Solution Augmentation for ARC-AGI Problems Using GFlowNet: A Probabilistic Exploration Approach

Anonymous authors
Paper under double-blind review

Abstract

One of the core challenges in building general reasoning systems lies in generating diverse, human-aligned solution trajectories—different yet valid paths by which a problem can be solved. Prior approaches often rely on handcrafted templates, rule-based augmentations, or human demonstrations, which are limited in scalability and stylistic diversity. To address this, we explore the use of Generative Flow Networks (GFlowNets) for automated solution augmentation in reasoning tasks. We propose a framework that learns to generate diverse reasoning trajectories with probabilities proportional to their quality, guided by a human-inspired reward function and a novel geometric forward policy. This enables the generation of multiple plausible solution paths without relying on manual supervision. We evaluate our framework on the Abstraction and Reasoning Corpus (ARC-AGI), a benchmark designed to test compositional and abstract reasoning. Our results show that GFlowNets can effectively explore the space of valid reasoning processes, producing trajectories that are diverse, concise, and consistent with human reasoning patterns. These findings suggest that GFlowNets offer a promising foundation for modeling structured reasoning in automated trajectory generation. Our code is here: `https://anonymous.4open.science/r/GFN_to_ARC-B500/`

1 Introduction

One of the central challenges in developing intelligent systems is endowing them with the ability to reason—to solve novel problems by composing abstract concepts, drawing inferences, and planning intermediate steps. Unlike perception or pattern recognition, which can often be improved through large-scale data and end-to-end optimization, reasoning requires an explicit modeling of multi-step, structured processes. In humans, reasoning is not just about arriving at correct conclusions, but about navigating diverse reasoning paths (trajectories) toward a solution. This diversity—different strategies, representations, or step sequences—plays a critical role in building a robust understanding and generalization. Hence, to improve reasoning capabilities in AI models, it is essential not only to optimize for the final answer, but also to model and expose the system to the wide range of processes by which those answers can be reached (Wei et al., 2022; Wang et al., 2023).

Recent research increasingly emphasizes the value of exposing language models to explicit reasoning trajectories—structured sequences of intermediate steps guiding problem-solving. This includes prompt-based strategies such as Chain-of-Thought (CoT) (Wei et al., 2022) and Tree-of-Thought (ToT) (Yao et al., 2023), sampling-based methods like Self-Consistency (Wang et al., 2023) and Best-of-N selection (Liu et al., 2025a), and training-time approaches that filter or reward high-quality reasoning paths (Zelikman et al., 2022; Luo et al., 2024a; Wang & Su, 2015; Jiang et al., 2024). More recently, test-time iterative refinement has gained attention: models like Reflexion (Shinn et al., 2023), DeepSeek-R1 (AI, 2024), and OpenAI's o1 (OpenAI, 2025; 2024) generate, evaluate, and revise their own reasoning traces in real-time. While these trends have yielded significant progress, most approaches still rely on prompt engineering, heuristic sampling, or post-hoc selection—limiting scalability and process-level alignment. In contrast, our work introduces a generative framework that learns to produce diverse reasoning trajectories in a unified and scalable manner.

An emerging consensus across recent studies is that reasoning performance improves when models are trained or prompted with diverse, high-quality solution trajectories. Much like how humans benefit from seeing multiple approaches to a problem, models learn more robust strategies when exposed to varied reasoning processes. However, despite this growing recognition, generating such diverse reasoning trajectories remains a fundamental bottleneck. Most current methods rely on manually curated prompts, handcrafted templates, or rule-based augmentations that limit the scope and diversity of solution paths. Some approaches collect human-written solutions (Park et al., 2023; Shim et al., 2024), but these are costly, narrow in style, and difficult to scale. As a result, models trained in this way may struggle when encountering unfamiliar or structurally novel reasoning tasks.

This leads us to a central research question: *Can we develop an automated framework that can generate diverse reasoning trajectories—without relying on handcrafted rules or human-written demonstrations—and use them to enhance model reasoning performance?* Such a framework would ideally be flexible enough to explore multiple plausible paths for the same problem, allowing the model to learn how to reason in more general, compositional, and interpretable ways.

To address this challenge, we explore a generative framework based on **Generative Flow Networks (GFlowNets)** (Bengio et al., 2021; 2023). GFlowNets are probabilistic models designed to generate a diverse set of trajectories with probabilities proportional to their reward, making them naturally suited for learning distributions over reasoning processes. Rather than collapsing onto a single "optimal" path, GFlowNets can explore diverse high-reward paths, each representing a different way to reach the solution. This probabilistic exploration mechanism makes GFlowNets especially appealing for reasoning tasks, where solutions can take many different—but equally valid—forms. By controlling the reward signal and policy behavior, we can guide GFlowNets to generate diverse reasoning paths that balance diversity, efficiency, and correctness.

To evaluate the effectiveness of our proposed framework, we conduct experiments on the **Abstraction and Reasoning Corpus (ARC-AGI, hereafter ARC)** (Chollet, 2019), a benchmark specifically designed to test reasoning capabilities. ARC consists of few-shot, grid-based tasks where each instance requires the solver to infer an underlying transformation rule from a small number of input–output demonstrations. While the task format is visually simple, solving ARC problems often demands *compositional reasoning, abstract generalization, and logical coherence*. Indeed, ARC provides a unique lens through which to assess these dimensions of reasoning—*compositionality, productivity, logical coherence* (Lee et al., 2025), and the ability to generalize from limited examples.

Importantly, ARC's structure presents a particularly suitable environment for evaluating our solution augmentation framework. First, the dataset is inherently underspecified: multiple reasoning trajectories can lead to the same correct output, making it ideal for exploring the benefits of diverse solution generation. Second, the limited data regime and absence of predefined task categories emphasize the need for flexible and generalizable reasoning strategies—exactly the kind our method aims to generate. Finally, since each ARC task typically requires multi-step abstract inference rather than surface-level pattern recognition, improvements in trajectory-level modeling are more likely to translate into tangible performance gains. These characteristics make ARC a compelling and challenging testbed for assessing the effectiveness of learning and leveraging diverse solution trajectories.

Contributions. Our main contributions are summarized as follows:

1. **GFlowNet-Based Reasoning Augmentation:** We propose the first framework to use GFlowNets for generating diverse reasoning trajectories, enabling automated solution augmentation for reasoning-intensive tasks.

2. **Human-Inspired Policy and Reward Design:** We introduce two architectural innovations—a geometric-forward action policy and a goal-conditioned reward function—that explicitly encode human reasoning biases into the learning process.

3. **Empirical Evaluation on ARC:** We demonstrate through extensive experiments on ARC that our method improves both solution diversity and accuracy compared to existing approaches, particularly in low-data regimes.

2 Background

2.1 Reasoning with Diverse Solutions

Effective reasoning involves not just arriving at the correct answer, but also navigating structured and coherent intermediate steps. Many tasks—particularly those involving abstraction, logic, or multi-hop inference—admit multiple valid solution trajectories, each representing a different way to solve the same problem. This diversity is analogous to the variation in strategies employed by humans and is essential for improving generalization, robustness, and interpretability in AI systems.

To incorporate this inductive bias, recent research has introduced techniques that explicitly model the reasoning process. Chain-of-Thought (CoT) prompting (Wei et al., 2022; Zhang et al., 2025a), Tree-of-Thought (ToT) (Yao et al., 2023), and Process Reward Models (PRMs) (Zhang et al., 2025b) supervise or sample intermediate reasoning steps to better align with human-like cognition. These methods show that exposing models to diverse reasoning paths improves their ability to generalize, particularly in tasks with sparse supervision or structural novelty. Additional evidence across various reasoning benchmarks—including GSM8K (Cobbe et al., 2021), SVAMP, and AQuA-RAT (Ling et al., 2017)—further supports this approach, demonstrating consistent performance improvements when models explore and aggregate multiple reasoning trajectories.

Complementary approaches such as automated process supervision (Luo et al., 2024a) and planning-based trajectory optimization (Jiang et al., 2024) further validate the effectiveness of learning from reasoning paths directly. These methods synthesize reward signals that capture desirable reasoning characteristics–such as conciseness, non-redundancy, or clear subgoal structure–without relying on exhaustive human annotations. Similarly, reasoning-driven process reward modeling (She et al., 2025), and step-by-step human feedback (Lightman et al., 2024) have achieved substantial performance gains by leveraging automated or semi-automated trajectory supervision. As a result, models trained with these signals exhibit more structured and interpretable problem-solving behavior, underscoring a growing consensus that trajectory-level supervision significantly enhances reasoning abilities.

This emphasis on process-level reasoning has also become a central trend in the development of large language models (LLMs). Recent models like OpenAI's *o3* series (OpenAI, 2025) and DeepSeek's *R1* (AI, 2024) incorporate internal deliberation and iterative refinement to improve reasoning performance. Likewise, Self-Consistency sampling (Wang et al., 2023) aggregates multiple reasoning chains through majority voting, achieving improved accuracy and robustness on benchmarks such as GSM8K (Cobbe et al., 2021). These examples highlight a growing consensus: modeling diverse and structured reasoning processes is key to achieving reliable and human-aligned inference.

However, many of these approaches depend on human-written demonstrations, handcrafted rules, or prompt-based sampling, which constrain scalability and stylistic diversity. Moreover, outcome-only supervision fails to distinguish between logically correct but inefficient reasoning paths and concise, interpretable ones. Without process-level reward signals, models are unable to prioritize solution trajectories that mirror human preferences.

These limitations motivate our goal of building a generative framework that can learn to produce diverse, high-quality reasoning trajectories in a scalable and automated manner. In this work, we leverage Generative Flow Networks (GFlowNets) (Bengio et al., 2021; 2023) to model solution generation as a reward-proportional sampling problem, enabling the creation of multiple human-aligned reasoning paths without requiring explicit supervision for each trajectory.

2.2 Data Augmentation for Reasoning Tasks

Data augmentation addresses the inherent scarcity of supervision in reasoning tasks and can be broadly categorized into input-output augmentation and solution-process (trajectory) augmentation.

Input-Output Pair Augmentation. Methods such as RE-ARC (Hodel, 2024), AugARC (Bikov et al., 2025), and SOLAR (Kim et al., 2024) generate new input-output examples to enhance training diversity in ARC tasks. Similar strategies, including synthetic data generation and paraphrasing, have been widely applied across various reasoning domains. For math word problems, Lu et al. (2024) leveraged question back-translation to produce synthetic examples, significantly enhancing model performance on the GSM8K benchmark. In logical reasoning, LogicAsker generated QA pairs by programmatically translating formal logic into natural language, achieving notable gains in logical consistency (Wan et al., 2024). Likewise, in code generation tasks, WizardCoder synthesized diverse coding problems and solutions, leading to state-of-the-art results on benchmarks like HumanEval and MBPP (Luo et al., 2024b). For scientific reasoning tasks, cross-lingual data augmentation using large language models produced multilingual synthetic QA pairs, demonstrating considerable improvements in cross-lingual generalization (Whitehouse et al., 2023). These input-output augmentation techniques substantially improve model generalization by diversifying the task distribution, though they typically do not explicitly expose underlying reasoning processes.

Solution-Process (Trajectory) Augmentation. Techniques explicitly augmenting reasoning trajectories leverage models' intermediate solution paths. Self-Consistency sampling (Wang et al., 2023), Tree-of-Thought deliberative search (Yao et al., 2023), and iterative refinement methods like Reflexion (Shinn et al., 2023) exemplify trajectory-level augmentation. Methods such as STaR (Zelikman et al., 2022) and Automate-CoT (Shum et al., 2023) demonstrated that training on model-generated chains-of-thought significantly boosted accuracy across various benchmarks, including commonsense and math word problems. Synthetic Prompting (Shao et al., 2023) further enriched trajectory diversity by iteratively synthesizing questions and corresponding reasoning paths, yielding substantial gains in numerical and algorithmic reasoning tasks. Similarly, frameworks like Flow of Reasoning (Yu et al., 2024) leveraged planning-based trajectory sampling to explore diverse reasoning paths, surpassing previous state-of-the-art performances on puzzle-solving benchmarks. AlphaCode's strategy of sampling numerous candidate solutions and evaluating correctness tests illustrates another successful trajectory-level augmentation approach in code generation tasks (Li et al., 2022). Collectively, these trajectory-focused methods enable models to explore and validate multiple solution paths systematically, greatly enhancing performance and interpretability.

Other Augmentation Techniques. Best-of-N (BoN) sampling and ensemble decoding represent additional augmentation strategies. BoN methods, applied in models like Minerva and AlphaCode, sample multiple candidate outputs to select the best-performing solution (Lewkowycz et al., 2022; Li et al., 2022). Ensemble decoding aggregates outputs from multiple models or reasoning chains, typically employing voting or averaging strategies. These methods primarily function at the output level and rely heavily on external evaluation metrics or heuristic criteria.

Limitations and Challenges. Despite their effectiveness, existing augmentation methods face challenges including computational scalability, genuine trajectory diversity, and the absence of reliable intermediate-step supervision. Techniques such as Tree-of-Thought and extensive sampling-based methods incur significant computational overhead (Yao et al., 2023; Wang et al., 2023). Furthermore, approaches relying on outcome-only evaluation lack explicit guidance for intermediate reasoning steps, potentially causing logical inconsistencies or inefficient reasoning pathways.

Overall, integrating diverse augmentation strategies—ranging from input-output pair expansion to trajectory-level solution enhancement—can significantly improve model robustness and generalizability. These methods collectively form an essential toolkit for addressing the inherent limitations of scarce supervision in complex reasoning tasks like ARC and beyond.

2.3 Generative Flow Networks (GFlowNets)

Generative Flow Networks (GFlowNets) (Bengio et al., 2021; 2023) are a class of generative models designed to produce diverse solution trajectories by learning from reward signals. They are particularly effective for structured generation problems where multiple valid outputs exist, such as generating human-like solutions in reasoning tasks.

Unlike standard generative models, which often produce one-shot outputs, GFlowNets construct solutions incrementally by sampling sequences of actions. This approach is inspired by reinforcement learning (RL), enabling GFlowNets to combine the exploration strengths of RL with the flexibility of generative modeling.

In contrast to conventional RL, which typically seeks a single optimal policy, GFlowNets learn a stochastic policy that assigns higher probability to trajectories with higher terminal rewards. This makes them well-suited for sparse-reward environments and problems with multiple equally good solutions, such as ARC. GFlowNets have demonstrated their utility across a wide range of structured generation tasks, including molecular design and biosequence modeling (Bengio et al., 2021; Liu et al., 2025b; Jain et al., 2022; Jang et al., 2024; Zhang et al., 2022), symbolic regression (Zhang et al., 2023b), combinatorial optimization (Jain et al., 2023; Zhang et al., 2023a), and other applications (Sendera et al., 2024; Zhang et al., 2022; Pan et al., 2023).

Flow Matching Condition and Trajectory Balance (TB) Loss. To understand the underlying mechanism of GFlowNets, it is essential to grasp the *flow matching condition*, which maintains consistency between the inflow and outflow of probability mass at any state in the state space:

$$\sum_{s'} F(s' \to s) \;=\; \sum_{s''} F(s \to s''), \tag{1}$$

where $F(s' \to s)$ denotes the probability flow from state s' to state s. This condition ensures that the forward policy $P_F(s_t \mid s_{t-1})$ (governing state transitions from an initial state to terminal states) and the backward policy $P_B(s_{t-1} \mid s_t)$ (governing transitions in reverse) remain consistent with the reward model's distribution.

This consistency is operationalized through the *Trajectory Balance (TB) loss* (Malkin et al., 2022), defined as:

$$Z \prod_{t=1}^{n} P_F(s_t \mid s_{t-1}) \;=\; R(x) \prod_{t=1}^{n} P_B(s_{t-1} \mid s_t), \tag{2}$$

where $R(x)$ represents the reward for a trajectory $x = (s_0, \ldots, s_n)$, and Z is a learnable normalization constant ensuring proper distribution over trajectories. Minimizing the squared difference of the log probabilities on both sides yields the TB loss:

$$\mathcal{L}_{\mathrm{TB}}(\theta) = \left(\log Z_\theta + \sum_{t=1}^{n} \log P_F(s_t | s_{t-1}; \theta) - \log R(x) - \sum_{t=1}^{n} \log P_B(s_{t-1} | s_t; \theta) \right)^2 . \tag{3}$$

By optimizing this objective, GFlowNets learn to sample trajectories proportionally to their rewards, naturally prioritizing high-quality trajectories. This allows effective exploration of complex solution spaces.

Training Steps and DAG Structure. The practical training loop of GFlowNets involves several key steps: (1) *Forward Sampling*, starting from an initial state and sequentially sampling states via P_F until a terminal state is reached; (2) *Reward Computation*, evaluating the sampled trajectory using the reward function $R(x)$; (3) *Backward Sampling*, optionally reconstructing the trajectory in reverse via P_B to reinforce flow matching; and (4) *Loss Computation and Parameter Update*, adjusting model parameters by minimizing the TB loss. To ensure stable training and avoid cyclic or infinite loops, state transitions are typically constrained within a Directed Acyclic Graph (DAG), limiting each state to a single visitation and ensuring convergence.

Reward Modeling and Oracle Networks. Since GFlowNets explicitly learn the distribution shaped by the reward function, accurate reward modeling is paramount. Often, rewards are assigned through pre-trained or heuristic oracle networks tailored to specific tasks. These oracle models encapsulate task-specific structural insights and desired properties, significantly enhancing the effectiveness of the generative process. For instance, in molecular generation tasks, oracle networks evaluate chemical viability, steering GFlowNet exploration toward feasible and high-quality solutions. The careful design and training of these oracle networks thus play a critical role in the performance of GFlowNets, particularly in environments with sparse or complex reward structures.

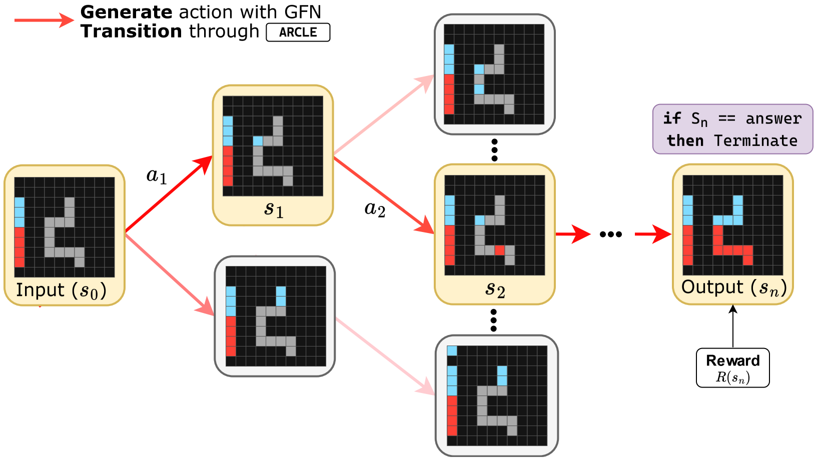


Figure 1: GFlowNet Concept Diagram for Solution Generation in ARC

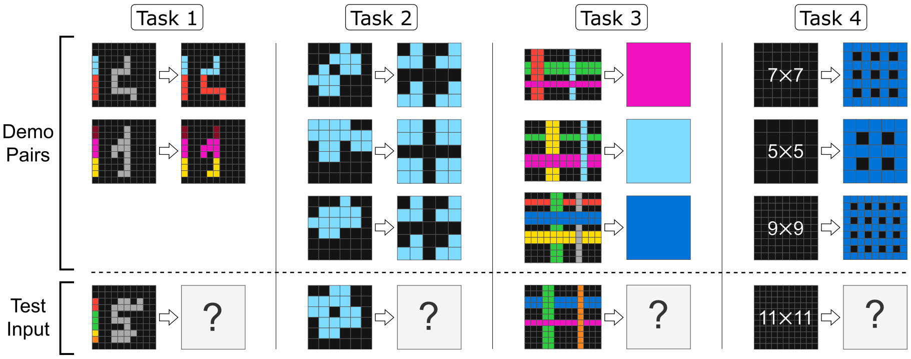


Figure 2: Some Problems of Abstraction and Reasoning Corpus (ARC)

The structural exploration capability and explicit reward modeling of GFlowNets position them uniquely to capture intricate reasoning structures and produce diverse, valid solutions in reasoning tasks. By leveraging human-inspired reward models, GFlowNets can systematically generate solution trajectories reflective of human reasoning patterns. This capability makes GFlowNets particularly suited for automated trajectory augmentation tasks, such as those required by reasoning-intensive benchmarks like ARC.

Specifically, GFlowNets directly support logical coherence through forward and backward flow consistency, ensuring generated solutions maintain logical consistency. Productivity is enabled by effective generalization from sparse examples via oracle guidance, and compositionality is facilitated by structurally exploring compositional reasoning paths evaluated by oracles.

2.4 ARC as a Benchmark for Process-Level Reasoning

The Abstraction and Reasoning Corpus (ARC) (Chollet, 2019) is a challenging benchmark comprising visual grid-based puzzles that require inferring abstract and compositional rules from very few provided examples. Each ARC task includes only 2–5 input-output examples, with no repetition of input-output pairs across different tasks. The ARC dataset was originally introduced as a proxy for evaluating core reasoning abilities of AI systems, specifically in terms of logical coherence, productivity, and compositionality (Lee et al., 2025).

Recently, despite impressive performances by state-of-the-art models such as OpenAI's O3–low, which achieved over 75% accuracy on ARC (OpenAI, 2025), performance sharply declined to below 10% on the newly introduced ARC-AGI-2 dataset (Chollet, 2024; Greg Kamradt, 2025). This drop clearly indicates the complexity of ARC and highlights the significant gap between current AI capabilities and human-like abstraction and reasoning, indirectly underscoring the distance to achieving Artificial General Intelligence (AGI).

The inherent difficulty of ARC arises primarily from its unique structure: each problem set has a single, unique input-output relationship, and these relationships vary drastically across different tasks. Therefore, conventional data augmentation methods struggle as additional synthetic examples cannot be trivially generated. Given these constraints, ARC becomes particularly suitable for testing the capability of automated solution augmentation methods. Specifically, ARC provides a rigorous environment to examine whether generative frameworks, such as GFlowNets, can effectively capture structural insights from carefully designed, human-inspired reward models, thus facilitating the discovery of diverse and valid solution trajectories under highly sparse and unique example constraints. These characteristics position ARC as an ideal benchmark for exploring and validating advanced reasoning frameworks like GFlowNets.

Conventional augmentation methods applied specifically to ARC, such as input-output pair augmentation ((Hodel, 2024; Bikov et al., 2025; Kim et al., 2024)), lack explicit trajectory-level reasoning, treating the transformation processes as black boxes. Rule-based augmentation methods, meanwhile, rely on predefined human-crafted rules, severely limiting scalability and the generalization capacity to novel, unseen problem structures. Consequently, these approaches fail to adequately address ARC's unique demands for explicit and diverse trajectory generation, highlighting the need for automated, trajectory-level augmentation frameworks such as GFlowNets.

3 Methods

3.1 GFlowNet Architecture for Human-Aligned Solution Generation in ARC

We now provide a concise overview of how our GFlowNet is structured and generates solutions tailored specifically for ARC problems. Building on the general components described in Sections 3.1–3.2, our architecture differs from conventional GFlowNet models primarily by using asymmetric sampling distributions: a geometric distribution for forward sampling to prioritize shorter, more efficient action sequences, and a categorical distribution for backward sampling to ensure effective trajectory reconstruction. Additionally, we explicitly incorporate human trajectory priors into our reward function, penalizing redundant or excessively long trajectories to enhance learning efficiency and promote human-like reasoning. In what follows, we detail this architecture and elaborate on the rationale behind these adaptations.

High-Level Process. At the start of each episode, an initial state derived from the ARC input grid is set, containing minimal information needed for the ARC task. The GFlowNet forward policy then samples actions to move from one grid configuration to another, guided by the *Geometric* distribution described above. Actions that prove too long or repetitive are penalized by our reward model, while succinct, high-quality trajectories receive higher rewards. After reaching a final state, the backward policy verifies or backtracks using a *Categorical* distribution, helping to ensure the global flow matching necessary for Trajectory Balance. Over many episodes, this bidirectional sampling process adapts the GFlowNet parameters to favor human-like solutions that meet the puzzle's requirements. We summarize the full GFlowNet training process in Algorithm 1, provided in Appendix C. The pseudocode highlights the use of geometric sampling for forward actions, categorical backward modeling, and off-policy updates with cycle-penalized reward shaping.

Algorithmic Flow. Algorithm 1 (Appendix C) outlines the main steps: *(1) Initialization* of the forward/backward networks, *(2) Forward Sampling* where actions are chosen via the Geometric distribution based on predicted logits, *(3) Reward Assignment* using a human-inspired reward model that applies discounting, cycle penalties, trajectory-length constraints, or a combination thereof, depending on the specific characteristics of the trajectory, *(4) Backward Sampling* where the backward policy samples a probable reverse action corresponding to the forward-selected action, ensuring local consistency required for trajectory balance, and *(5) Parameter Update* by minimizing the trajectory balance (TB) loss across the sampled trajectory. Repeating these steps allows the GFlowNet to efficiently converge on a diverse set of high-reward, human-like trajectories.

ARC-Specific Adaptations. Unlike many domains where actions naturally exhibit uniform or categorical distributions, ARC problems benefit from the combination of a *Geometric-forward* and *Categorical-backward* design. (Section 3.2). In our setup, each trajectory typically ends with a "Submit" action that finalizes the generated grid, triggering a binary reward if the puzzle is solved correctly. This setup not only aligns with the binary reward structure of ARC but also enables GFlowNet to efficiently prioritize high-quality, human-like trajectories within its generative process.

3.2 Human-Trajectory Guided Reward Design

GFlowNet cannot effectively learn with a purely sparse reward model. A naive sparse reward assignment—1 for correct answers and 0 for incorrect ones—fails to provide the necessary gradient to differentiate between high-quality and low-quality trajectories, severely limiting learning efficiency or even halting progress entirely.

When applying GFlowNet to the ARC problem, this limitation became apparent. For example, as shown in Figure 3, some trajectories solve the problem efficiently, while others reach the correct answer with unnecessary and repetitive actions. Assigning the same reward to both types of trajectories makes it difficult for GFlowNet to prioritize high-quality solutions.

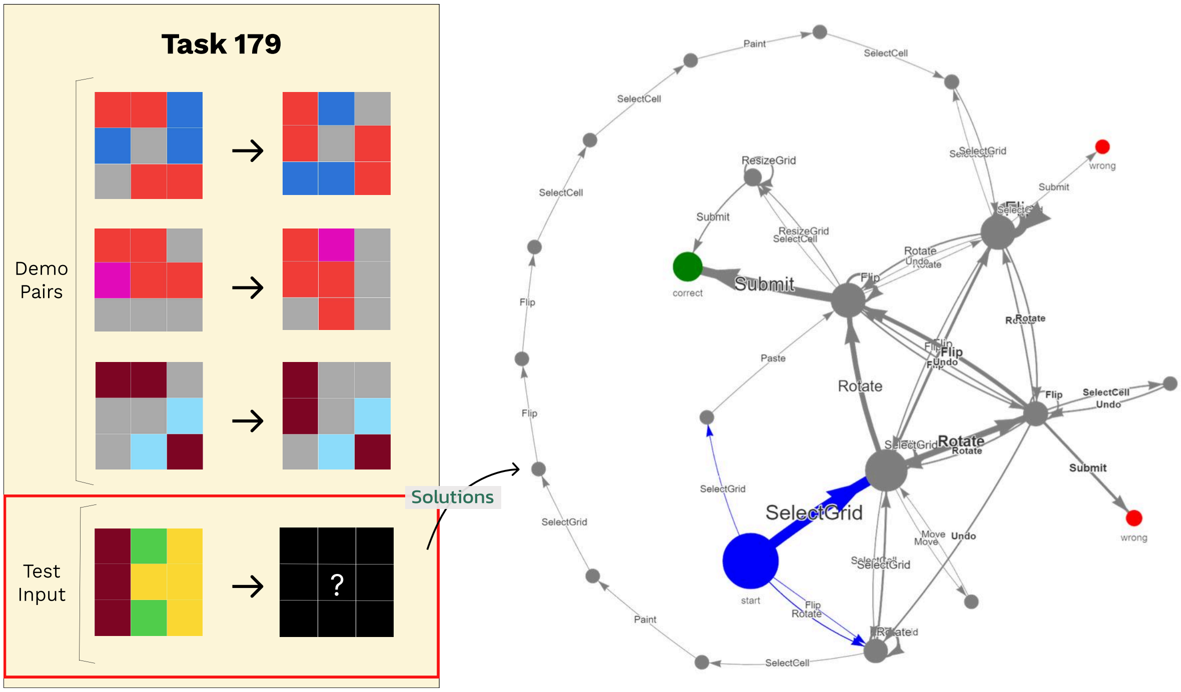


Figure 3: **State Space Graph of Trajectories for ARC Problem**. The trajectories were collected through O2ARC (Shim et al., 2024), illustrating various solution paths from the start node (blue) to the correct solution node (green). Additionally, incorrect solution nodes (red) are displayed to represent alternative, unsuccessful attempts. This figure highlights the diversity of possible solutions for a single problem, emphasizing the multiple ways in which the problem can be approached and solved.

To address this, we analyzed human trajectory data from O2ARC 3.0 (Shim et al., 2024) to identify clear quality differences and inform the design of a reward model that highlights superior trajectories. Two major characteristics of high-quality trajectories were identified:

- **Final Action:** The last action in a trajectory must always be the '*Submit*' action. This deliberate conclusion indicates that the solution is complete and allows for feedback. Trajectories that end with the Submit action are prioritized in the reward model.

- **Avoidance of Redundancy:** High-quality trajectories avoid unnecessary repetition, using only essential actions to reach the correct answer. Most general problems rarely require more than 10 steps (Kim et al., 2025). Repetitive actions, particularly in a Directed Acyclic Graph (DAG) context, can create cycles that complicate learning.

To reflect these human-derived structural preferences explicitly, we propose integrating the following mechanisms into our reward design:

Discount Factor ($R_{\mathbf{DF}}$) The first approach involves applying a discount factor γ (set to 0.9), commonly used in reinforcement learning. This method inherently favors shorter trajectories by scaling the reward R at each step t as:

$$R_{\text{DF}} = R \cdot \gamma^t \tag{4}$$

While effective in promoting brevity, this approach can negatively impact tasks that require longer solutions.

Trajectory Length Regularization ($L_{\mathbf{TLR}}$) The second strategy introduces a regularization term directly into the objective function. This term penalizes deviations from a target trajectory length L_{target}, providing flexibility while discouraging excessively long trajectories:

$$L_{\text{TLR}} = \lambda_{\text{reg}} \cdot (L_\tau - L_{\text{target}})^2 \tag{5}$$

Here, λ_{reg} controls the regularization strength. The total objective function combines this term with the Trajectory Balance Loss (L_{TB}):

$$L_{Total}(\theta) = (1 - \alpha) \cdot L_{\text{TB}} + \alpha \cdot L_{\text{TLR}} \tag{6}$$

where α balances trajectory balance and length regularization.

Cycle Detection ($R_{\mathbf{cycle}}$) Finally, cycle detection penalizes repetitive actions by flagging repeated states within a trajectory. A penalty proportional to the number of detected cycles $C(\tau)$ is applied:

$$R_{cycle}(\tau) = \begin{cases} r(S_T) - \lambda \cdot C(\tau) & \text{if a cycle is detected,} \\ r(S_T) & \text{otherwise.} \end{cases} \tag{7}$$

This encourages efficient exploration while avoiding redundant actions.

Thus, we define the reward for a trajectory $x = (s_0, \ldots, s_T)$ as:

$$R(x; y) = \mathbf{1}_{[f(x)=y]} \cdot \gamma^{L(x)} \cdot \exp\left(-\lambda_{\text{cycle}} \cdot C(x)\right),$$

where:

- $\mathbf{1}_{[f(x)=y]}$ is a binary indicator of whether the trajectory leads to a correct solution,
- $\gamma \in (0, 1)$ is a discount factor applied to the trajectory length $L(x)$,
- $C(x)$ is the number of detected cycles (i.e., repeated states) in the trajectory,
- λ_{cycle} is the penalty weight for cyclic behavior.

This reward is used as input to the Trajectory Balance (TB) loss, and is log-transformed prior to optimization.

In addition to the reward-based objective, we introduce a separate trajectory length regularization loss to further encourage concise reasoning:

$$L_{\text{TLR}}(x) = \lambda_{\text{TLR}} \cdot \left(L(x) - L_{\text{target}}\right)^2 ,$$

where L_{target} is a reference trajectory length (e.g., 3 or 4), and λ_{TLR} controls the penalty strength.

The overall training objective is then defined as:

$$L_{\text{total}} = L_{\text{TB}} + L_{\text{TLR}},$$

where L_{TB} denotes the Trajectory Balance loss computed using the log-transformed reward $\log R(x; y)$.

3.3 Forward Policy via Geometric Action Sampling

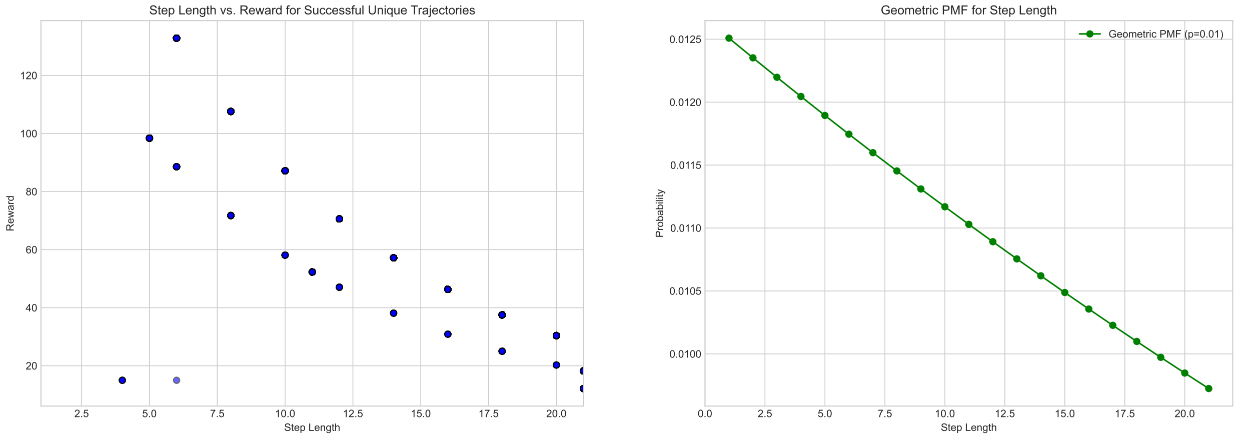


Figure 4: (Left) **Reward Distribution by Step Length**: This figure shows the reward distribution across trajectories with varying lengths for ARC problems. Higher rewards are concentrated in shorter steps, while rewards decrease as the step length increases. (Right) **Geometric PMF**: General form of PMF for the Geometric distribution.

To effectively model the distribution of reward functions derived from human trajectories, it is crucial to account for the sequential nature of actions and their dependencies in ARC problems. While conventional GFlowNets typically employ the Categorical distribution to model action probabilities, this approach fails to capture the observed relationship between reward and step length in ARC problems.

As shown in Figure 4, rewards are heavily skewed toward shorter trajectories, with diminishing rewards as step lengths increase. This motivates the use of the Geometric distribution to address the limitations of the Categorical model.

The Geometric distribution is well-suited for modeling action sequences under a framework where actions are treated as probabilistic steps toward success. Specifically, it represents the number of steps required until the first success in a sequence of trials, with a constant probability of success p. In the context of ARC problems, success can be interpreted as reaching the target state, and each action step is viewed as a probabilistic attempt to move closer to that state.

To formalize this interpretation, we treat each candidate action a_i as associated with a success probability p_i obtained via softmax over logits z_i. The number of steps until success under this distribution is modeled as a geometric random variable:

$$X_i \sim \text{Geometric}(p_i), \quad \mathbb{E}[X_i] = \frac{1}{p_i}.$$

Then, selecting the action with the minimum expected steps corresponds to:

$$a^* = \arg\min_i \mathbb{E}[X_i] = \arg\max_i p_i.$$

This provides a probabilistic justification for the use of Geometric sampling in the forward policy: it naturally prioritizes actions with higher expected efficiency in sparse reward settings. Moreover, by selecting actions based on their expected time-to-success, the policy aligns with human-like decision-making that favors shorter and more effective trajectories.

This aligns well with the structure of GFlowNet, where the model explores action sequences that maximize long-term reward under trajectory-level uncertainty. Building on the efficiency bias introduced by geometric sampling, this formulation has proven particularly effective in sparse-reward environments such as ARC. In addition, the preference toward shorter trajectories helps mitigate over-exploration of redundant or excessively long paths, which can dilute learning signals especially during early stages of training. This theoretical bias toward shorter, high-probability trajectories is further supported by empirical evidence.

The rationale for using the Geometric distribution is as follows:

- **Reward Trends in ARC problems**: Higher rewards are observed in shorter trajectories, which can be naturally modeled by the Geometric distribution's property of assigning higher probabilities to fewer steps. By prioritizing trajectories with shorter action sequences, the model effectively aligns exploration with the observed reward trends in human problem-solving.
- **Action Sequences as Trials**: Each action in a sequence can be interpreted as an independent attempt to reach the target state (success). The Geometric distribution allows the model to evaluate and prioritize actions based on their probabilities of success, guiding the GFlowNet to focus on trajectories that are both efficient and high-reward.

To integrate the Geometric distribution into GFlowNet's forward policy, the policy network outputs a logits vector $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ for a given state $\mathbf{s}$. This vector is transformed into probabilities $\mathbf{p} = (p_1, p_2, \ldots, p_N)$ via the softmax function. For each action a_i, the probability of success is modeled as a Geometric random variable $X_i \sim \text{Geometric}(p_i)$, where the selection of an action a^* is defined as:

$$a^* = \arg\min_i X_i. \tag{8}$$

Here, the action with the fewest expected steps to success is prioritized, effectively capturing the reward distribution's dependency on step length and enhancing the exploration process. This step-based probability assignment allows GFlowNet to balance between exploration and exploitation, focusing on high-reward trajectories while maintaining diversity.

Conversely, the backward policy retains the use of the Categorical distribution. While the Geometric distribution is advantageous for exploration, its properties are less suited for reconstructing paths from the goal to the start state. The backward policy instead relies on the Categorical distribution to guide path regression, where capturing dependencies between actions is less critical.

Attempts to apply the Geometric distribution to the backward policy yielded suboptimal results, as it failed to represent the necessary dependencies for effective path reconstruction. Therefore, the use of distinct distributions—Geometric for forward exploration and Categorical for backward reconstruction—is both empirically justified and theoretically grounded.

4 Experiments

4.1 Experimental Design

4.1.1 Research Questions

Our central goal is to explore whether Generative Flow Networks (GFlowNets) can be adapted to generate reasoning trajectories that are **aligned with human problem-solving heuristics** in the ARC domain. Since direct comparison with human trajectories is beyond our current scope, we instead assess *alignment* through structural properties inspired by human behavior (e.g., brevity, intentionality, loop avoidance). To this end, we pose the following sub-questions:

- **RQ1. Can reward functions informed by human priors promote human-aligned reasoning patterns?** We assess how different reward shaping techniques (e.g., brevity encouragement, loop avoidance, trajectory regularization) influence the structure and quality of generated trajectories.
- **RQ2. Does using a Geometric distribution in the forward policy enhance efficiency than conventional GFlowNets with a categorical forward policy in diverse trajectory generation?** We evaluate whether modeling forward decisions via the Geometric distribution, which prioritizes concise action sequences, improves alignment with diverse reasoning behaviors.
- **RQ3. Why does a Geometric forward policy excel in goal-conditioned reasoning tasks?** We compare forward–backward policy combinations (G–C, G–G, C–G, C–C) to uncover how the inductive bias of a Geometric forward policy, together with a stabilizing backward policy, leads to higher success and diversity under sparse rewards.

4.1.2 Dataset: Task Selection Rationale

The dataset used in our experiments is based on the Abstraction and Reasoning Corpus (ARC), specifically focusing on Task 178 (*Diagonal Flip*) for most experiments. This task exemplifies problems where the solution requires transformation of the entire grid, rather than localized subregions, making it suitable for evaluating solution augmentation in full-grid reasoning contexts.

Formally, the selected tasks satisfy:

$$f(x_k) = y_k, \quad S = G, \quad \text{where } G \in \mathbb{R}^{m \times n} \tag{9}$$

Here, S denotes the selection region and G represents the entire grid with dimensions $m \times n$. By defining $S = G$, each action operates on the whole grid, requiring full-grid comprehension and transformation.

To further constrain the solution space and evaluate the impact of full-grid transformations, we restrict our experiments to ARC problems solvable via *entire-grid* operations—such as flips, rotations, translations, and global color replacements. This design choice is both practically and theoretically motivated. In the ARCLE environment (Lee et al., 2024), the default formulation allows any subset of the grid—including individual pixels or subgrids—to be transformed. However, this leads to an exponentially large action space of roughly $O(A^L \times 2^{mn})$ for a grid of size $m \times n$, where A is the set of atomic actions and L is the typical sequence length. For a 30×30 grid, the subregion selection space alone reaches 2^{900}, rendering effective exploration intractable. By instead focusing on whole-grid transformations (i.e., $S = G$), we reduce the action space complexity to approximately $O(A^L)$, which not only enhances tractability but also aligns more closely with the structural reasoning patterns observed in human trajectories.

In addition to Task 178, other tasks solvable by whole-grid selection include: 53, 87, 129, 140, 150, 155, 179, 241, 322, 339, 346, 355, 380, and 385 are tested. These are shown in appendix B.4.

4.1.3 Evaluation Metrics

The following metrics are used to evaluate the performance of GFlowNet:

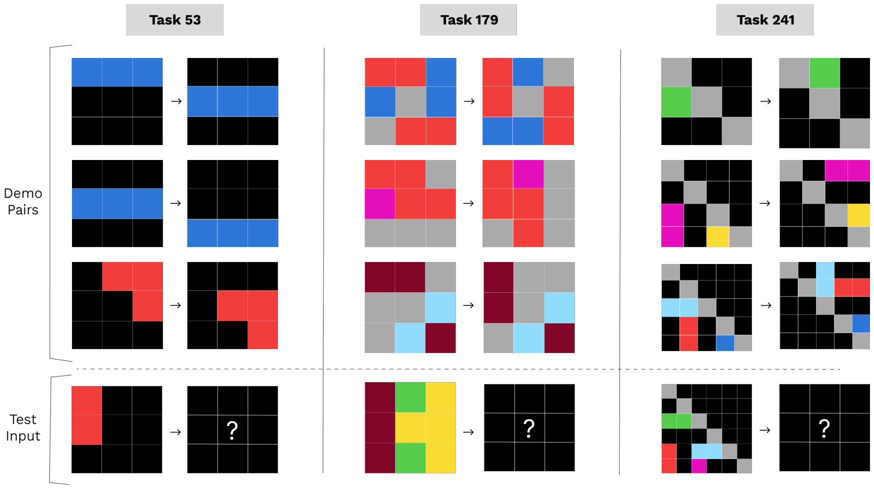


Figure 5: Example of an Entire grid selection ARC problem

- **Validation Accuracy (Val_ACC)**: The proportion of correctly generated trajectories among 100 attempts. A trajectory is considered correct if it reaches the target solution while adhering to the task constraints. This metric evaluates the model's ability to generate valid and successful trajectories efficiently.
- **Trajectory Diversity ($D_{\mathbf{traj}}$)**: The fraction of unique successful trajectories among all generated unique trajectories. A higher diversity score indicates that the model explores multiple valid solutions rather than converging on a single mode. This reflects the model's ability to balance exploration and exploitation in trajectory generation.
- **Reward Distribution Diversity ($D_{\mathbf{reward}}$)**: The variation in assigned rewards, measured using the Shannon Index. A higher value suggests that the model assigns rewards more evenly across different trajectories, encouraging diverse exploration. Conversely, a lower value indicates that rewards are concentrated on a limited set of trajectories, leading to mode collapse.

4.1.4 Hyperparameters and Glossary

Below are Table 1, which presents key hyperparameter settings, and Table 2, which defines key abbreviations used throughout the paper.

Hyperparameter	**Value**
Learning Rate (lr)	10^{-4}
# of Actions	5
Episode Length	10
Base Reward	15 (O), 0 (X)
Discount Factor (γ)	0.9
Loss Weight (α)	0.2
Trajectory Regularization Weight	0.01

Table 1: Key Hyperparameter Settings

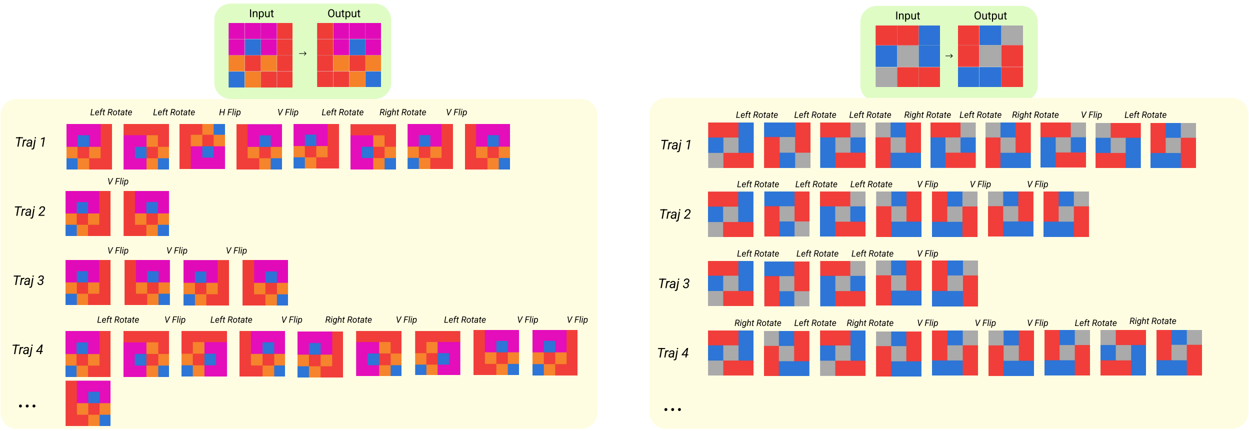
Abbreviation	**Definition**
P_F	Forward Probability
P_B	Backward Probability
TLR	Trajectory Length Regularization
DF	Discount Factor
cycle	Cycle Penalty Reward
Splus	Submit plus Reward
L_{TB}	Trajectory Balance Loss

Table 2: Glossary of Terms and Abbreviations

4.2 Performance of Revised GFlowNet for Solution Augmentation

This section evaluates the effectiveness of our proposed GFlowNet modifications across the three research questions defined in Section 4.1.

To qualitatively illustrate the generated trajectories discussed throughout our experiments, we present examples from Task 150 and Task 179—two ARC problems included in our benchmark suite (Appendix B.4). These examples highlight the variety of transformation sequences leading to correct solutions.



(a) Task 150

(b) Task 179

Figure 6: Visualization of generated trajectories for two ARC tasks. Each trajectory represents a sequence of transformations applied to the input grid to achieve the correct output. Task 150 (a) and Task 179 (b) demonstrate diverse action sequences, including rotations and flips, leading to successful solutions.

4.2.1 RQ1. Can reward functions informed by human priors promote human-aligned reasoning patterns?

To investigate whether human-inspired reward signals can effectively shape the structure of generated trajectories, we evaluate five reward models, each designed to encode specific reasoning heuristics:

- **Base**: Sparse reward only (no inductive bias).
- **Splus**: Emphasizes the Submit action.
- **Cycle**: Penalizes repeated actions (loop avoidance).
- **DF**: Discount factor encourages early goal completion.
- **TLR**: Penalizes long trajectories (conciseness).

Results and Analysis We first analyze the effects of individual reward models (Tables 3 and 4). The Base model performs poorly, with only 5.26% unique success and reward entropy of 0.29, reflecting ineffective trajectory exploration. Among single reward settings, the DF model achieves the highest total success rate (**89.33%**) but with relatively low diversity (1.04), suggesting that its strong preference for short paths leads to early convergence on a narrow solution set.

TLR achieves lower total accuracy (67.33%) but the highest reward entropy (**1.08**), producing the most structurally diverse successful trajectories (**50** in Table 3). This suggests that TLR allows for a wider range of valid solution strategies while still guiding the model toward goal-oriented reasoning.

Cycle, while reflecting the human tendency to avoid repetition, generates the largest number of unique trajectories (**100**), but with relatively low success (32 successful), indicating that diversity alone does not imply effective reasoning. Splus, despite its alignment with the task-specific "submit" action, shows limited impact (13.68% accuracy), implying that local task signals are insufficient without broader structural constraints.

We next examine reward combinations to assess whether structural biases interact synergistically (Tables 5, 6). DF + Cycle + TLR achieves perfect total success (**100%**), but with reduced diversity (entropy = 1.81),

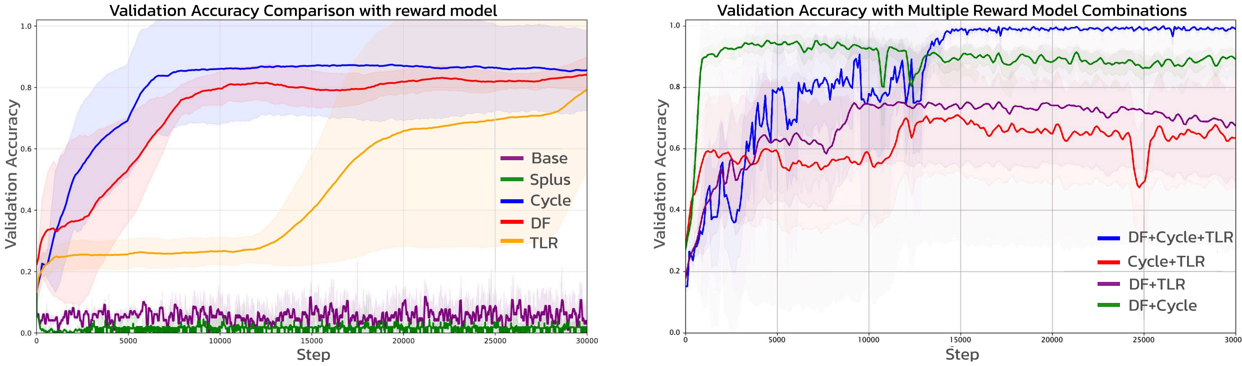


Figure 7: (Left) Validation accuracy for individual reward models. (Right) Validation accuracy for combined reward models.

Reward	Total Unique Trajectories	Successful Unique Trajectories	Val_ACC (%)
DF	36	**27**	**75.47**
Cycle	**100**	32	32.00
TLR	79	**50**	63.03
Base	76	4	5.26
Splus	95	13	13.68

Table 3: Success rate based on unique trajectories for different reward models.

Reward	Total Trajectories	Total Successful Trajectories	Val_ACC (%)	D_{reward}
DF	300	**268**	**89.33**	1.04
Cycle	300	96	32.00	0.90
TLR	300	202	67.33	**1.08**
Base	300	15	5.00	0.29
Splus	300	39	13.00	0.56

Table 4: Success rate and reward diversity based on total trajectories.

suggesting a strong convergence effect. DF + TLR, in contrast, yields a balance of high success (86%) and the highest diversity among all models (**2.96**), highlighting its effectiveness at preserving reasoning flexibility while maintaining performance.

Notably, combinations that include Cycle often lead to more constrained trajectory spaces. For example, Cycle + TLR achieves 89% success but with the lowest diversity (0.92), reinforcing that while repetition avoidance is important, overly rigid enforcement may limit the model's exploration. These results support the notion that reward combinations do not simply stack effects but influence trajectory patterns in complex and often non-linear ways.

Reward Combination	Unique Trajectories	Successful Unique Trajectories	Val_ACC (%)
DF + Cycle + TLR	5	5	**100.00**
DF + TLR	35	26	74.29
Cycle + TLR	38	29	76.32
DF + Cycle	47	29	61.70

Table 5: Success rate and reward diversity based on unique trajectories for reward combinations.

Across all experiments, the average trajectory length was approximately 7 steps, close to the episode maximum of 10. In contrast, human-provided solutions for similar ARC tasks often require fewer than 4 steps on

Reward Combination	Total Trajectories	Successful Total Trajectories	Val_ACC (%)	D_{reward}
DF + TLR	100	86	86.0	**2.96**
DF + Cycle + TLR	100	100	**100.0**	1.81
Cycle + TLR	100	89	89.0	0.92
DF + Cycle	100	91	91.0	1.50

Table 6: Success rate and reward diversity based on total trajectories for reward combinations.

average (Kim et al., 2025). This further underscores the role of reward shaping in steering the model toward concise, human-aligned reasoning structures.

Implications These findings suggest that well-designed reward functions not only affect trajectory success but also influence the underlying structure and interpretability of the reasoning process. Individual rewards serve as proxies for distinct human problem-solving heuristics, and their combinations can shape exploration in either beneficial or restrictive ways.

While more constraints can increase success, they may also reduce diversity and generalization. For instance, the DF + TLR model demonstrates a favorable trade-off between structure and flexibility, whereas overly rigid combinations like DF + Cycle + TLR lead to narrower solution modes. These results support the idea that reward shaping is not merely about scoring optimization, but about modeling the reasoning biases that underlie human-like solution strategies.

Rather than claiming full cognitive alignment, our work shows that GFlowNet can be guided to reflect aspects of human reasoning—brevity, efficiency, and structural clarity—through reward design. Future research should explore adaptive reward weighting and training-phase scheduling to dynamically manage these trade-offs in more complex or open-ended reasoning domains.

4.2.2 RQ2. Does using a Geometric distribution in the forward policy enhance efficiency in diverse trajectory generation?

We compare two GFlowNet variants that differ only in the choice of forward policy distribution: the original version using a Categorical distribution, and the revised version using a Geometric distribution. In both cases, the backward policy (P_B) remains Categorical.

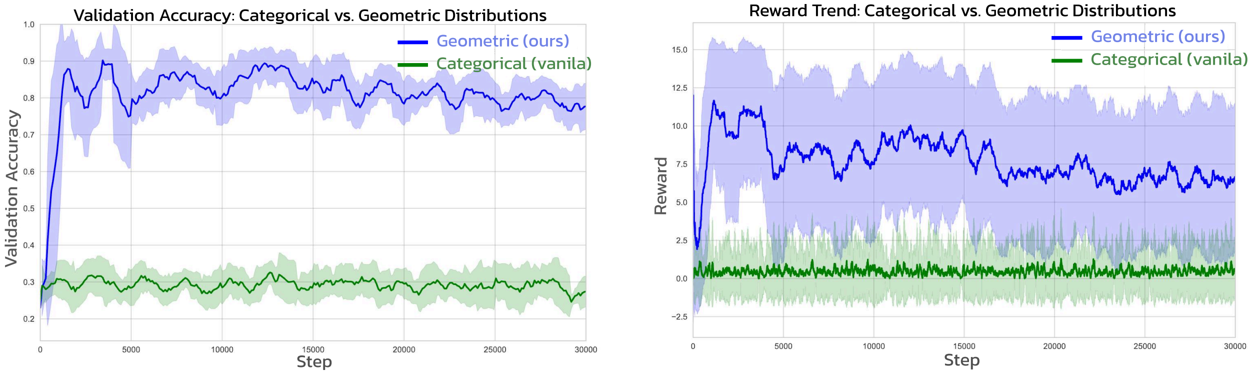


Figure 8: (Left) Validation accuracy comparison between Categorical (green) and Geometric (blue) distributions. (Right) Reward profiles comparison over training steps.

Results and Analysis The Geometric PF GFlowNet introduces an inductive bias toward concise and decisive action sequences—an intuition derived from analyzing human trajectory patterns, particularly in structured reasoning tasks like ARC. Unlike the original Categorical forward policy, which samples actions

Distribution	**Unique Trajectories**	**Successful Unique Trajectories**	**Val_ACC (%)**
Categorical	75	10	13.33
Geometric	45	**36**	**80.00**

Table 7: Success rate based on unique trajectories for Categorical and Geometric distributions.

Distribution	**Total Trajectories**	**Successful Total Trajectories**	**Val_ACC (%)**	D_{reward}
Categorical	100	28	0.28	**3.38**
Geometric	100	**89**	0.89	0.88

Table 8: Success rate based on total trajectories for Categorical and Geometric distributions.

uniformly, the Geometric distribution prioritizes earlier actions, naturally biasing the model toward shorter trajectories.

As shown in Figure 8, the Geometric PF GFlowNet consistently achieves higher validation accuracy throughout training. Table 7 confirms that, while the Categorical policy produces a larger number of unique trajectories (**75** vs. 45), it leads to significantly fewer successful outcomes (10 vs. **36**). Likewise, Table 8 shows that the Geometric model outperforms in overall success rate (**89%** vs. 28%), despite exhibiting lower reward diversity ($D_{reward} = 0.88$ vs. **3.38**).

This contrast highlights that, although the Categorical policy exhibits broader surface-level exploration, much of it fails to reach valid solutions. In other words, diversity without effectiveness may not equate to useful reasoning. In contrast, the trajectories generated by the Geometric PF GFlowNet are more likely to be both correct and structurally sound. Notably, successful trajectories under the Geometric policy typically reach the goal within 2 to 4 steps—well below the episode limit of 10—demonstrating the model's inherent preference for efficient reasoning paths.

These paths exhibit key characteristics commonly associated with human problem-solving behavior: minimal reversals, consistent transformation patterns, and clear subgoal alignment. Although we do not claim full cognitive equivalence, this emergent structure indicates a meaningful alignment between the model's exploratory behavior and human-like reasoning strategies.

Implications These results suggest that leveraging distributional biases—such as those imposed by the Geometric forward policy—can steer solution augmentors toward generating concise and interpretable reasoning paths. While the diversity of the Categorical model appears greater numerically, the Geometric PF GFlowNet better balances success and efficiency, especially in sparse-reward settings like ARC.

Moreover, the fact that this model can discover multiple valid reasoning paths for ARC-178—a task solvable in just a few steps—suggests that even simple inductive biases can facilitate structured solution generation. When combined with human-inspired reward functions (Section 4.2.3), this opens the possibility of extending the framework to more complex reasoning domains. Rather than replicating human reasoning in its entirety, the proposed approach reflects an early but promising step toward learning structural features of expert reasoning—such as brevity, decisiveness, and goal alignment—and embedding them within generative solution frameworks.

4.2.3 RQ3. Why does a Geometric forward policy excel in goal-conditioned reasoning tasks?

We evaluate different combinations of forward (P_F) and backward (P_B) policies:

- **G-C:** Geometric for P_F and Categorical for P_B.
- **G-G:** Geometric for both P_F and P_B.
- **C-C:** Categorical for both P_F and P_B.

- **C-G:** Categorical for P_F and Geometric for P_B.

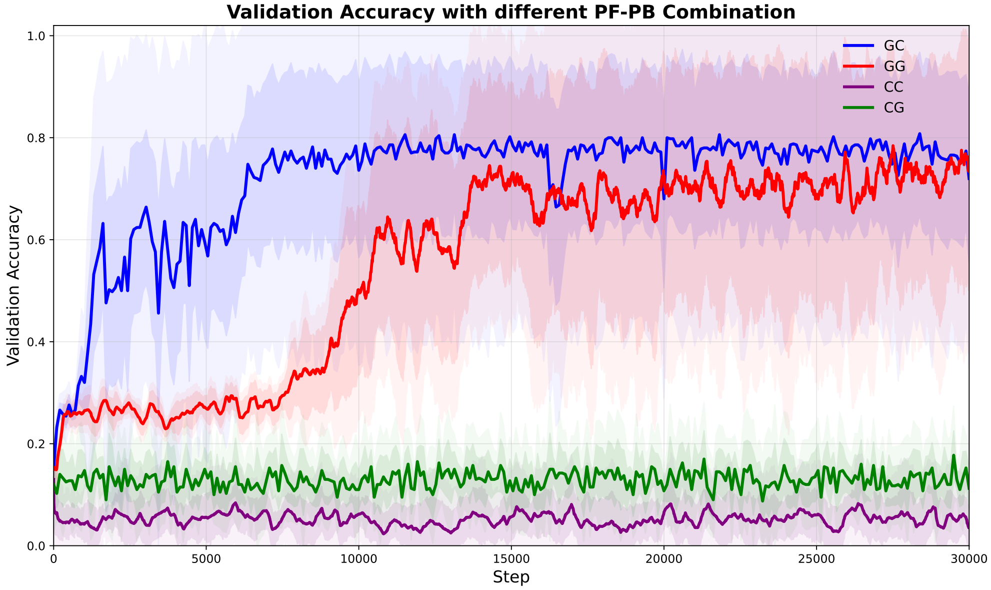


Figure 9: Validation accuracy for different P_F-P_B combinations.

P_F-P_B	**Unique Trajectories**	**Successful Unique Trajectories**	**Val_ACC** (%)
G-G	21	18	**85.71**
G-C	56	**49**	**87.50**
C-G	59	10	16.95
C-C	**75**	10	13.33

Table 9: Unique trajectory performance for different P_F-P_B combinations.

P_F-P_B	**Total Trajectories**	**Successful Total Trajectories**	D_{reward}
G-G	100	**97**	**2.15**
G-C	100	**97**	**2.09**
C-G	100	11	0.73
C-C	100	10	3.39

Table 10: Overall success rates for different P_F-P_B combinations.

Results and Analysis We compare four configurations of forward (P_F) and backward (P_B) policies. Both G-G and G-C achieve the highest total success rate (97.0%), but the G-C pairing yields a significantly higher number of successful unique trajectories (49 vs. 18), indicating better structural generalization. C-G and C-C perform poorly, both in accuracy and trajectory success.

This pattern suggests that the forward and backward policies contribute differently to learning dynamics. The forward policy (P_F) is responsible for generating exploratory trajectories, while the backward policy (P_B) adjusts the reverse probabilities to match the desired flow based on reward signals. When both policies are heavily biased (e.g., G-G), the backward policy may no longer act as a corrective mechanism. Instead, it may reinforce the forward bias, narrowing the trajectory space and reducing diversity.

In contrast, the G-C setup preserves high success while also producing a wider range of successful solutions. This indicates that an asymmetric policy design—where P_F is biased for efficient exploration and P_B remains uniform to stabilize flow alignment—achieves better reasoning diversity without sacrificing performance. The observed D_{reward} values support this: G-G has slightly higher entropy (2.15) than G-C (2.09), but the diversity of correct solutions is much higher in G-C.

We also observe that C-C has the highest reward entropy (3.39) but fails to achieve meaningful performance, indicating that reward distribution diversity alone does not imply reasoning quality. High entropy may reflect exploratory breadth, but it must be interpreted alongside success metrics. Ultimately, reward diversity without trajectory effectiveness may signal unstructured exploration, not reasoning robustness.

Insights These findings reinforce the importance of decoupling exploration and correction roles in bidirectional learning. The forward policy drives goal-directed trajectory generation, while the backward policy should facilitate stable reward-aligned flow learning. We hypothesize that if both policies impose strong, aligned biases (as in G-G), the resulting feedback loop can distort the reward structure and impair the model's ability to generalize across diverse solutions.

Moreover, a successful unique trajectory count emerges as a valuable proxy for structural reasoning diversity. Models such as G-C not only succeed frequently but also do so via a wider variety of valid solution paths. This suggests that asymmetric policy pairing enables the model to learn broader reasoning structures—a key capability in ARC-like tasks where flexible generalization is crucial.

Finally, reward distribution diversity, as captured by entropy, offers useful information about the model's exploratory behavior, but does not alone indicate reasoning effectiveness. Reward entropy should be interpreted in conjunction with success metrics to evaluate the utility—not just the scope—of the model's exploration.

5 Conclusion

In this study, we proposed a novel GFlowNet-based framework for solution augmentation in ARC problems. By leveraging a Geometric forward policy in combination with a Categorical backward policy and integrating human-inspired reward models, our method efficiently discovers concise and high-quality solution trajectories in sparse-reward environments.

5.1 Summary and Implications

Our experiments demonstrate that:

- A Geometric distribution for the forward policy (P_F) induces a structural bias toward shorter, goal-directed trajectories, resulting in higher success rates compared to Categorical policies.
- Reward models inspired by human reasoning–such as discounting, cycle detection, and trajectory length regularization–substantially improve learning by guiding the model toward concise and interpretable solution paths.
- The combination of a Geometric P_F with a Categorical backward policy (P_B) balances directed exploration and flow correction, yielding not only high overall performance but also structurally diverse successful trajectories.

These findings suggest that carefully structured inductive biases—in both policy and reward design—can guide GFlowNet toward expert-aligned solution augmentation, highlighting its utility for complex reasoning tasks such as ARC.

5.2 Limitations and Future Work

While our proposed framework yields promising results, several limitations remain:

- **On-Policy Training Sensitivity:** The current approach relies on on-policy training, which is sensitive to the quality of initial trajectory samples. Early poor samples may cause high variance in learning. Future work will explore hybrid training schemes that combine the robustness of off-policy updates with the stability of on-policy guidance.
- **Restricted Task Scope:** This study focuses on ARC tasks involving full-grid transformations. Extending the framework to handle tasks that require localized sub-grid manipulations poses a significant challenge due to the exponentially growing action space. We plan to address this by leveraging ARCLE (Lee et al., 2024) and developing submodules capable of supporting fine-grained selection and transformation operations.
- **Evaluation of Downstream Utility:** Although our method generates diverse and valid solutions, we have not yet evaluated whether these augmented solutions meaningfully improve the performance of downstream models. Future research will empirically investigate whether training on these augmented solutions leads to measurable improvements in AI reasoning or generalization. In particular, we believe that exposing large language models (LLMs) to curated and diverse reasoning trajectories may help expand their solution space and improve their ability to generalize over structurally complex tasks.

5.3 Final Remarks

Our work demonstrates the potential of GFlowNet as a powerful mechanism for automated solution augmentation in reasoning-intensive tasks. By incorporating human-inspired reward structures and asymmetric policy design, we guide the model to generate concise, diverse, and high-quality solutions—an essential capability in sparse-reward environments.

Rather than functioning purely as a solver, GFlowNet here serves as an expert-aligned solution generator. This opens opportunities for integrating our approach into broader reasoning pipelines, where diverse solution candidates are required for further verification, ranking, or learning. We believe that future advances in exploration strategies, structural bias modeling, and downstream integration will further enhance the applicability of GFlowNet to System-2-level tasks, ultimately contributing to more general and interpretable artificial intelligence systems.

References

DeepSeek AI. Deepseek r1: Towards reliable and verifiable reasoning in large language models. `https://www.deepseek.com/research`, 2024. Accessed: 2025-04-19.

Emmanuel Bengio, Moksh Jain1, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow Network Based Generative Models for Non-Iterative Diverse Candidate Generation. In *NeurIPS*, 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. GFlowNet Foundations. *Journal of Machine Learning Research*, 2023.

Kiril Bikov, Mikel Bober-Irizar, and Soumya Banerjee. AugARC: Augmented Abstraction and Reasoning Benchmark for Large Language Models. *AAAI Workshop on Preparing Good Data for Generative AI: Challenges and Approaches*, 2025.

François Chollet. On the Measure of Intelligence. *arXiv:1911.01547*, 2019.

François Chollet. OpenAI o3 Breakthrough High Score on ARC-AGI-Pub , 2024. URL `https://arcprize.org/blog/oai-o3-pub-breakthrough`.

Karl Cobbe, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

Greg Kamradt. Announcing arc-agi-2 and arc prize 2025, 2025. URL `https://arcprize.org/blog/announcing-arc-agi-2-and-arc-prize-2025`. Accessed: 2025-04-19.

Michael Hodel. Addressing the Abstraction and Reasoning Corpus via Procedural Example Generation. *arXiv:2404.07353*, 2024.

Moksh Jain, Emmanuel Bengio, Alex-Hernandez Garcia, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ekbote, Jie Fu, Tianyu Zhang, Micheal Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological Sequence Design with GFlowNets. In *ICML*, 2022.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-Objective GFlowNets. In *ICML*, 2023.

Hyosoon Jang, Minsu Kim, and Sungsoo Ahn. Learning Energy Decompositions for Partial Inference in GFlowNets. In *ICLR*, 2024.

Yuxuan Jiang, Zihan Wang, Haozhuo Jiang, Jipeng Zhang, and Zhiyuan Liu. Learning Planning-based Reasoning by Trajectories Collection and Process Reward Synthesizing. *EMNLP*, 2024.

Sejin Kim, Hosung Lee, and Sundong Kim. Addressing and Visualizing Misalignments in Human Task-Solving Trajectories. *ICLR Workshop on Bidirectional Human-AI Alignment*, 2025.

Yunho Kim, Jaehyun Park, Heejun Kim, Sejin Kim, Byung-Jun Lee, and Sundong Kim. Diffusion-Based Offline RL for Improved Decision-Making in Augmented ARC Task. *arXiv:2410.11324*, 2024.

Hosung Lee, Sejin Kim, Seungpil Lee, Sanha Hwang, Jihwan Lee, Byung-Jun Lee, and Sundong Kim. ARCLE: The Abstract and Reasoning Corpus Learning Environment for Reinforcement Learning. In *CoLLAs*, 2024.

Seungpil Lee, Woochang Sim, Donghyeon Shin, Wongyu Seo, Jiwon Park, Seokki Lee, Sanha Hwang, Sejin Kim, and Sundong Kim. Reasoning Abilities of Large Language Models: In-Depth Analysis on the Abstraction and Reasoning Corpus. *ACM Transactions on Intelligent Systems and Technology*, 2025.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Tamas Gutman-Solo, Andrew Lampinen, Charles Chan, Surya Ganguli, Samuel S. Schoenholz, and Justin Gilmer. Solving quantitative reasoning problems with language models. *arXiv:2206.14858*, 2022.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step: Improving mathematical reasoning with process supervision. In *ICLR*, 2024.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 158–167, 2017.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Pairwise rm: Perform best-of-n sampling with knockout tournament. *arXiv preprint arXiv:2501.13007*, 2025a.

Yucheng Liu, Wengong Jin, Wenshuai Gao, Connor W. Coley, Regina Barzilay, and Yoshua Bengio. Retrogfn: One-step retrosynthesis with generative flow networks. *Information Sciences*, 676:119264, 2025b. ISSN 0020-0255. doi: 10.1016/j.ins.2024.119264. URL `https://www.sciencedirect.com/science/article/pii/S0020025525003263`.

Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. Mathgenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2732–2747, Bangkok, Thailand, 2024.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv:2406.06592*, 2024a.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024b.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory Balance: Improved Credit Assignment in GFlowNets. In *NeurIPS*, 2022.

OpenAI. Learning to reason with llms (openai o1). `https://openai.com/index/learning-to-reason-with-llms`, 2024.

OpenAI. openai-o3-mini. `https://openai.com/index/openai-o3-mini/`, 2025.

Ling Pan, Dinghuai Zhang, Aaron Courville, Longbo Huang, and Yoshua Bengio. Generative Augmented Flow Networks. In *ICLR*, 2023.

Jaehyun Park, Jaegyun Im, Sanha Hwang, Mintaek Lim, Sabina Ualibekova, Sejin Kim, and Sundong Kim. Unraveling the ARC Puzzle: Mimicking Human Solutions with Object-Centric Decision Transformer. In *ICML Workshop*, 2023.

Marcin Sendera, Minsu Kim, Sarthak Mittal, Pablo Lemos, Luca Scimeca, Jarrid Rector-Brooks, Alexandre Adam, Yoshua Bengio, and Nikolay Malkin. Improved off-policy training of diffusion samplers. *arXiv:2402.05098*, 2024.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 30706–30775, Honolulu, HI, 2023.

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. R-prm: Reasoning-driven process reward modeling. *arXiv:2503.21295*, 2025.

Max W. Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards Understanding and Improving GFlowNet Training. In *ICML*, 2023.

Suyeon Shim, Dohyun Ko, Hosung Lee, Seokki Lee, Doyoon Song, Sanha Hwang, Sejin Kim, and Sundong Kim. O2ARC 3.0: A Platform for Solving and Creating ARC Tasks. In *IJCAI Demo*, 2024. URL `https://o2arc.com`.

Noah Shinn, Beck Labash, Ashwin Gopinath, Ishita Dasgupta, Anca D. Dragan, and Trevor Darrell. Reflexion: Language agents with verbal reinforcement learning. *arXiv:2303.11366*, 2023.

Kashun Shum, Shizhe Diao, and Tong Zhang. Automatic prompt augmentation and selection with chain-of-thought from labeled data. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 12113–12139, Singapore, 2023.

Yuxuan Wan, Wenxuan Wang, Yiliu Yang, Youliang Yuan, Jen-tse Huang, Pinjia He, Wenxiang Jiao, and Michael Lyu. Logicasker: Evaluating and improving the logical reasoning ability of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2124–2155, Miami, Florida, USA, 2024.

Ke Wang and Zhendong Su. Automatic Generation of Raven's Progressive Matrices. In *IJCAI*, 2015.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*, 2022.

Chenxi Whitehouse, Monojit Choudhury, and Alham Fikri Aji. LLM-powered data augmentation for enhanced cross-lingual performance. *arXiv:2305.14288*, 2023.

Shunyu Yao, Dian Yu, Zhao Jeffrey, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *NeurIPS*, 2023.

Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. Flow of reasoning: Efficient training of llm policy with divergent thinking. *arXiv:2406.05673*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Self-taught reasoner – bootstrapping reasoning with reasoning. In *NeurIPS*, 2022.

David W. Zhang, Corrado Rainone, Markus Peschl, and Roberto Bondesan. Robust Scheduling with GFlowNets. In *ICLR*, 2023a.

Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. Generative Flow Networks for Discrete Probabilistic Modeling. In *ICML*, 2022.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron Courville, Yoshua Bengio, and Ling Pan. Let the Flows Tell: Solving Graph Combinatorial Optimization Problems with GFlowNets. In *NeurIPS*, 2023b.

Yufeng Zhang, Xuepeng Wang, Lingxiang Wu, and Jinqiao Wang. Enhancing Chain of Thought Prompting in Large Language Models via Reasoning Patterns. In *AAAI*, 2025a.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The Lessons of Developing Process Reward Models in Mathematical Reasoning. In *arxiv:2501.07301*, 2025b.

A Extended Background Details

A.1 GFlowNet Training Mechanism and Flow Matching

Flow Matching Condition. GFlowNets maintain a flow $F(s' \to s)$ for each directed edge (s', s) in a Directed Acyclic Graph (DAG), ensuring no probability mass is lost or gained within each state:

$$\sum_{s'} F(s' \to s) \;=\; \sum_{s''} F(s \to s''), \tag{10}$$

where $F(s' \to s)$ is the flow from state s' to s. This condition (Equation 10) ensures that each state's inflow equals its outflow. In practice, $F(s' \to s)$ is factored through a forward policy $P_F(s \mid s')$.

Trajectory Balance (TB) Loss. Building on Flow Matching, the TB loss (Malkin et al., 2022) ensures a global consistency between forward and backward paths:

$$Z \prod_{t=1}^{n} P_F(s_t \mid s_{t-1}) \;=\; R(x) \prod_{t=1}^{n} P_B(s_{t-1} \mid s_t), \tag{11}$$

where Z is a trainable constant and $R(x)$ the reward function for trajectory x. Minimizing:

$$\mathcal{L}_{\text{TB}}(\theta) \;=\; \Big[\log Z_\theta \;+\; \sum_{t=1}^{n} \log P_F(s_t \mid s_{t-1}; \theta) \;-\; \log R(x) \;-\; \sum_{t=1}^{n} \log P_B(s_{t-1} \mid s_t; \theta)\Big]^2 \tag{12}$$

aligns the probability of forward-sampled trajectories with their corresponding reward-proportional flows.

Training Steps. We outline the GFlowNet training procedure:

1. **Forward Sampling:** From initial state s_0, sample actions $a_t \sim P_F(\cdot \mid s_{t-1})$. Collect the resulting trajectory $x = (s_0, \ldots, s_n)$.
2. **Reward Computation:** Evaluate $R(x)$, e.g. 1 if it solves an ARC puzzle, else 0.
3. **Backward Sampling:** Use P_B to reconstruct or partially revisit states from s_n to s_0.
4. **TB Loss Computation:** Compute $\mathcal{L}_{\text{TB}}$ via Equation equation 12.
5. **Parameter Update:** Optimize θ to minimize $\mathcal{L}_{\text{TB}}$, adjusting both P_F and P_B accordingly.

By iterating these steps, GFlowNets learn to focus on high-reward trajectories while maintaining a diverse distribution of solutions.

A.2 Extended ARC Details

Although this work focuses on *whole-grid transformations*, many ARC puzzles require partial selection (e.g. coloring only a sub-region). This drastically increases the action space because selecting subsets of a 30×30 grid can be combinatorial. We note:

- **Dataset Complexity:** Some tasks have grid dimensions smaller than 30×30, but the upper limit still poses a challenge.
- **Potential Approaches for Partial-Selection:** We can adopt hierarchical policies that first *select* a region, then *transform* it. This approach can be integrated into GFlowNets by factoring the forward policy into multiple steps (Appendix of (Lee et al., 2024)).
- **Sparse Rewards and OOD Issues:** ARC test grids often deviate from training grids (different shapes, new patterns), demanding robust generalization. GFlowNets' ability to maintain multiple solutions is beneficial, but design of the reward function must handle rarely-seen corner cases.

Reasoning Dimensions in ARC. Following the framework of Lee et al. (2025), we consider three core cognitive dimensions of reasoning assessed by ARC:

- **Compositionality:** The ability to combine simple building blocks (e.g., functions or rules) into more complex transformations. ARC tasks often require models to compose multiple abstract operations.
- **Productivity:** The ability to extrapolate beyond observed examples and generate novel input-output mappings consistent with the inferred rules. ARC evaluates this by testing generalization to new input grids.
- **Logical Coherence:** The ability to maintain consistent internal logic throughout the reasoning process. This includes applying learned transformations in a rule-consistent manner across diverse contexts.

A.2.1 Formal Problem Definition for Whole-Grid ARC

Formally, each ARC task is a function f mapping $x \in \mathcal{X}$ to $y \in \mathcal{Y}$. We define a sequence of transformations $a_1, \ldots, a_T$ operating on the entire grid to produce y.

$$x^{(t+1)} = f\big(x^{(t)},\, a_t\big) \quad \text{with} \quad x^{(0)} = x,\;\; x^{(T)} = y.$$

We use a binary reward $R(x) = 1$ if $x^{(T)}$ matches the target output y, else 0. Further details on Markov Decision Process (MDP) formulations, action definitions, and potential expansions can be found in (Lee et al., 2024).

B Ablation Studies

The ablation studies focus on identifying which components of GFlowNet contribute most to its success in solving ARC problems. We evaluate the effects of reward scaling, different distribution combinations, and on-policy vs off-policy learning.

B.1 Effect of Reward Scale

In this study, we analyze the impact of varying reward scales on GFlowNet's performance. By adjusting the magnitude of rewards, we observe changes in learning efficiency, trajectory diversity, and reward distribution, aiming to identify the optimal reward scale for effective solution generation.

Experiment Setup As described in the hyperparameter settings, the base rewards were set at 15 for correct solutions, 0 for incorrect ones, and 10 for solutions ending with the *submit* action without reaching the maximum episode length. To examine the effects of different reward scales, we conducted experiments with the following settings:

- **Low Scale**: Reduced base rewards to 3 (by dividing the base reward by 5) for correct solutions and 2 for submit-ended trajectories.
- **Moderate Scale (Base)**: Maintained the default base rewards of 15 for submit-ended trajectories, 10 for trajectories that ended due to the maximum episode limit, and 0 for incorrect trajectories.
- **High Scale**: Increased base rewards to 225 for correct solutions and 150 for submit-ended trajectories (by multiplying the base reward by 15).
- **Very High Scale**: Significantly increased base rewards to 1500 for correct solutions and 1000 for submit-ended trajectories (by multiplying the base reward by 100).

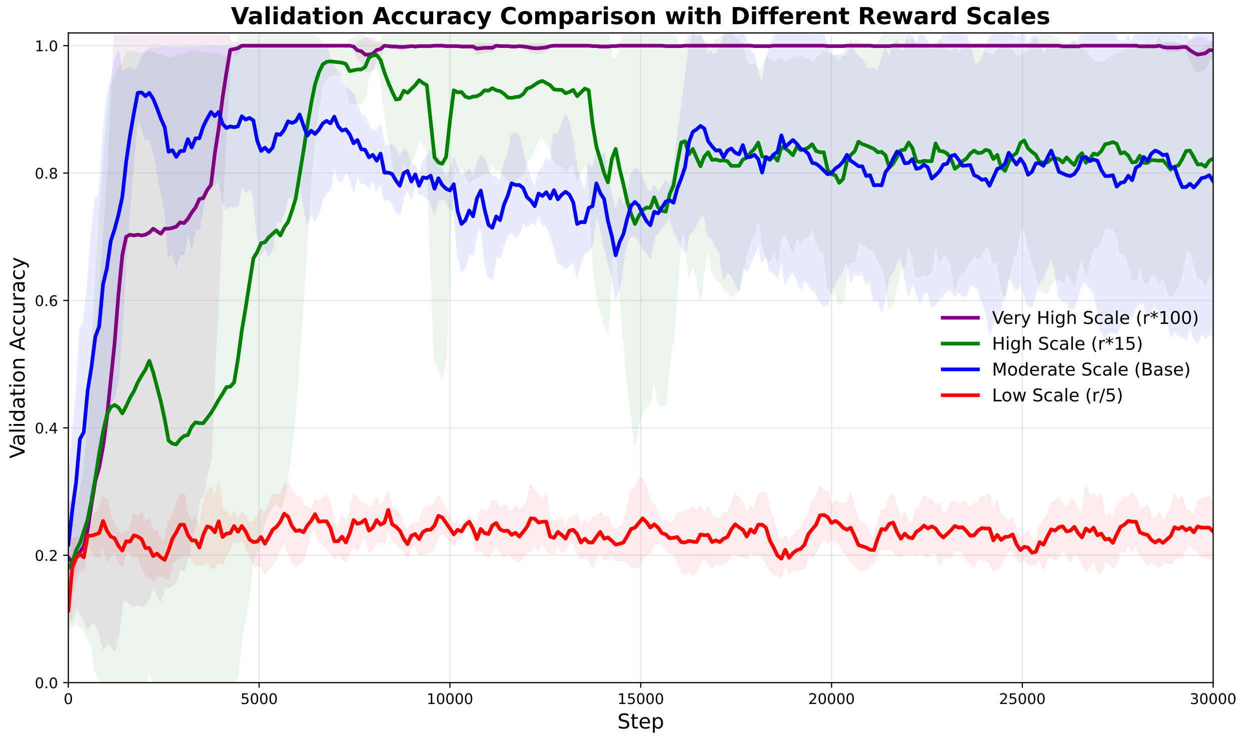


Figure 10: Performance comparison across different reward scales, showing the impact of reward values on validation accuracy over training steps.

Results and Analysis Tables 11 and 12 summarize the effects of different reward scales on success rates for unique and total trajectories. A moderate reward scale (Base) demonstrated a balanced improvement in solution diversity and success rate, achieving a 71.43% success rate for unique trajectories and 59.0% for total trajectories.

The effect of excessively high scaling (e.g., $100r$) led to a 100% success rate for both unique and total trajectories. However, it also significantly reduced solution diversity, producing only a single unique trajectory. This outcome suggests that while very high rewards lead to success, they may limit exploration. To confirm this effect, we calculated the diversity metric D_{reward}, which showed a reduction in reward distribution diversity at very high rewards, supporting the conclusion that excessive scaling discourages exploration.

Conversely, a lower reward scale ($r/5$) resulted in no successful trajectories, as indicated by a 0% success rate for both unique and total trajectories. This lack of success demonstrates that low rewards do not sufficiently reinforce correct solutions, causing the model to struggle to distinguish high- from low-quality trajectories.

Figure 10 illustrates the learning curves for each reward setting, showing that lower reward scales lead to noticeably poorer initial performance due to insufficient reinforcement. In contrast, moderate scaling enables broader exploration of the solution space, allowing the model to generalize effectively across different ARC tasks.

Furthermore, the diversity metrics, including trajectory diversity D_{traj} and reward distribution diversity D_{reward}, show that moderate reward scaling promotes greater exploration of successful trajectories. This finding suggests that a well-scaled reward signal enhances the model's ability to discover correct solutions while encouraging diverse trajectory exploration.

Conclusion These findings suggest that while moderate reward scaling enables GFlowNet to balance efficient learning and exploration, excessively high rewards yield diminishing returns in terms of diversity. Although high rewards increase task completion rates, they may reduce exploration and solution diversity.

Reward Scale	Unique Trajectories	Successful Unique Trajectories	Val_ACC (%)
Low Scale	100	0	0.00
Moderate Scale (Base)	56	40	71.43
High Scale	74	53	71.62
Very High Scale	1	1	100

Table 11: Success rate and reward distribution diversity based on unique trajectories across different reward scales.

Reward Scale	Total Trajectories	Successful Total Trajectories	Val_ACC (%)	D_reward
Low Scale	100	0	0.0	0.5139
Moderate Scale (Base)	100	59	59.0	0.8817
High Scale	100	75	75.0	0.6739
Very High Scale	100	100	100.0	0.0

Table 12: Success rate based on total trajectories across different reward scales.

Conversely, low reward scales do not provide sufficient reinforcement for effective learning. Optimal reward scaling is thus essential for supporting both accuracy and diversity, meeting the ARC task's requirements for generalization and high-quality solutions.

Statistical tests (e.g., chi-squared and Fisher's exact tests) confirmed a significant difference in success rates and diversity metrics between moderate and very high reward scales, reinforcing that balanced reward scaling is critical for effective exploration and efficient learning in complex tasks.

B.2 Action Number Performance Comparison

In this ablation study, we investigate the impact of varying the number of actions (3, 4, 5, and 10) on learning efficiency, exploration capacity, and trajectory diversity, further validating the insights obtained from the primary experiments.

Experiment Setup To solve ARC Task 178, which features a "Diagonal Flip" transformation, a minimum of three actions is required. While alternative solutions exist, the shortest solution involves three transformations: rotating the grid by 90 degrees, performing a horizontal flip, and executing the submit action. Starting with this minimal action set, we incrementally increased the number of actions to observe how each configuration affected validation accuracy and trajectory diversity over a maximum episode length of 10. This setup allowed us to explore the relationship between action granularity and the model's learning and exploration capabilities.

Results and Analysis As shown in Figure 11, models with three and five actions achieved high validation accuracy, with the three-action configuration reaching high performance relatively early in training. This indicates that a smaller action space allows more focused exploration, leading to efficient learning. Notably, the five-action configuration achieved near-optimal accuracy, outperforming both the four- and ten-action configurations in terms of learning efficiency. In contrast, the ten-action configuration, despite its expanded action space, failed to improve accuracy, suggesting that an excessive number of actions may impede effective exploration within the same training duration.

The reward trend graph further illustrates the differences across action configurations. The five-action setup shows stable and high rewards, indicating that this configuration balances action granularity with exploration capacity. The four-action configuration stabilizes at a moderate reward level after an initial exploration phase. In contrast, the ten-action configuration exhibits consistently low rewards, suggesting that GFlowNet struggles to optimize effectively in larger action spaces without additional guidance. This finding aligns with prior research, suggesting that overly large action or state spaces can dilute exploration, leading to suboptimal learning.

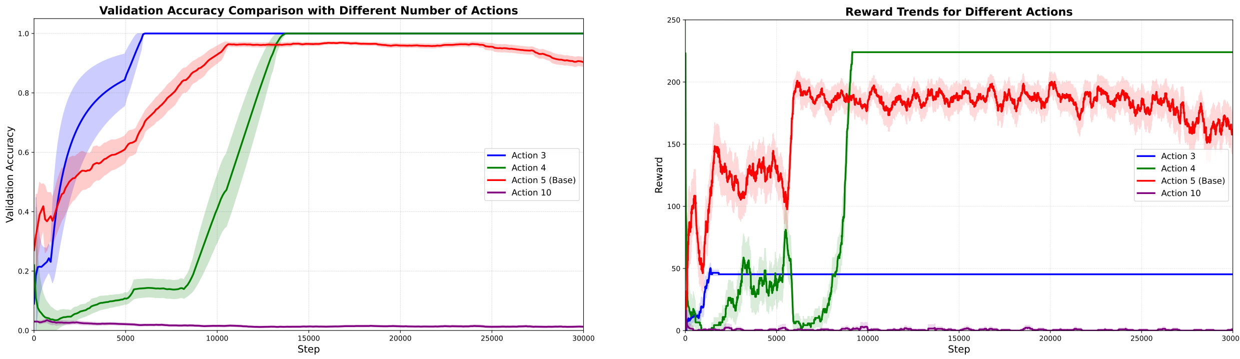


Figure 11: (Left) Performance comparison across number of actions, showing the impact of action count on validation accuracy over training steps. (Right) Reward comparison: comparison of reward values over training steps for each action configuration, highlighting differences in reward trends.

While expanding the action space increases trajectory diversity, it does not necessarily correlate with higher validation accuracy. For example, although the ten-action configuration showed greater trajectory diversity, it suffered from reduced learning efficiency and failed to achieve high rewards. This suggests that, while larger action spaces introduce more potential trajectories, they may also complicate exploration, especially without additional guidance.

The calculated *Reward Distribution Diversity* D_{reward} values provide further insight. As seen in Tables 13 and 14, D_{reward} for the three- and four-action configurations is zero, indicating uniform rewards due to limited trajectory diversity. In contrast, the ten-action configuration has a D_{reward} of 0.4439, and the five-action configuration yields 0.6739, reflecting greater reward diversity as the action space expands. Although increased diversity reflects broader exploration, it does not necessarily lead to improved performance, as shown by the ten-action configuration's low success rate and accuracy.

File	**Unique Trajectories**	**Successful Unique Trajectories**	**Val_ACC (%)**	**Total Trajectories**	**D_reward**
$n(a) = 3$	1	1	100.00	100	0.0
$n(a) = 4$	1	1	100.00	100	0.0
$n(a) = 5$ (Base)	41	26	63.41	100	0.6739
$n(a) = 10$	95	11	11.58	100	0.4439

Table 13: Summary of unique and successful trajectories, success rates, total trajectories, and reward distribution diversity for each action configuration.

File	**Successful Total Trajectories**	**Val_ACC (%)**	**D_reward**	**Comments**
$n(a) = 3$	100	100.0	0.0	Minimal exploration with consistent success
$n(a) = 4$	100	100.0	0.0	Limited diversity but high success rate
$n(a) = 5$ (Base)	81	81.0	0.6739	Balanced exploration and success
$n(a) = 10$	11	11.0	0.4439	High diversity, low success rate

Table 14: Total success rates, reward distribution diversity, and additional comments on trajectory diversity and performance for each action configuration.

The three-action and four-action configurations both achieved 100% success rates for unique trajectories, indicating that each generated trajectory successfully completed the task, though with limited diversity. Conversely, the ten-action configuration showed the highest number of unique trajectories (95) but had a low success rate (11.58%), suggesting that a large action space dilutes effective exploration, leading to lower task completion rates. The five-action configuration balanced exploration and success, yielding 41 unique trajectories with a success rate of 63.41%.

For the five- and ten-action configurations, statistical tests (chi-squared and Fisher's exact tests) confirmed a statistically significant difference in success rates, with the five-action configuration performing better among unique trajectories. This reinforces the idea that an optimal action space size is essential for effective learning.

These findings suggest that balancing action space size is critical for effective exploration and learning. Larger action spaces may require additional mechanisms, such as hybrid training approaches or managed replay buffers, to selectively sample high-quality data for focused exploration (Shen et al., 2023; Sendera et al., 2024). In this study, we implemented off-policy mechanisms to address this challenge. Future research could develop models capable of handling extensive search spaces more effectively, potentially by incorporating adaptive exploration techniques that adjust to the complexities of larger action spaces.

B.3 episode Length Performance Evaluation

In this ablation study, we examine how varying episode Lengths affect GFlowNet's performance in solving ARC tasks. By allowing the model a limited number of steps versus unlimited steps, we aim to understand the impact of step limitations on GFlowNet's ability to find high-quality solutions.

Experiment Setup The action space was fixed at 10 actions, and we varied the episode Length to observe how each setting influenced the model's performance. The episode Lengths tested were 4, 5, 10, 20, and 50. This setup allowed us to assess how different step limitations impact solution quality, exploration diversity, and learning efficiency.

Results and Analysis The results show a trend of increasing validation accuracy with longer episode lengths. For shorter episodes (4 and 5 steps), the model struggled to consistently find optimal solutions, likely due to limited exploration capacity. As the episode length increased to 10, 20, and 50 steps, validation accuracy improved significantly. This suggests that longer episodes allow GFlowNet to explore a broader solution space, increasing the chances of finding correct solutions.

The *Reward Distribution Diversity* D_{reward} values further illustrate the effect of episode length on exploration diversity. As shown in Tables 15 and 16, diversity values increase with episode length, indicating a broader exploration range. For example, Episode Length 4 has a D_{reward} of 0.32, while Episode Length 20 achieves the highest diversity with a D_{reward} of 8.62. This pattern suggests that allowing more steps enables the model to explore a wider variety of reward trajectories, enhancing trajectory robustness.

However, the results also indicate that excessive reward diversity, such as with Episode Length 50, does not necessarily lead to higher validation accuracy or success rates. The validation accuracy graph (Figure 12) shows that, while Episode Length 50 allows extensive exploration, it may introduce too much complexity, resulting in suboptimal learning outcomes.

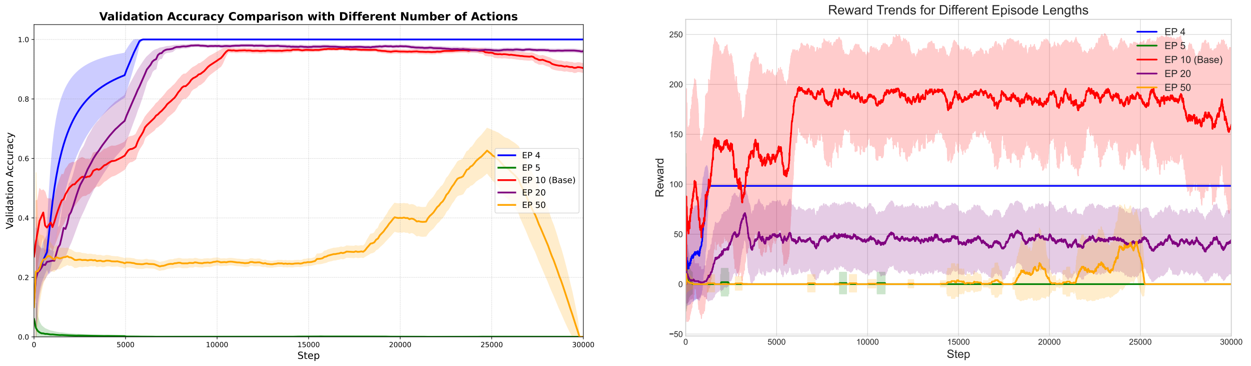


Figure 12: (Left) Performance comparison across episode length, showing the impact of action count on validation accuracy over training steps. (Right) Reward comparison: comparison of reward values over training steps, highlighting differences in reward trends.

Additional Insights Analysis of reward distribution diversity D_{reward} and success rates for unique and total trajectories across different episode lengths highlights distinct patterns:

1. **Reward Distribution Diversity:** D_{reward} represents variability in the reward values encountered, reflecting the extent of exploration. The highest diversity (2.14) is achieved with Episode Length 20, indicating broad exploration and high variability in rewards. Moderate diversity is observed for Episode Length 10 (0.88) and Episode Length 5 (0.20), while Episode Lengths 4 and 50 exhibit minimal diversity (0.0), suggesting limited exploration under these configurations.
2. **Unique and Total Trajectories:** Episode Length 4 and Episode Length 50 yield only 1 unique trajectory, with Episode Length 4 achieving complete success (100%) and Episode Length 50 showing no success (0%). This indicates that very short or very long episode lengths may hinder the model's exploration and success potential. Episode Length 20 exhibits a high success rate (95.95%) across 74 unique trajectories, suggesting an effective balance between exploration and solution quality. Episode Length 10 (Base) maintains moderate success (71.43%) with a diversity score of 0.88, reaffirming its utility as a stable baseline.
3. **Success Rate:** Both unique and total success rates are maximized with Episode Length 20, demonstrating that it supports optimal exploration for high-quality solutions. Low success rates in unique and total trajectories for Episode Length 50 (0.0%) indicate that overly long episodes may dilute effective exploration, preventing the model from reliably identifying successful solutions.

File	Unique Trajectories	Successful Unique Trajectories	Val_ACC (%)	D_reward
EP 4	1	1	100.00	0.0
EP 5	76	4	5.26	0.20
EP 10	56	40	71.43	0.88
EP 20	74	71	95.95	2.14
EP 50	1	0	0.00	0.0

Table 15: Trajectory success rates and reward distribution diversity for various episode lengths and reward settings.

File	Total Trajectories	Successful Total Trajectories	Val_ACC (%)
EP 4	100	100	100.0
EP 5	100	5	5.0
EP 10	100	59	59.0
EP 20	100	97	97.0
EP 50	100	0	0.0

Table 16: Total success rates for different episode lengths and reward configurations.

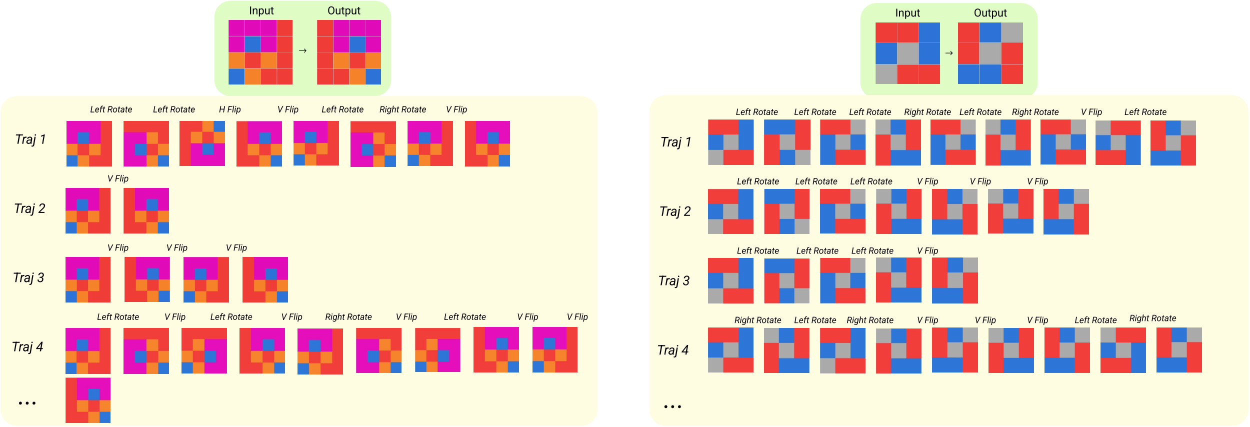
Conclusion These findings demonstrate that, while longer episode lengths generally allow GFlowNet to explore and converge on higher-quality solutions, an optimal range exists. Excessively long episodes increase reward diversity but can hinder learning efficiency by introducing unnecessary complexity. Balancing episode length with suitable exploration strategies is essential for maximizing GFlowNet's performance on ARC tasks. Future research could focus on adaptive episode lengths or exploration control mechanisms to fine-tune this balance, achieving more efficient learning in complex search spaces.

B.4 Generated Trajectories and application to other tasks

To further analyze GFlowNet's trajectory generation and evaluate its applicability to other tasks, we conducted task-specific experiments on selected ARC tasks. These experiments aimed to examine the diversity of generated trajectories, their success rates, and the consistency of reward-based learning across different task types.

Trajectory Visualization and Analysis Figure 13 illustrates multiple trajectories generated for two ARC tasks: Task 150 and Task 179. Each trajectory represents a unique sequence of transformations (e.g., rotations, flips) applied to the input grid to generate the correct output:

- **Task 150 (a):** The generated trajectories include efficient sequences (e.g., fewer steps) and redundant solutions with repeated actions. This demonstrates GFlowNet's exploration of diverse paths to the same output.
- **Task 179 (b):** While successful trajectories are generated, Task 179 highlights the tendency of some solutions to include unnecessary repetitive actions, suggesting room for further optimization.
- These examples showcase GFlowNet's ability to discover a range of action sequences, balancing diversity and correctness in trajectory generation.



(a) Task 150

(b) Task 179

Figure 13: Visualization of generated trajectories for two ARC tasks. Each trajectory represents a sequence of transformations applied to the input grid to achieve the correct output. Task 150 (a) and Task 179 (b) demonstrate diverse action sequences, including rotations and flips, leading to successful solutions.

Task-wise Results and Observations Table 17 summarizes the results of our task-specific analysis, including the number of unique trajectories, their success rates, and the reward diversity (entropy).

Task #	Unique Trajectories (Total)	Successful Unique (Total)	Success Rate (Unique, Total, %)	Reward Diversity (Entropy)
380	3 (300)	3 (300)	100.00 (100.00)	0.00
241	105 (300)	99 (280)	94.29 (93.33)	0.37
155	3 (300)	3 (300)	100.00 (100.00)	0.00
150	100 (300)	10 (30)	10.00 (10.00)	0.37
140	3 (300)	3 (300)	100.00 (100.00)	0.00
87	15 (300)	15 (300)	100.00 (100.00)	0.72

Table 17: Task-wise analysis of trajectories, success rates, and reward diversity. Unique values are shown with their corresponding total values in parentheses.

- **Unique Trajectories:** For tasks like 380, 155, and 140, the model consistently generated only a few successful trajectories, indicating that these tasks allow minimal variation in the solution space.

- **Success Rate:** Tasks such as 380, 155, and 87 achieved a 100% success rate, demonstrating that GFlowNet effectively explores correct solutions. However, for task 150, only 10% of unique trajectories were successful, suggesting a more complex solution space.
- **Reward Diversity**: Tasks with lower reward entropy (e.g., 380 and 140) reflect a solution space with limited diversity. In contrast, tasks like 87 exhibit higher entropy, indicating a greater variety of valid solutions.

B.5 Off-Policy Training Analysis

Experiment Setup We explored off-policy training using a replay buffer with various sampling methods:

- **Priority Sampling (PRT)**: Prioritizes high-reward experiences.
- **Epsilon-Greedy Sampling**: Randomly samples with probability ϵ and high-reward trajectories otherwise.
- **Fixed Ratio Sampling**: Samples a fixed mix (e.g., 8:2 or 9:1) of high- and low-reward trajectories.

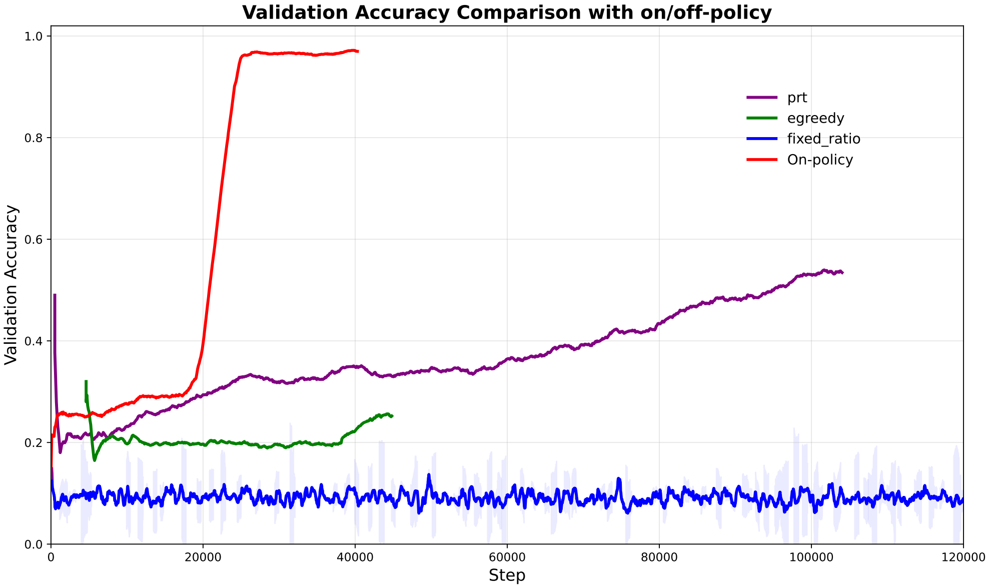


Figure 14: Validation accuracy comparison between on-policy and off-policy training methods. On-policy training converges much faster than off-policy sampling methods.

Results and Analysis Off-policy training with PRT sampling achieved lower variance across seeds but required significantly more steps (approximately 120,000) to converge, compared to on-policy training which reached near-perfect accuracy within 1,000 steps. This trade-off indicates that while off-policy methods enhance stability, their slower convergence makes on-policy training more practical for ARC problems.

(Additional Analysis) In particular, because ARC problems are highly sparse in rewards and often solved with short sequences, quickly stumbling upon a good trajectory is crucial. Off-policy learning, which replays a mix of older experiences, can excessively re-emphasize suboptimal early trajectories unless carefully managed. On-policy methods, by contrast, adapt more rapidly to newly discovered successful paths—leading to faster improvement.

C Algorithm

Algorithm 1 GFlowNet Architecture for ARC Task

```
Input: ARC environment env with input $x_k$, forward policy parameters $\theta_F$, backward policy parameters $\theta_B$,
       total flow $Z_\theta$, reward function $r(s)$, max episode length $T$, Replay buffer BUFFER (optional)
Output: Trajectory $\tau$
$s \leftarrow s_0 \leftarrow x_k$
foreach example do
    foreach training step do
        $\tau \leftarrow [s]$
        $\log p_F^{total} \leftarrow 0$
        $\log p_B^{total} \leftarrow 0$
        $t \leftarrow 0$
        while $t < T$ and $\neg$ env.is_done(s) do
            $t \leftarrow t + 1$
            $z \leftarrow \pi_F(s; \theta_F, \theta_B)$
            Split $z$ into $z_F$, $z_B$

            # Step 1: Forward Pass
            $p_F \leftarrow \text{Softmax}(z_F)$
            Sample $X_i \sim \text{Geometric}(p_{F_i})$ // $X_i = \{3, 14, 2, \dots, 7\}$
            $a^* \leftarrow \arg\min_i X_i$
            $\log p_F \leftarrow \log \text{GeomPMF}(X_{a^*})$
            $s' \leftarrow env.\text{step}(s, a^*)$
            $\tau \leftarrow \tau \cup \{s'\}$

            # Step 2: Backward Pass
            $p_B \leftarrow \text{Softmax}(z_B)$
            Sample $a' \sim \text{Categorical}(p_B)$
            $\log p_B \leftarrow \log p_B(a')$

            # Step 3: Reward
            $r(s') \leftarrow r_{final}$
            if cycle is detected then
                $r(s') \leftarrow r(s') - \lambda C(\tau)$
            end
            $\log p_F^{total} \leftarrow \log p_F^{total} + \log p_F$
            $\log p_B^{total} \leftarrow \log p_B^{total} + \log p_B$
            Store in BUFFER if off-policy (optional)
            $s \leftarrow s'$
        end
        $loss \leftarrow (\log Z_\theta + \log p_B^{total} - \log p_F^{total} - \log r(s))^2$
        Optimize $\theta_F$, $\theta_B$ to minimize $loss$
        Off-policy updates using BUFFER (optional)
        Update sampling model (optional)
    end
end
return $\tau$
```