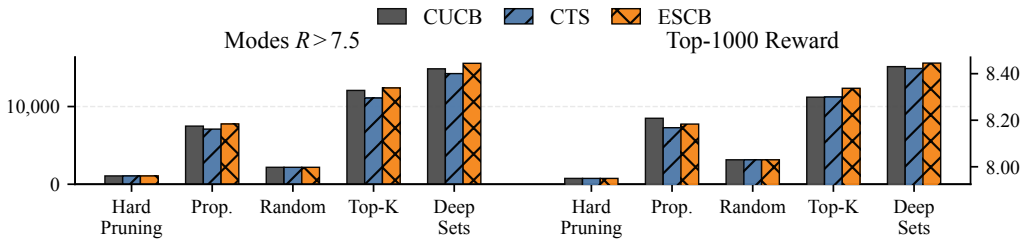


Figure 6: **Molecule ablations.** Three bandit-inspired scoring variants (CUCB, CTS, ESCB) and five set-selection strategies. Left: modes with $R > 7.5$. Right: Top-1000 reward.

5.4 Ablation Study

We examine bandit-inspired arm scoring and set selection on molecule generation. We compare the CUCB, CTS, and ESCB scoring variants under learned Deep Sets selection and four alternatives: sampling arms proportional to their scores, selecting arms uniformly at random, hard pruning, and selecting the top-K highest-scoring arms.

Figure 6 shows that the choice of set-selection strategy has a larger impact than the arm-scoring variant. Deep Sets performs best across all three variants, followed by Top-K. Random selection and hard pruning yield fewer modes and lower reward. These results support learning set-level scores beyond ranking individual proxies. ESCB is broadly comparable to or slightly better than CUCB, whereas CTS tends to underperform, especially in mode discovery.

6 Conclusion and Limitation

Conclusion: CBFlowNet guides GFlowNet exploration through an online set scorer trained on subspace-level rewards, with policy-conditioned arm statistics as auxiliary features. Experiments show improved high-reward discovery, and ablations support learned set selection. The analysis establishes support preservation and zero-loss TB consistency and characterizes cumulative feature-estimation and current-policy selection errors.

Limitations: The method relies on useful action types shared across states and may be less effective when legality constraints substantially change their meaning. Although our experiments use a fixed terminal reward function, the policy and its induced feedback distributions still evolve. Externally changing rewards remain untested. Set prediction and search are approximate, and the analysis does not establish coupled-training convergence, unconditional sublinear regret, or efficient finite-budget mode coverage.

AI use statement

AI-assisted tools were used for language editing and to help identify mathematical issues and draft revisions to theoretical statements and proofs. The authors are responsible for independently verifying the mathematical arguments, implementation details, and experimental claims, and for ensuring that this disclosure accurately reflects the final submission.

References

Mridul Agarwal, Vaneet Aggarwal, Abhishek Kumar Umrawal, and Chris Quinn. Dart: Adaptive accept reject algorithm for non-linear combinatorial bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6557–6565, 2021.

Andreas Bender and Robert C Glen. Molecular similarity: a key technique in molecular informatics. *Organic & biomolecular chemistry*, 2(22):3204–3218, 2004.

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.

Ruishuo Chen, Xun Wang, Rui Hu, Zhuoran Li, and Longbo Huang. Beyond the proxy: Trajectory-distilled guidance for offline gflownet training. *arXiv preprint arXiv:2505.20110*, 2025.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International conference on machine learning*, pp. 151–159. PMLR, 2013.

Wei Chen, Wei Hu, Fu Li, Jian Li, Yu Liu, and Pinyan Lu. Combinatorial multi-armed bandit with general reward functions. *Advances in Neural Information Processing Systems*, 29, 2016.

Wei Chen, Liwei Wang, Haoyu Zhao, and Kai Zheng. Combinatorial semi-bandit in the non-stationary environment. In *Uncertainty in Artificial Intelligence*, pp. 865–875. PMLR, 2021.

Miruna Cretu, Charles Harris, Ilia Igashov, Arne Schneuing, Marwin Segler, Bruno Correia, Julien Roy, Emmanuel Bengio, and Pietro Liò. Synflownet: Design of diverse and novel molecules with synthesis constraints. *arXiv preprint arXiv:2405.01155*, 2024.

Fares Fourati, Mohamed-Slim Alouini, and Vaneet Aggarwal. Federated combinatorial multi-agent multi-armed bandits. *arXiv preprint arXiv:2405.05950*, 2024a.

Fares Fourati, Christopher John Quinn, Mohamed-Slim Alouini, and Vaneet Aggarwal. Combinatorial stochastic-greedy bandit. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12052–12060, 2024b.

Guoju Gao, He Huang, Mingjun Xiao, Jie Wu, Yu-E Sun, and Sheng Zhang. Auction-based combinatorial multi-armed bandit mechanisms with strategic arms. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10. IEEE, 2021.

Rui Hu, Yifan Zhang, Zhuoran Li, and Longbo Huang. Beyond squared error: Exploring loss design for enhanced training of generative flow networks. *arXiv preprint arXiv:2410.02596*, 2024.

Zarif Ikram, Ling Pan, and Dianbo Liu. Evolution guided generative flow networks. *arXiv preprint arXiv:2402.02186*, 2024.

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

Hyeonah Kim, Minsu Kim, Sanghyeok Choi, and Jinkyoo Park. Genetic-guided gflownets for sample efficient molecular optimization. *arXiv preprint arXiv:2402.05961*, 2024a.

Minsu Kim, Taeyoung Yun, Emmanuel Bengio, Dinghuai Zhang, Yoshua Bengio, Sungsoo Ahn, and Jinkyoo Park. Local search gflownets. *arXiv preprint arXiv:2310.02710*, 2023.

Minsu Kim, Sanghyeok Choi, Hyeonah Kim, Jiwoo Son, Jinkyoo Park, and Yoshua Bengio. Ant colony sampling with gflownets for combinatorial optimization. *arXiv preprint arXiv:2403.07041*, 2024b.

Minsu Kim, Sanghyeok Choi, Taeyoung Yun, Emmanuel Bengio, Leo Feng, Jarrid Rector-Brooks, Sungsoo Ahn, Jinkyoo Park, Nikolay Malkin, and Yoshua Bengio. Adaptive teachers for amortized samplers. *arXiv preprint arXiv:2410.01432*, 2024c.

Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Tight regret bounds for stochastic combinatorial semi-bandits. In *Artificial Intelligence and Statistics*, pp. 535–543. PMLR, 2015.

Elaine Lau, Stephen Lu, Ling Pan, Doina Precup, and Emmanuel Bengio. Qgfn: Controllable greediness with action values. *Advances in neural information processing systems*, 37:81645–81676, 2024.

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*, pp. 23467–23483. PMLR, 2023.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35: 5955–5967, 2022a.

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. Gflownets and variational inference. *arXiv preprint arXiv:2210.00580*, 2022b.

Jarrid Rector-Brooks, Kanika Madan, Moksh Jain, Maksym Korablyov, Cheng-Hao Liu, Sarath Chandar, Nikolay Malkin, and Yoshua Bengio. Thompson sampling for improved exploration in gflownets. *arXiv preprint arXiv:2306.17693*, 2023.

Max W Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards understanding and improving gflownet training. In *International conference on machine learning*, pp. 30956–30975. PMLR, 2023.

Tiago Silva, Rodrigo Barreto Alves, Eliezer de Souza da Silva, Amauri H Souza, Vikas Garg, Samuel Kaski, and Diego Mesquita. When do gflownets learn the right distribution? In *The Thirteenth International Conference on Learning Representations*, 2025.

Sam Sinai, Richard Wang, Alexander Whatley, Stewart Slocum, Elina Locane, and Eric D Kelsic. Adalead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint arXiv:2010.02141*, 2020.

Aleksandrs Slivkins et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

Nikhil Vemgal, Elaine Lau, and Doina Precup. An empirical study of the effectiveness of using a replay buffer on mode discovery in gflownets. *arXiv preprint arXiv:2307.07674*, 2023.

Siwei Wang and Wei Chen. Thompson sampling for combinatorial semi-bandits. In *International Conference on Machine Learning*, pp. 5114–5122. PMLR, 2018.

David W Zhang, Corrado Rainone, Markus Peschl, and Roberto Bondesan. Robust scheduling with gflownets. *arXiv preprint arXiv:2302.05446*, 2023a.

Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. Generative flow networks for discrete probabilistic modeling. In *International Conference on Machine Learning*, pp. 26412–26428. PMLR, 2022.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron C Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with gflownets. *Advances in neural information processing systems*, 36:11952–11969, 2023b.

Dinghuai Zhang, Ling Pan, Ricky TQ Chen, Aaron Courville, and Yoshua Bengio. Distributional gflownets with quantile flows. *arXiv preprint arXiv:2302.05793*, 2023c.

Yudong Zhang, Xuan Yu, Xu Wang, Zhaoyang Sun, Chen Zhang, Pengkun Wang, and Yang Wang. Coflownet: Conservative constraints on flows enable high-quality candidate generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

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.

A Broader Impacts

Our work aims to improve sample-efficient exploration in large combinatorial spaces, which may benefit scientific discovery tasks such as molecular design and RNA sequence optimization by helping researchers identify diverse high-quality candidates more efficiently. More broadly, improving coverage of multiple high-reward regions may provide better candidate sets for downstream expert evaluation while reducing trial-and-error costs. At the same time, methods that more effectively discover high-scoring candidates could be misused in sensitive domains, including the design of harmful molecules or other dual-use artifacts. Moreover, if the reward function is biased, incomplete, or poorly aligned with real-world constraints, the method may amplify undesirable preferences even when it is functioning as intended. We therefore view CBFlowNet as a research tool that should be used with domain-specific constraints, expert oversight, and appropriate safety screening before any real-world deployment.

B Theoretical Analysis

The analysis follows the roles of the components in adaptive subspace selection. We first establish support preservation and consistency of the zero-loss TB target under gated trajectory collection. We then bound the estimation error of cumulative auxiliary arm features and decompose the quality of the selected set under the current model. We learn set values from set-level feedback without assuming that the arm features determine those values. The decomposition keeps policy drift, feedback bias, prediction error, and selection error explicit, without assuming a certified combinatorial oracle or convergence of the coupled training process.

B.1 Trajectory support and the GFlowNet target

Consider a finite action-labelled DAG with a unique initial state s_0. Every state is reachable from s_0 and can reach a terminal state in $\mathcal{X}$. A stopping action, if used, is represented by an edge to a terminal copy. Let $d(s) = |\mathcal{A}_s|$ for nonterminal states. Forward and backward policies are normalized over valid outgoing and incoming edges, respectively.

Proposition B.1 (Support preservation and zero-loss TB consistency)**.** *Suppose every trajectory is collected with an ϵ-uniform action mixture, for a fixed $\epsilon > 0$, and the remaining component is a normalized policy on valid actions. For each terminal x, fix a complete path τ_x and set*

$$p_x = \prod_{s \in \tau_x \text{ before termination}} \frac{\epsilon}{d(s)}.$$

Even if the selected gates and model parameters change adaptively between samples,

$$\Pr\{x \text{ is not sampled in the first } N \text{ trajectories}\} \le (1 - p_x)^N \le e^{-Np_x}. \tag{11}$$

Consequently every terminal is eventually sampled almost surely.

Separately, fix strictly positive, normalized model policies P_F, P_B, a positive target reward R_{tar}, and $Z > 0$. If the population trajectory-balance loss

$$\mathbb{E}_{\tau \sim b}\left[\left(\log Z + \log P_F(\tau) - \log R_{\mathrm{tar}}(x_\tau) - \log P_B(\tau \mid x_\tau)\right)^2\right]$$

is zero under any behavior distribution b with full trajectory support, then

$$P_F^{\mathrm{term}}(x) = \frac{R_{\mathrm{tar}}(x)}{\sum_y R_{\mathrm{tar}}(y)}, \qquad Z = \sum_y R_{\mathrm{tar}}(y). \tag{12}$$

Here the target includes any reward exponent used in training.

Proof. Conditional on the entire past, the probability of following τ_x is at least p_x. If E_n denotes missing x through sample n, the tower property gives $\Pr(E_n) \le (1 - p_x)\Pr(E_{n-1})$. Iteration proves Eq. (11), without requiring independent trajectories. Taking $N \to \infty$ and a union over the finite terminal set proves eventual coverage.

For the TB statement, positive behavior probability and nonnegative squared loss imply $ZP_F(\tau) = R_{\mathrm{tar}}(x)P_B(\tau \mid x)$ on every complete trajectory. Backward paths from x reach the unique root with probability one on this finite DAG, so $\sum_{\tau:s_0 \rightsquigarrow x} P_B(\tau \mid x) = 1$. Summing the identity over these paths gives $ZP_F^{\mathrm{term}}(x) = R_{\mathrm{tar}}(x)$. Summing over x proves Eq. (12). This is the usual zero-loss TB argument (Malkin et al., 2022a). □

The coverage result applies to the training behavior that contains the uniform mixture, not automatically to unrestricted evaluation. Hard gates must define a normalized valid-action policy. If their retained mask is empty, an unrestricted fallback suffices. The lower bound p_x can be exponentially small in path length. Thus support does not imply efficient finite-budget coverage. The TB result concerns zero population loss, not optimizer convergence or zero loss on a finite batch. The gated behavior distribution itself need not be reward proportional.

B.2 Cumulative auxiliary features and set selection

Scope and evaluation clock. We compare feasible sets under the *current frozen GFlowNet*. This benchmark measures instantaneous sampling quality and does not compare counterfactual training histories. Fix a deterministic horizon T for the evaluation/selection checkpoints, including initialization. At most one scalar auxiliary observation per arm is recorded per checkpoint, and this clock includes every observation used in the cumulative statistics. Initialization may finish at a stopping time t_0. Decisions occur at $\mathcal{D} = \{t_0 + 1, \ldots, T\}$, with $\mathcal{D} = \varnothing$ if initialization has not finished by T. The probability bounds are simultaneous within this fixed horizon, without conditioning on initialization completion. Let $\mathcal{G}_t$ contain the information before checkpoint t's feedback is revealed, including the current model and the arms eligible for recording. Eligibility refers to the previous training subspace, and the next selected set is chosen after the feedback is processed.

Let q_t be the unrestricted terminal distribution at this model snapshot and let $r_t : \mathcal{X} \to [0, 1]$ be a reference reward normalization fixed given $\mathcal{G}_t$. For $A_i = \{x : i \in x\}$, assume $q_t(A_i) > 0$ and define the auxiliary feature mean and the set-level utility, respectively, as

$$\mu_{i,t} = \mathbb{E}_{q_t}[r_t(X) \mid A_i], \qquad f_t(S) = \mathbb{E}_{\pi_t^S}[r_t(X)].$$

Here π_t^S applies the specified hard or soft gating and exploration mixture to the *same* model snapshot for every feasible S. The feature vector μ_t provides auxiliary information to the learned scorer. We assume neither an additive decomposition nor a uniquely specified mapping from μ_t to f_t.

The scorer is supervised by the observed interval-level set feedback in Eq. 4. Because the GFlowNet changes during each training interval, this feedback aggregates samples from changing policies and is not generally an unbiased observation of the frozen-policy utility $f_t(S_t)$. The prediction residual below includes this temporal mismatch and any difference between the reward scale used for set supervision and the reference normalization r_t. Separately, unrestricted arm observations estimate the auxiliary features, with their preprocessing and aggregation bias represented by $b_{i,s}$ below.

Cumulative feedback statistics. Let $U_{i,s} \in \{0, 1\}$ indicate that arm i actually receives a recorded, nonempty auxiliary observation $X_{i,s} \in [0, 1]$ at checkpoint s. Empty batches are skipped. Define the conditional observation mean and its bias relative to the current feature mean by

$$\eta_{i,s} = \mathbb{E}[X_{i,s} \mid \mathcal{G}_s, U_{i,s} = 1], \qquad b_{i,s} = \eta_{i,s} - \mu_{i,s}.$$

Set $X_{i,s} = 0$ when $U_{i,s} = 0$. When recording has zero conditional probability, $\eta_{i,s}$ and $b_{i,s}$ can be assigned arbitrarily because no corresponding observation is used. The cumulative count and mean, including initialization, and their historical drift and bias terms are

$$T_{i,t} = \sum_{s=1}^{t} U_{i,s}, \qquad \widehat{\mu}_{i,t} = \frac{1}{T_{i,t}} \sum_{s=1}^{t} U_{i,s} X_{i,s},$$

$$D_{i,t} = \frac{1}{T_{i,t}} \sum_{s=1}^{t} U_{i,s} |\mu_{i,s} - \mu_{i,t}|, \qquad B_{i,t} = \frac{1}{T_{i,t}} \sum_{s=1}^{t} U_{i,s} |b_{i,s}|.$$

The means and error terms are defined whenever $T_{i,t} > 0$, as ensured at decision checkpoints by initialization. Every valid feedback record has equal weight in the cumulative mean. Thus $D_{i,t}$ measures the discrepancy between historical and current feature means over all recorded feedback.

Lemma B.2 (Adaptive cumulative-mean estimation)**.** *For any $\delta \in (0,1)$, with probability at least $1-\delta$, simultaneously for all arms i and checkpoints $t \le T$ with $T_{i,t} > 0$,*

$$|\widehat{\mu}_{i,t} - \mu_{i,t}| \le a_{i,t} + D_{i,t} + B_{i,t}, \qquad a_{i,t} := \sqrt{\frac{\log(2mT/\delta)}{2T_{i,t}}}. \tag{13}$$

No independence across arms is required.

Proof. Conditional Hoeffding's lemma, applied only when an observation is recorded, gives for every real λ

$$\mathbb{E}\Big[e^{\lambda U_{i,s}(X_{i,s}-\eta_{i,s})-\lambda^2 U_{i,s}/8} \mid \mathcal{G}_s\Big] \le 1.$$

This follows by conditioning on $U_{i,s}$. When $U_{i,s} = 0$ the exponential is one. Fix an observation count $k \in \{1,\dots,T\}$ and let $W_s = \mathbf{1}\{T_{i,s-1} < k\}$, with $T_{i,0} = 0$. This selector is known before feedback. Consequently

$$M_u = \exp\left\{ \lambda \sum_{s\le u} W_s U_{i,s}(X_{i,s} - \eta_{i,s}) - \frac{\lambda^2}{8} \sum_{s\le u} W_s U_{i,s} \right\}$$

is a nonnegative supermartingale with $\mathbb{E}M_T \le 1$. Any intervening adaptive training is handled by conditioning first on $\mathcal{G}_s$ and then on the preceding history. On the event $T_{i,T} \ge k$, the sums contain exactly the first k recorded observations. Markov's inequality with $\lambda = 4z$ and $\lambda = -4z$ bounds the joint probability that these k observations are recorded by T and their absolute centered sum is at least kz by $2e^{-2kz^2}$. We do not condition on the event that k observations are recorded.

Set $z = \sqrt{\log(2mT/\delta)/(2k)}$ and union-bound over the m arms and T possible observation counts. At each checkpoint with $T_{i,t} > 0$, the cumulative mean uses the first $k = T_{i,t}$ recorded observations. Hence it differs from the average of the corresponding $\eta_{i,s}$ by at most $a_{i,t}$, simultaneously at all such checkpoints. Adding $\eta_{i,s} - \mu_{i,s}$ and $\mu_{i,s} - \mu_{i,t}$ and applying the triangle inequality proves the claim. □

If recording eligibility is fixed before sampling and feedback averages all matching draws in a fixed-size i.i.d. batch under a frozen policy and normalization, retaining duplicate draws, then the nonempty-batch mean is conditionally unbiased and $b_{i,s} = 0$. Updating min–max normalization from the same batch, deduplicating candidates, stopping based on observed rewards, or aggregating changing policies need not preserve this equality. Lemma B.2 retains the resulting $B_{i,t}$ term.

Set-value prediction with explicit residuals. Let $h_t(v, S)$ denote the current deterministic Deep Sets prediction network trained from set-level feedback, with dropout disabled and its learned parameters fixed at decision t. Its input v supplies auxiliary arm features. Actual Monte Carlo dropout affects the selection rule and is accounted for below. Assume that, for all feasible S,

$$|h_t(v, S) - h_t(w, S)| \le L_t \|v - w\|_\infty. \tag{14}$$

The domain must include $[0,1]^m$ and every realized, possibly *unclipped*, UCB-style feature vector. For example, it may be $[0,\infty)^m$. This regularity condition controls the predictor's sensitivity to its inputs. It makes no sufficiency assumption about the features. For the sum-pooled Deep Sets architecture, if the encoder and decoder have Lipschitz constants $L_{\phi,t}$ and $L_{\rho,t}$ with matching embedding norms, one may take $L_t = K L_{\phi,t} L_{\rho,t}$ for sets of size at most K: sum pooling contributes at most K times the largest embedding perturbation. Prediction accuracy and the quality of set search are represented separately by

$$\xi_t := \max_{S \in \mathcal{S}_{\text{feas}}} |f_t(S) - h_t(\mu_t, S)|, \tag{15}$$

$$e_t := \max_{S \in \mathcal{S}_{\text{feas}}} h_t(v_t, S) - h_t(v_t, S_t) \ge 0, \qquad v_{i,t} = \widehat{\mu}_{i,t} + c_{i,t}. \tag{16}$$

Here $c_{i,t} \ge 0$ is the algorithm's actual feature bonus, and S_t is its actual selection. The residual ξ_t measures the mismatch between the learned set predictor and the current-policy utility, including errors from stale supervision, reward scaling, and generalization to unobserved sets. The gap e_t includes effects of greedy construction, partial-set queries, and dropout-based exploration. These definitions impose no approximation guarantee on Deep Sets and no bound on its extrapolation error.

Theorem B.3 (Current-policy selection-error decomposition)**.** *Under the conditions above, let* $S_t^* \in \arg\max_S f_t(S)$ *and define the cumulative current-policy selection gap by* $\mathrm{DReg}_{\mathcal{D}} = \sum_{t\in\mathcal{D}}[f_t(S_t^*) - f_t(S_t)]$*. With probability at least* $1-\delta$*,*

$$\mathrm{DReg}_{\mathcal{D}} \le \sum_{t\in\mathcal{D}} \left[2\xi_t + e_t + 2L_t \max_i\{a_{i,t} + D_{i,t} + B_{i,t} + c_{i,t}\}\right]. \tag{17}$$

Proof. For each realized decision, insert the predictor at μ_t and v_t for the optimal and chosen sets. Equations (14)–(16) yield

$$\begin{aligned} f_t(S_t^*) - f_t(S_t) &\le 2\xi_t + h_t(\mu_t, S_t^*) - h_t(\mu_t, S_t) \\ &\le 2\xi_t + 2L_t\|v_t - \mu_t\|_\infty + h_t(v_t, S_t^*) - h_t(v_t, S_t) \\ &\le 2\xi_t + e_t + 2L_t\|v_t - \mu_t\|_\infty. \end{aligned}$$

On the simultaneous event of Lemma B.2, $\|v_t - \mu_t\|_\infty \le \max_i(a_{i,t} + D_{i,t} + B_{i,t} + c_{i,t})$. Summation proves the result. The argument is pathwise on that event, so adaptive predictor training and randomized set selection require no additional independence assumption. □

Theorem B.3 separates feature estimation, drift, and feedback bias from set prediction and set search. The notation $\mathrm{DReg}_{\mathcal{D}}$ expresses the cumulative gap to the best set at each frozen model snapshot. The theorem does not bound the learning dynamics of the predictor or establish that its residuals decrease.

As an arm accumulates valid observations, its statistical radius decreases, but its cumulative mean still includes feedback from earlier policies and normalization scales. The historical drift term $D_{i,t}$ includes initialization and need not decrease with the count. No decay of $D_{i,t}$, $B_{i,t}$, ξ_t, or e_t has been established. The decomposition therefore does not assert unconditional sublinear regret or convergence of the coupled training dynamics.

Approximation-oracle benchmarks. A multiplicative oracle guarantee cannot by itself replace the additive gap above in ordinary regret. Even with exact stationary values, an (α, β) oracle can leave an expected per-round gap of $(1 - \alpha\beta)f(S^*)$, producing linear ordinary regret. For example, a $(1/2, 1)$ oracle can always return value $1/2$ when the optimum is 1. Standard CMAB approximation-regret analyses instead adjust the comparison benchmark (Chen et al., 2013; 2021). No such multiplicative guarantee is assumed here for the learned selector.

B.3 Supporting result: stability of policy-conditioned features

The following result describes how the auxiliary feature means behave if the evaluation policy and reward normalization converge. It supplies a conditional stability property of the statistics, separate from the set-selection analysis.

Theorem B.4 (Conditional reward stabilization)**.** *Let* $\mathcal{X}$ *be finite and let the unrestricted evaluation distributions satisfy* $d_t = \|q_t - q_*\|_{\mathrm{TV}} \to 0$*, where* $\|p - q\|_{\mathrm{TV}} = \frac{1}{2}\sum_x |p(x) - q(x)|$*. For each arm under consideration, assume* $a_{i,*} = q_*(A_i) > 0$*, with* $A_i = \{x : i \in x\}$*. For sufficiently large* t*, define* $Q_{i,t} = q_t(\cdot \mid A_i)$ *and* $Q_{i,*} = q_*(\cdot \mid A_i)$*. Then*

$$\|Q_{i,t} - Q_{i,*}\|_{\mathrm{TV}} \le \frac{d_t}{\max\{q_t(A_i), q_*(A_i)\}} \le \frac{d_t}{a_{i,*}}. \tag{18}$$

For a fixed normalized reward map $r : \mathcal{X} \to [0,1]$*, the conditional reward laws converge in total variation, and their means converge with the same bound. If instead* $r_t : \mathcal{X} \to [0,1]$ *converges pointwise to* r_**, let* $\nu_{i,t}$ *be the law of* $r_t(X)$ *for* $X \sim Q_{i,t}$*, and define* $\nu_{i,*}$ *analogously. Then*

$$|\mu_{i,t} - \mu_{i,*}| \le W_1(\nu_{i,t}, \nu_{i,*}) \le \|r_t - r_*\|_\infty + \frac{d_t}{a_{i,*}} \longrightarrow 0. \tag{19}$$

Here W_1 *is the Wasserstein distance with cost* $|u - v|$ *on* $[0,1]$*. The latter statement gives weak convergence of reward laws, but need not give total-variation convergence.*

Proof. Write $A = A_i$, $a = q_t(A)$ and $b = q_*(A) > 0$. Since $|a - b| \le d_t$, eventually $a > 0$. For such t,

$$\sum_{x \in A} \left| \frac{q_t(x)}{a} - \frac{q_*(x)}{b} \right| \le \frac{\sum_{x \in A} |q_t(x) - q_*(x)| + |a - b|}{a} \le \frac{2d_t}{a}.$$

The final inequality uses $|a - b| \le \sum_{x \notin A} |q_t(x) - q_*(x)|$. Interchanging the distributions also gives the bound $2d_t/b$. Dividing by two proves Eq. (18). Applying a fixed reward map contracts total variation, and expectation differences of functions in $[0, 1]$ are bounded by total variation.

For varying normalization, pointwise convergence is uniform on finite $\mathcal{X}$. Couple $Q_{i,t}$ and $Q_{i,*}$ with mismatch probability equal to their total-variation distance. The rewards have unit range, so applying r_t costs at most this mismatch probability. Replacing r_t with r_* under $Q_{i,*}$ costs at most $\|r_t - r_*\|_\infty$. The triangle inequality proves Eq. (19), and the mean bound follows from the same coupling. On $[0, 1]$, convergence in W_1 implies weak convergence. To see why TV need not converge, take a single terminal x and $r_t(x) = 1/t$. Then $\nu_{i,t} = \delta_{1/t}$ and $\nu_{i,*} = \delta_0$ have TV distance one for every finite t. □

Policy convergence and positive limiting arm incidence are assumptions. They are not implied by the training behavior's exploration floor. The result concerns the conditional terminal-reward distribution underlying an auxiliary feature, not the finite-sample law of the batch mean X_i^t. It neither makes these features sufficient for set-value prediction nor establishes convergence of the coupled training algorithm.

C Empirical Reward-Gap Diagnostic

C.1 Scope of the Empirical Diagnostic

Theorem B.3 decomposes the quality of a selected set under the current frozen policy. The empirical diagnostic below summarizes observed set feedback. It does not estimate the theorem's error terms or establish a sublinear dynamic-regret guarantee.

We report a cumulative empirical reward-gap diagnostic. Its comparator is the best empirical set observed up to time t, denoted by $\hat{\mu}_t(\mathbb{S}^*)$, rather than the true optimum over all feasible sets:

$$\hat{R}(T) = \sum_{t=1}^{T} \left(\hat{\mu}_t(\mathbb{S}^*) - \hat{\mu}_t(\mathbb{S}) \right). \tag{20}$$

This diagnostic summarizes gaps within the observed feedback stream. The visited sets, estimates, and learned policies can differ between methods, so the curves do not share the theoretical comparator in Theorem B.3. We include RandGFN, which selects K arms uniformly, as an empirical reference.

Figure 7 compares this empirical diagnostic for CBFlowNet and RandGFN. The gap between their curves is smaller on the bit-sequence task and larger on molecule design and L14-RNA1. Since the diagnostic depends on each method's observed sets and estimates, these curves alone do not identify differences in true dynamic regret. The direct reward and mode-discovery metrics provide the performance comparisons.

D Differences with QGFN

QGFN (Lau et al., 2024) combines GFlowNet probabilities with state-dependent action values. Its p-greedy variant mixes with the original policy, whereas its p-quantile and p-of-max variants mask actions using thresholds that may change during training. Our controller learns set-level scores from interval rewards and auxiliary arm features, selects a shared subset of action types, and holds it over a training interval. Both approaches can collect trajectories off-policy. Our uniform action component preserves training-time trajectory support (Proposition B.1). Reselecting subsets may improve coverage in practice, but neither support preservation nor the selection-error decomposition guarantees finite-budget diversity.

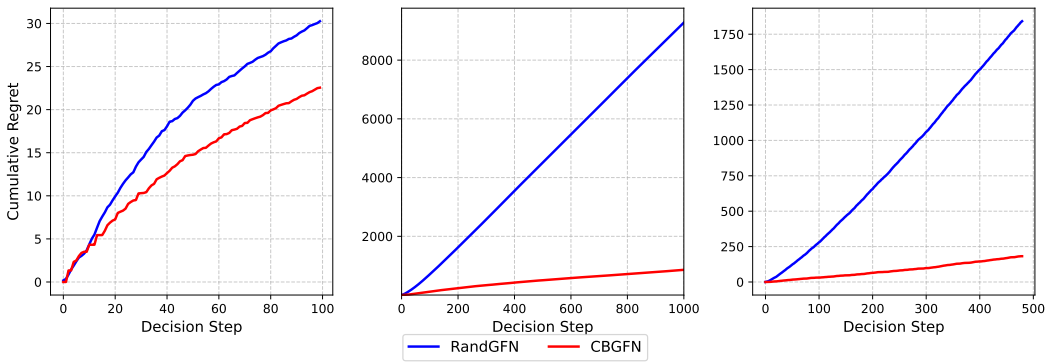


Figure 7: **Cumulative empirical reward-gap diagnostics.** Left: bit sequences. Center: molecule design. Right: L14-RNA1. These observed-feedback diagnostics do not estimate the current-policy selection gap in Theorem B.3.

Table 3: Key hyperparameter setting in Bit Sequence Generation task

Parameter	**Value**
Batch size	16
Number of steps	10000
k-bits	4
Lamda	1.9
Learning rate	1e-3
Z Learning rate	1e-3
β	2
Explore Epsilon	0.01
K	4
Decision Interval	100

E EXPERIMENT DETAILS: BIT SEQUENCE

Table 3 gives the bit-sequence configuration. Adopting the framework of Malkin et al. (2022a), we reduce training from 50,000 to 10,000 steps. Each action represents $k = 4$ bits. We selected $K = 4$ from $\{2, 4, 6, 8, 10\}$ and a controller interval of 100 steps from $\{50, 100, 200, 300, 400, 500\}$ through empirical validation.

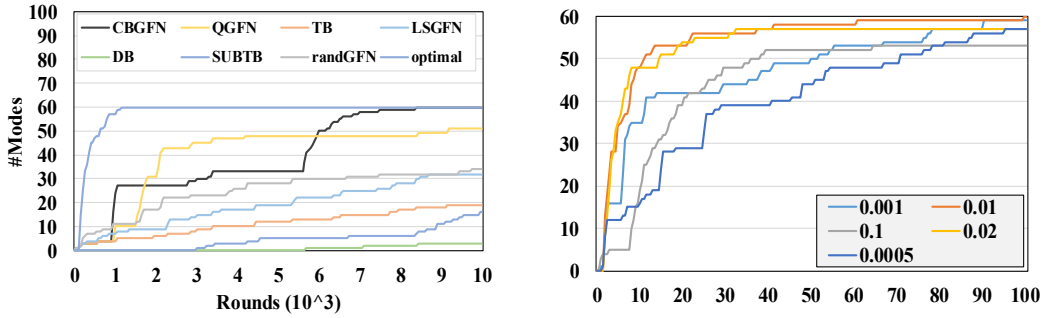
E.1 ILLUSTRATIVE BEST-SET ANALYSIS

The set M is constructed through random combinations of substrings derived from the base patterns $\{00000000, 11111111, 11110000, 00001111, 00111100\}$. For the case where $K = 4$, exhaustive evaluation in this synthetic setting identifies a best-performing super arm $\mathbb{S} = \{0000, 1111, 1100, 0011\}$ under the task reward. This set enables perfect mode identification within its substate space, achieving an average distance of 0.0 to all target modes. We evaluate the behavior when repeatedly selecting this fixed best-performing set in Figure 8a.

E.2 EXPERIMENTS ON DIFFERENT ϵ

We evaluate the exploration parameter ϵ, which controls the probability of random actions in the GFlowNets framework and affects sampling within substate spaces. Figure 8b compares ϵ values $\{0.1, 0.01, 0.02, 0.001, 0.0005\}$. Among these settings, $\epsilon = 0.01$ gives the best results for mode discovery and sample quality.

These results suggest that $\epsilon = 0.01$ balances exploration and exploitation in this task. Higher values introduce excessive randomness, while lower values provide too little exploration of the state space.



(a) Empirical arm-level reward proxies

(b) Number of modes discovered

Figure 8: Supplementary Experimental Results for the Bit Sequence Task.

F Implementation Details of the Deep Sets Selector

The set scorer learns from interval labels Y_k (Eq. 4). Its inputs combine action identity with an auxiliary exploration feature $v_a = \overline{\mu}_a$:

$$z_a(v) = [\text{onehot}(a); v_a]. \tag{21}$$

The set encoder applies a shared MLP ϕ_ψ to each arm feature, sums the resulting embeddings, and feeds the pooled representation to another MLP ρ_ψ:

$$\hat{y}_\psi(\mathbb{S}; v) = \rho_\psi\left(\sum_{a\in\mathbb{S}} \phi_\psi(z_a(v))\right). \tag{22}$$

In our implementation, both ϕ_ψ and ρ_ψ are two-layer MLPs with hidden size 128, and dropout is inserted between hidden layers to support uncertainty estimation.

After each controller interval k, we store $(\mathbb{S}_k, v_k, Y_k)$: the selected set, its selection-time feature vector, and the mean terminal reward collected under its gated behavior policy. Stored examples retain their original features rather than substituting current arm estimates. The scorer is optimized with Adam using learning rate 10^{-3}, minibatch size 32, and 10 gradient steps per controller update, with mean-squared error loss:

$$\mathcal{L}_{\text{DS}}(\psi) = \frac{1}{|\mathcal{B}|} \sum_{(\mathbb{S},v,y)\in\mathcal{B}} \left(\hat{y}_\psi(\mathbb{S}; v) - y\right)^2. \tag{23}$$

At selection time, we use the current feature vector v, seed the set with the highest-scoring arm, and greedily add the remaining $K-1$ arms. The scorer evaluates each candidate addition with a Monte Carlo dropout exploration bonus:

$$\text{score}(a \mid \mathbb{S}) = \hat{y}_\psi(\mathbb{S}\cup\{a\}; v) + \hat{\sigma}_\psi(\mathbb{S}\cup\{a\}; v), \tag{24}$$

where $\hat{\sigma}_\psi$ is the standard deviation across 8 stochastic forward passes. This is an exploration heuristic rather than a calibrated confidence bound. The scorer is trained on complete sets, but greedy construction also queries partial sets. Generalization across these sizes is not guaranteed. The interval label Y_k also reflects changing policies, while the theoretical benchmark fixes a single policy and reference normalization. The prediction and selection residuals in Theorem B.3 retain these mismatches and search errors. Direct set supervision does not establish that the features fully describe the current policy or that prediction errors vanish.

G Experiment Details: Molecule Design

Table 4 gives the Molecule Design configuration, based on the framework of Bengio et al. (2021). Through empirical evaluation, we selected a set size $K = 30$ from $\{8, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ and a controller interval of 400 steps from $\{50, 100, 200, 300, 400, 500\}$.

Table 4: Key hyperparameter setting in Molecule Design

Parameter	**Value**
Batch size	4
Number of steps	100000
Lamda	0.99
Learning rate	5e-4
Z Learning rate	5e-3
Tanimoto Similarity Threshold	0.7
β	8
Explore Epsilon	0.05
K	30
Decision Interval	400

G.1 Matched Total Reward-Evaluation Budget

Matching training rounds alone does not account for the additional unrestricted evaluations used to update the CMAB arm statistics. These samples are not used for GFlowNet gradient updates, but still consume reward evaluations. We therefore additionally compare TB, QGFN, and CBFlowNet on Molecule Design under the same total budget of 500k reward evaluations, counting both training and bandit-evaluation samples.

Table 5: Molecule Design results under a matched total reward-evaluation budget. Total evaluations include both training and bandit evaluations. Top-1000 denotes the average reward of the best 1000 candidates.

Method	Total Reward Evals	Training Evals	Bandit Evals	Modes $R > 7.5 \uparrow$	Top-1000 $\uparrow$
TB	500k	500k	0	3117	8.287
QGFN	500k	500k	0	10567	8.409
CBFlowNet	500k	400k	100k	**14854**	**8.438**

As shown in Table 5, CBFlowNet discovers more high-reward modes and achieves a higher Top-1000 reward than both baselines under this stricter budget matching, despite using fewer samples for gradient updates (400k versus 500k).

G.2 Hard versus Soft Subspace Gating

We compare hard and soft gating on Molecule Design to examine how the strength of subspace restriction affects high-reward discovery. Hard gating renormalizes the learned forward policy over valid selected-type actions and stopping actions, with an unrestricted fallback if the retained mask is empty. Soft gating instead applies the logit bias in Eq. 2: finite $\eta > 0$ favors selected actions while preserving the learned-policy contribution of unselected actions. Both variants use the same uniform exploration probability ϵ, so hard gating still permits exploration outside the selected subspace. Here, "hard" refers to the action mask applied to the currently selected set. The controller continues to reselect sets, unlike the fixed Hard Pruning baseline.

We keep the Deep Sets selector, set size $K = 30$, controller interval, training objective, and all other hyperparameters fixed to the Molecule Design configuration in Table 4. Each variant is trained for $100{,}000$ steps with the same training and unrestricted-evaluation sample budgets, using the same five random seeds. We report the number of modes at reward thresholds 7.5 and 8, the mean reward of the Top-1000 candidates, and their Tanimoto similarity, following Table 2.

Table 6 compares the two gating schemes. Hard gating achieves the best overall performance among the compared variants, supporting its use as our default. Soft gating offers a tunable relaxation of this restriction, but does not improve overall performance in this comparison. However, the Top-1000 similarity achieved by soft gating is slightly better.

Table 6: Hard versus soft gating on Molecule Design. Higher mode counts and Top-1000 reward are better, while lower similarity indicates greater diversity. The symbols η_1, η_2, η_3 denote the soft-gating strengths to be specified.

Gating	η	Modes $R > 7.5$ ↑	Modes $R > 8$ ↑	Top-1000 reward ↑	Top-1000 similarity ↓
Hard (default)	n/a	14833	3036	8.441	0.49
Soft	1	12838	2527	8.395	0.46
Soft	5	13921	2711	8.423	0.48
Soft	10	14479	2876	8.438	0.48

Table 7: Key hyperparameter setting in RNA-Binding task

Parameter	**Value**
Batch size	32
Number of steps	5000
RNA length	14
MDP style	Prepend and Append
Lamda	0.9
Learning rate	1e-4
Z Learning rate	1e-2
Mode metric	Hamming Ball 1
β	8
Explore Epsilon	0.01
K	2/3
Decision Interval	50

H EXPERIMENT DETAILS: RNA-BINDING

In this section, we give the hyperparameters used for each of our experiments' RNA-Binding Task as shown in Table 7. In our experimental setup, the learning rate of 1×10^{-4} is selected from $\{1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}\}$ and the Z learning rate of 1×10^{-2} is selected from $\{1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}\}$. The Lambda for SUBTB uses 0.9 out of $\{0.8, 0.9, 0.99, 0.999\}$. The explore epsilon is used to control the random action probability, five values are tested, including $\{0.1, 0.01, 0.001, 0.0001, 0.0005\}$. We set the reward exponent β to 8. K is the number of base arms to compose the super arm selected from $\{1, 1/2, 2, 2/3, 3, 3/4, 4\}$, where $x/x+1$ denotes we can choose x or $x+1$ base arms as a super arm. Please note that the RNA1 environment we use is a little different from Teacher(Kim et al., 2024c). Our environment has 1590 modes in total, but Kim et al. (2024c) has 8967 modes. We replace the environment file of Teacher with our environment file to make a fair comparison. The environment is constructed following Kim et al. (2023).

H.1 DIFFERENT SETTINGS: EXPONENT β

Reward exponentiation, $R_{\text{tar}}(x) = R(x)^{\beta}$, biases the GFlowNet target toward higher-reward candidates (Bengio et al., 2021). Larger β can increase numerical instability and reduce diversity (Lau et al., 2024). It also changes the learned sampling policy and hence the feedback available to the subspace controller, even when the terminal reward function used for reporting performance is fixed. Figures 9, 10, and 11 compare the evaluated settings.

On L14-RNA1, models with different β values consistently identified over 100 distinct modes. The $\beta = 20$ setting performed best in mode discovery and achieved a higher average reward than the other settings. Performance was poorer with $\beta = 1$. All settings gave comparable top-1000 rewards.

On L14-RNA2, all models identified fewer modes than on L14-RNA1, indicating that mode discovery was more difficult. The ranking of settings was similar, with $\beta = 20$ performing best across all metrics.

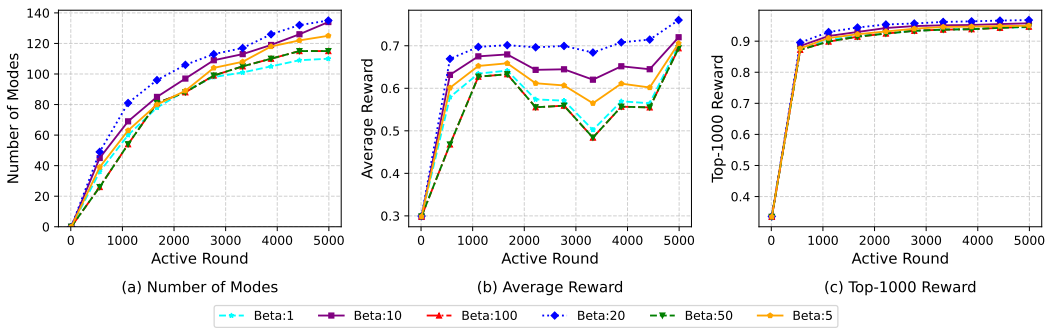


Figure 9: **Experimental result on RNA-Binding Task 1 with different β.**

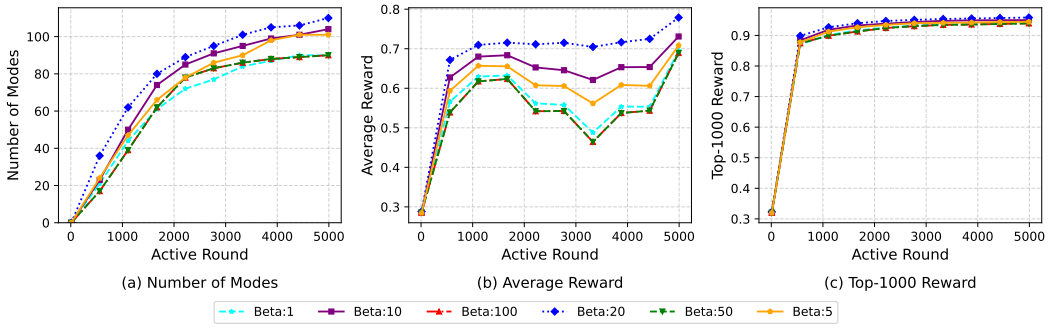


Figure 10: **Experimental result on RNA-Binding Task 2 with different β.**

L14-RNA3 was the most difficult task, with fewer discovered modes and lower average rewards than the other tasks. Here, $\beta = 10$ gave the best overall performance. The $\beta = 20$ setting achieved a higher average reward but was less stable. In this harder task, a very large β may be suboptimal because prioritizing high-reward candidates too strongly can neglect mid-reward solutions and increase the risk of mode collapse.

I ILLUSTRATION OF WORKFLOW

Here, we present a case study to illustrate how the proposed method works in the bit sequence generation task. The set M is constructed by randomly combining substrings derived from the base patterns $00000000, 11111111, 11110000, 00001111, 00111100$. For $K = 4$, exhaustive evaluation in this synthetic setting identifies $S = \{0000, 1111, 1100, 0011\}$ as the best super arm, which achieves perfect mode identification within its substate space with an average distance of 0.0 to all target modes.

The arm-feature trajectories in Figure 3 show increasing reward association for 0000. Consider replacing 0001 in $S = \{0000, 1111, 1100, 0001\}$ with 0011 to obtain the reference set above. Algorithm 1 evaluates such combinations through the learned set scorer using action identities and exploration-adjusted arm features. Selection also depends on greedy construction and the dropout bonus, so it cannot be attributed to an individual UCB score alone. Samples collected under the next selected set update its interval reward label, while separate unrestricted evaluation updates the eligible arm features. The reference set describes this synthetic task's structure, not a proven optimum for every learned-policy checkpoint.

Even when a best set is known in this synthetic task, uncertainty-based selection may revisit alternative sets. The explicit random-action component preserves trajectory support, but does not imply that every super arm is selected or that every mode is found within the experimental budget.

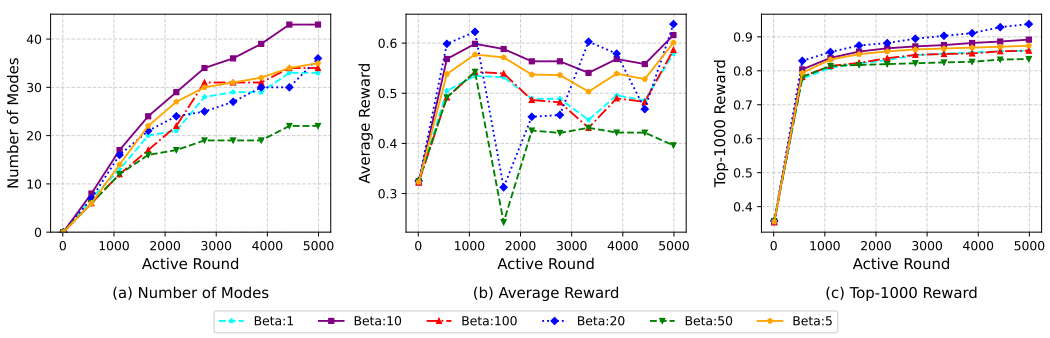


Figure 11: **Experimental result on RNA-Binding Task 3 with different β.**

Table 8: Profiling breakdown of runtime and memory consumption on Molecule Design (per 100 rounds).

Method	Sampling Time	Training Time	Update Time	Other Time	GPU Memory	CPU Memory
CBFlowNet (10^2)	61.07 s	308.90 s	0.03 s	151.9 s	4125.0 MB	2226.9 MB
CBFlowNet (10^4)	61.69 s	311.23 s	0.15 s	153.1 s	4148.3 MB	3051.2 MB
GFlowNet	50.70 s	310.64 s	0.00 s	153.7 s	4112.4 MB	2217.3 MB

J TIME AND GPU MEMORY CONSUMPTION

Table 8 profiles runtime and memory in molecule generation. The parenthetical values 10^2 and 10^4 denote the scale of arm bookkeeping in these profiling configurations, not the selected-set size K. GFlowNet training time is similar across methods ($\sim$311 s per 100 rounds). Controller overhead includes unrestricted evaluation, arm-feature updates, and learned set selection. Because the reported protocol adds 100 evaluations per 400 sampling steps, matching training rounds does not match total reward calls. In these measurements, most additional time is spent sampling, GPU memory increases by about 13–52 MB, and CPU memory increases by at most $\sim$850 MB.

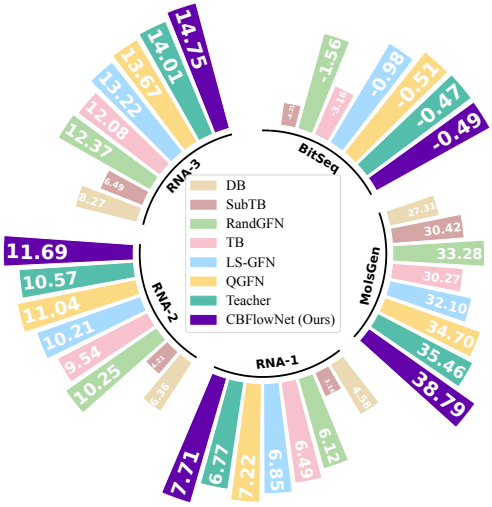


Figure 12: Experiment results on ELBO.

K EVIDENCE LOWER BOUND (ELBO)

For strictly positive terminal rewards, let $P_F(\tau) = \prod_j P_F(s_j \mid s_{j-1})$, $P_B(\tau \mid x) = \prod_j P_B(s_{j-1} \mid s_j)$, and $Z = \sum_x R(x)$. Normalized backward policies on a finite rooted DAG define the path distribution

$$Q(\tau) = \frac{R(x)P_B(\tau \mid x)}{Z}.$$

Whenever P_F is absolutely continuous with respect to Q,

$$\mathcal{L}_{\mathrm{ELBO}} = \mathbb{E}_{\tau \sim P_F}\left[\log R(x) + \log P_B(\tau \mid x) - \log P_F(\tau)\right] = \log Z - \mathrm{KL}(P_F \| Q) \leq \log Z. \quad (25)$$

Thus, for trajectories sampled from the unrestricted forward policy P_F, the corresponding Monte Carlo estimator is

$$\widehat{\mathcal{L}}_{\mathrm{ELBO}} = \frac{1}{M}\sum_{i=1}^{M}\left[\log R(x_i) + \sum_{j=1}^{n_i}\log P_B(s_{j-1} \mid s_j) - \sum_{j=1}^{n_i}\log P_F(s_j \mid s_{j-1})\right]. \quad (26)$$

Here, R must be the reward defining the fitted target, including any reward exponent when applicable. This unweighted estimator requires samples from the unrestricted forward policy. Samples collected from a gated behavior policy do not directly estimate the above expectation.

Figure 12 reports the ELBO as an auxiliary diagnostic rather than as the primary objective of CBFlowNet. Under the definition above, a larger ELBO corresponds to a smaller KL divergence between the unrestricted forward trajectory distribution and the global reward-proportional path distribution Q. However, global distribution matching is not the main finite-budget objective of CBFlowNet.

CBFlowNet deliberately biases the training visitation distribution toward multiple currently promising action-induced subspaces. This bias is intentional: the goal is to allocate a limited sampling and training budget more effectively toward discovering diverse high-reward candidates, rather than to preserve the same global exploration pattern as an unrestricted GFlowNet throughout training. Consequently, ELBO should not be interpreted as the principal criterion for comparing CBFlowNet with methods whose primary behavior remains global reward-proportional exploration. In particular, a method can achieve better finite-budget high-reward mode discovery without necessarily achieving uniformly better global distributional approximation at every stage of training.

The ELBO results should therefore be viewed as evidence that the learned GFlowNet remains compatible with the underlying reward-proportional target, rather than as the explanation for CBFlowNet's performance gains. The main empirical claim of CBFlowNet concerns efficient discovery of multiple high-scoring regions under a finite sampling budget.