C2-SeqRL: Reinforcement Learning for Conflict-Constrained Sequence Construction

Aoyu Pang, Chung Shue Chen, Maonan Wang, Man-On Pun, Yuan-Hsun Lo, Wing Shing Wong

C^2-SeqRL: Reinforcement Learning for Conflict-Constrained Sequence Construction

Aoyu Pang*†, Chung Shue Chen*, Maonan Wang†, Man-On Pun†, Yuan-Hsun Lo‡, Wing Shing Wong§
*The Chinese University of Hong Kong, Shenzhen, China
†Nokia Bell Labs, Paris-Saclay Center, 12 Rue Jean Bart, 91300 Massy, France
‡The Department of Applied Mathematics, National Pingtung University, Taiwan
§The Chinese University of Hong Kong (CUHK), Shatin, Hong Kong
Email: aoyupang@link.cuhk.edu.cn, chung_shue.chen@nokia-bell-labs.com, maonanwang@link.cuhk.edu.cn, simonpun@cuhk.edu.cn, yhlo0830@gmail.com, wswong@ie.cuhk.edu.hk

***Abstract*—Sequences underpin numerous applications across communication systems, scheduling, symbolic reasoning, and beyond. However, constructing sequences that satisfy complex structural constraints remains a persistent challenge. As sequence length and constraint complexity increase, the feasible solution space grows exponentially, rendering conventional search-based or rule-driven approaches computationally infeasible. To overcome these limitations, we introduce C^2-SeqRL, a deep reinforcement learning framework that formulates constrained sequence construction as an episodic decision-making problem. An agent progressively fills sequence positions while strictly adhering to conflict-avoidance constraints, enabling the discovery of latent structural patterns that characterize valid solutions. Furthermore, C^2-SeqRL leverages maskable reinforcement learning to proactively eliminate infeasible actions and integrates expert-guided policy learning to enhance solution quality and accelerate convergence. By jointly exploiting constraint awareness and informed exploration, C^2-SeqRL offers a scalable and adaptive paradigm for constrained combinatorial sequence generation, showing strong potential for real-world applications where structural validity is critical. Code is publicly available at C^2-SeqRL.**

***Index Terms*—Reinforcement Learning, Imitation Learning, Constrained Sequence Construction, Maskable Reinforcement Learning**

I. Introduction

Solving combinatorial optimization problems under strict constraints is a long-standing challenge to the intersection of discrete mathematics, computer science, and engineering [1]–[4]. These problems typically involve high-dimensional discrete search spaces, leading to exponential growth in the search complexity. Consequently, traditional search-based and rule-based approaches often become computationally infeasible.

To alleviate these limitations, reinforcement learning (RL) has emerged as a promising paradigm for exploring constrained combinatorial spaces and giving optimal or near-optimal solution [5]. Instead of exhaustively enumerating possible candidates, RL agents learn through interaction and feedback, enabling adaptive and guided exploration of high-dimensional search spaces while continuously improving decision quality. This data-driven search mechanism allows RL to discover high-quality feasible sequences or combinatorial solutions that are difficult to obtain using conventional algorithm or optimization techniques. While challenges in efficiency and convergence remain for these intrinsically NP-hard problems [6], RL provides a flexible and scalable approach for tackling constrained sequence generation.

Motivated by this, we focus on a challenging yet practically important subclass of constrained sequence construction problems, called *multiset color coding* [7]. Unlike conventional sequence codes where symbol order determines code identity or for user recognition purpose [2], multiset codes represent unordered collections that allow repeated elements and the order of the elements does not matter [8]. For instance, `AAB`, `ABA`, and `BAA` are all identical and correspond to the same multiset $\{$`A`,`A`,`B`$\}$, as they have the same number of `A`'s and `B`'s. That is, multiple sequences can map to the same multiset code and introduce complex combinatorial dependencies. When additional structural constraints such as fixed multiset cardinality (i.e., the number of individual objects the multiset contains) or Gray-code-style adjacency [9] are imposed, the solution search or algorithmic complexity largely grows, rendering enumeration or handcrafted strategies inefficient and often infeasible. We are motivated to develop scalable learning-based solutions capable of efficiently exploring multiset-structured solution spaces and solving hard combinatorial optimization problems.

To tackle the above challenge, we formulate the multiset-constrained sequence generation problem as a constrained RL problem, where an agent sequentially assigns symbols to positions in a candidate sequence under strict feasibility constraints that ensure the multiset uniqueness and correct symbol multiplicities. We propose a unified framework, termed C^2-SeqRL (Conflict-Constrained Sequence Reinforcement Learning), which integrates knowledge-guided dynamic masking with imitation learning (IL) [10]. The dynamic masking mechanism proactively suppresses infeasible actions to avoid invalid states during exploration, while IL incorporates expert demonstrations to guide policy initialization and improve sample efficiency. Together, these components enable more effective exploration and faster convergence, leading to robust perfor-

The research was partially funded by the National Science and Technology Council of Taiwan under Grants NSTC 114-2628-M-153-001-MY3.
(Corresponding authors: Chung Shue Chen; Man-On Pun)

mance on multiset-constrained sequence generation tasks.

The main contributions of this work are summarized below:

- We incorporate IL to provide expert guidance during RL training, enabling the agent to learn valid action patterns more efficiently and accelerating policy convergence in constrained sequence generation tasks for solving combinatorial optimization problem.
- By integrating maskable RL, the agent is restricted to selecting only constraint-satisfying actions, which substantially reduces unnecessary exploration in infeasible regions and enhances learning efficiency in complex combinatorial spaces.
- Extensive experiments demonstrate the superiority of our framework in both solution quality and computational efficiency compared to traditional rule-based and search-based baselines, indicating a promising method for hard combinatorial problem.

II. Preliminaries

A. Problem Formulation

a) Problem Setup: In this paper, we focus on a one-dimensional sequence of length L, where each position can take a value or symbol from a discrete set of size C. In many combinatorial design problem and application settings such as stream encryption [3] and object localization [2], [7], every contiguous subsequence of length m needs to satisfy a *uniqueness constraint* on its encoded representation, i.e., the encoding of each length-m subsequence should be distinct from all others in the sequence. Let $\mathbf{x}_{i:i+m-1}$ denote the subsequence starting at position i. Let $\mathcal{E}$ denote a predefined encoding table, where $\mathcal{E}(\mathbf{x}_{i:i+m-1})$ returns the integer code of the subsequence. The uniqueness constraint is expressible as:

$$0 \le i < j \le L - m, \quad \mathcal{E}(\mathbf{x}_{i:i+m-1}) \neq \mathcal{E}(\mathbf{x}_{j:j+m-1}), \quad (1)$$

where $\mathcal{R}_m$ denotes the set of valid encodings characterizing feasible subsequence patterns. This ensures both feasibility and global uniqueness of all length-m subsequences in the sequence. See illustrative example below. Consider a sequence of length $L = 6$, where each position takes one of $C = 3$ symbols from the set $\{0, 1, 2\}$, and we examine subsequences of length $m = 3$.

The sequence $(0, 0, 1, 1, 2, 2)$ is valid since all its length-3 subsequences have distinct multiplicity-based encodings:

- $(0, 0, 1)$: encoding $[2, 1, 0]$
- $(0, 1, 1)$: encoding $[1, 2, 0]$
- $(1, 1, 2)$: encoding $[0, 2, 1]$
- $(1, 2, 2)$: encoding $[0, 1, 2]$

Thus, no subsequence shares the same symbol-count representation, satisfying the uniqueness constraint.

b) Objective: Given an initial arbitrary sequence of length L, the objective is to iteratively update its elements until a solution is obtained, where a solution refers to a conflict-free sequence in which: (i) every length-m subsequence has a unique encoding, (ii) all positions or symbols in a sequence satisfy their domain constraints. In the above example, since $C = 3$, each position or symbol can be chosen from $\{0, 1, 2\}$, and (iii) the structural characteristics of the initial sequence are preserved as much as possible to maintain solution diversity and avoid excessive modifications to the original configuration.

B. MDP Formulation

To achieve this goal, we formulate the conflict-constrained sequence generation task as a Markov Decision Process (MDP). The state s_t represents the current sequence $\mathbf{x}^{(t)}$ along with its associated conflict information. An action (i, c) modifies the sequence by assigning symbol c to position i.

The transition dynamics are deterministic:

$$\mathbf{x}^{(t+1)} = U(\mathbf{x}^{(t)}, i_t, c_t). \quad (2)$$

The reward function promotes efficient conflict resolution and provides a positive terminal reward once the sequence becomes conflict-free. An episode terminates when all conflicts are eliminated or when a maximum step limit is reached.

This MDP formulation is to enable the agent to progressively correct local conflicts through reinforcement learning and ultimately produce a valid sequence that satisfies the combinatorial uniqueness constraints or requirements.

III. Method

As illustrated in Fig. 1, the proposed framework tackles conflict-constrained sequence generation problem through a RL formulation. The environment first receives an initial sequence as input and performs preprocessing operations including subsequence encoding and conflict detection. During each interaction step, the agent selects a position and assigns a feasible value, upon which the environment updates the sequence and re-evaluates the conflict conditions. This iterative process continues until all conflicts are eliminated, and the environment outputs a conflict-free sequence as the final solution. To ensure that exploration remains within the feasible search space, maskable RL is employed to dynamically suppress infeasible actions that would violate subsequence uniqueness constraints. Furthermore, to accelerate convergence and enhance policy quality in the early training stage, IL is incorporated to provide expert demonstrations as guidance. By leveraging the complementary strengths of maskable RL and IL, our framework enables efficient construction of globally valid solutions while strictly maintaining feasibility throughout the entire learning process.

A. RL Design

1) Observation Space: At each time step t, the agent receives an observation s_t that explicitly encodes the current sequence, modification history, conflict distribution, and subsequence-level structural information. Let the sequence be $\mathbf{x} = (x_1, \ldots, x_L)$, where $x_i \in \{0, \ldots, C-1\}$, and m denote the subsequence length used for conflict evaluation.

The observation is represented as a real-valued matrix:

$$s_t \in \mathbb{R}^{4 \times L}, \quad (3)$$

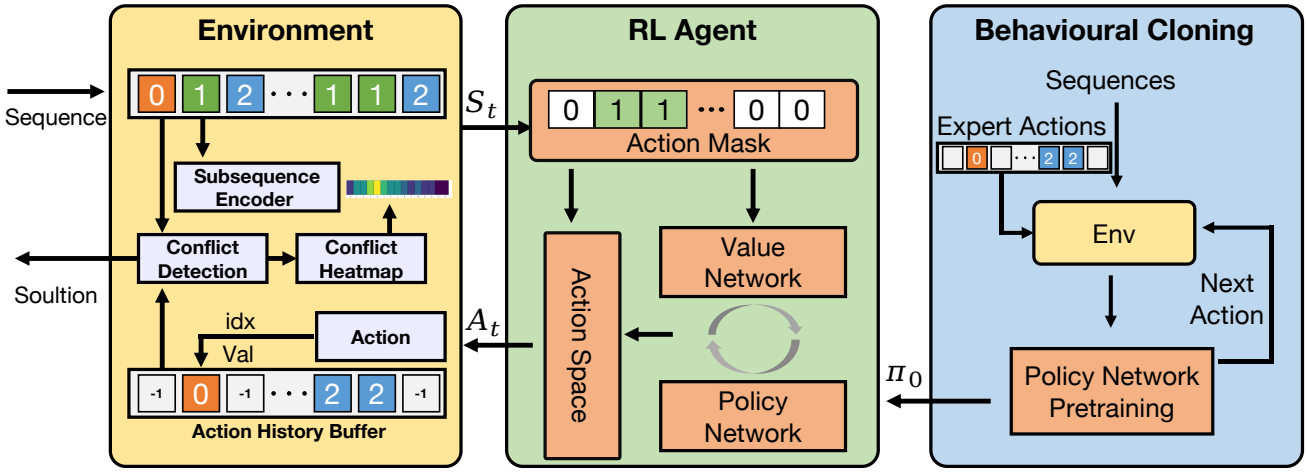

Fig. 1. Overall framework of the proposed RL approach integrating IL and maskable RL for valid sequence generation under conflict constraints. The agent sequentially assigns values to positions, while maskable RL reduces the search space by restricting the agent to feasible actions, and IL leverages expert demonstrations to accelerate policy convergence.

where each row corresponds to a specific type of information. There are four in total, corresponding to current sequence, modification record, conflict intensity, and subsequence encoding, respectively, described below.

a) Current sequence: The first row stores the current symbol assigned at each position:

$$s_t[0, i] = x_i, \quad i = 1, \ldots, L, \tag{4}$$

represented as floating-point values for neural network processing.

b) Modification record: The second row indicates the edit status of each position:

$$s_t[1, i] = \begin{cases} -1, & \text{if position } i \text{ has not been modified,} \\ x_i^{(t)}, & \text{if position } i \text{ has been updated.} \end{cases} \tag{5}$$

A value of -1 explicitly marks unchanged, while a valid non-negative value $x_i^{(t)}$ reflects the latest modification performed at step t. This representation allows the agent to distinguish between original and altered elements and to track the evolution of the editing process.

c) Conflict intensity: Let $\mathcal{C}_t$ denote the set of conflicting window pairs at step t. For two windows (i.e., subsequences):

$$w_{i_1} = (i_1,\, i_1 + 1,\, \ldots,\, i_1 + m - 1), \tag{6}$$

$$w_{i_2} = (i_2,\, i_2 + 1,\, \ldots,\, i_2 + m - 1). \tag{7}$$

A conflict is defined as equality of their encoded representations, i.e.,

$$(w_{i_1}, w_{i_2}) \in \mathcal{C}_t \iff \mathcal{E}(\mathbf{x}_{w_{i_1}}) = \mathcal{E}(\mathbf{x}_{w_{i_2}}), \tag{8}$$

indicating that these two windows violate the uniqueness constraint.

The third row records how many conflicting windows involve each position:

$$s_t[2, j] = \left| \left\{ (w_{i_1}, w_{i_2}) \in \mathcal{C}_t \;\middle|\; j \in w_{i_1} \cup w_{i_2} \right\} \right|. \tag{9}$$

Overlapping conflicts would produce larger values, reflecting higher correction significance.

d) Subsequence encoding: For each window $w_i = (i, \ldots, i + m - 1)$, we compute a histogram:

$$h_i = (n_0, \ldots, n_{C-1}), \quad n_c = |\{j \in w_i \mid x_j = c\}|. \tag{10}$$

Using the encoding table $\mathcal{E}$, Through the encoding table $\mathcal{E}$ below, each histogram is mapped to an integer code:

$$h_i \to \mathcal{E}(h_i) \in \{0, \ldots, |\mathcal{E}| - 1\}. \tag{11}$$

We assign the above code to the subsequence's start index such that:

$$s_t[3, i] = \begin{cases} \mathcal{E}(h_i), & \text{if } i \le L - m + 1, \\ -1, & \text{otherwise.} \end{cases} \tag{12}$$

Thus, for non-cyclic sequences, only the first $L - m + 1$ positions contain valid codes. For cyclic extensions, all L windows may be encoded by indexing modulo L.

This representation compactly integrates both global conflict distribution and local structural patterns, forming an informative state description for RL.

2) Action Space: At each step, the agent selects a single-position edit represented as an action (i, c), where $i \in \{1, \ldots, L\}$ denotes the index to be modified while $c \in \{0, \ldots, C - 1\}$ denotes the new value to assign, formally expressed as:

$$\mathcal{A} = \{(i, c)\}. \tag{13}$$

For discrete-action implementation, a bijection encodes actions into a scalar:

$$a = (i - 1)C + c, \quad i = \lfloor a/C \rfloor + 1, \quad c = a \bmod C. \tag{14}$$

Given the current sequence $\mathbf{x}^{(t)}$, applying (i_t, c_t) yields a deterministic update:

$$x_j^{(t+1)} = \begin{cases} c_t, & j = i_t, \\ x_j^{(t)}, & j \ne i_t. \end{cases} \tag{15}$$

To maintain feasibility, a binary mask $M_t \in \{0,1\}^{L\times C}$ restricts admissible actions to $\mathcal{A}_t^{\text{feas}} = \{a : M_t[a] = 1\}$. Two criteria guide mask construction: (i) edits that introduce new violations of subsequence uniqueness are disallowed, and (ii) only positions associated with existing conflicts are editable, focusing the agent on necessary refinements. This masked action space ensures constraint-compliant and conflict-resolving behavior during exploration.

Additionally, we maintain an action history

$$\mathcal{H}_t = \{(i_0, c_0), \ldots, (i_{t-1}, c_{t-1})\}, \tag{16}$$

which deterministically reconstructs the current sequence $\mathbf{x}^{(t)}$ from the initial sequence $\mathbf{x}^{(0)}$. Thus, the agent's decision trajectory is explicitly recorded throughout the editing process.

3) Reward Design: The reward function is constructed to simultaneously encourage rapid convergence and effective conflict resolution. At each timestep t, the reward is defined as:

$$r_t = -\alpha\, t - \beta\, d_t + \gamma\, \phi(\text{valid}(x^{(t)})), \tag{17}$$

where t denotes the current step count, d_t is the number of subsequences within the current partial sequence that violate the conflict constraint, and $\phi(\text{valid}(x^{(t)}))$ is an indicator function that returns 1 when the sequence is conflict-free and 0 otherwise. The coefficients α, β, and γ serve to balance the trade-offs among action efficiency, conflict reduction, and final solution validity.

This reward formulation jointly encourages efficient editing, conflict reduction, and successful completion. The step penalty limits unnecessary modifications, the conflict penalty drives progressive feasibility improvement, and the terminal reward enforces convergence to a conflict-free solution. Together, these components guide the agent toward generating valid sequences with minimal edits.

B. Imitation Learning

To accelerate policy learning and guide the agent toward high-quality solutions, we employ IL via Behavior Cloning (BC) pretraining [11]. Since there is no analytical algorithm to construct valid sequences, expert trajectories are generated through brute-force search. For each initial sequence containing conflicts, multiple bits are iteratively modified until all combinatorial constraints are satisfied. The minimal set of changes required to transform the sequence into a valid solution is recorded as an expert action set. This ensures that the agent learns to follow the shortest path toward feasibility during BC pretraining.

Although a valid transformation may involve multiple bit changes, the agent executes actions at the granularity of a single bit assignment. Therefore, each action set is decomposed into a sequence of single-bit actions, which are fed sequentially to the policy network. The agent observes the current state and predicts one action at a time until all actions in the expert set are executed and the sequence becomes a valid solution. During BC pretraining, the policy network is trained to imitate these expert actions by minimizing the cross-entropy loss between predicted and expert actions. This process enables efficient navigation of the combinatorial search space by avoiding infeasible actions in the early stage, thus offering higher-quality experiences for subsequent RL.

After BC pretraining, the learned policy weights π_0 are loaded into the RL agent, which continues training with reward feedback to potentially surpass the expert policy. By combining brute-force-generated expert trajectories, action decomposition, and supervised pretraining, this IL component significantly improves the efficiency, convergence speed, and solution quality of the agent in constrained sequence generation tasks.

C. Maskable Reinforcement Learning

To realize maskable RL in our framework, we instantiate the policy optimization component using Maskable Proximal Policy Optimization (MaskPPO) [12]. At each timestep, a maskable mechanism preserves only constraint-satisfying actions, thereby avoiding ineffective exploration and enhancing learning efficiency.

In our implementation, masking is performed by identifying all conflicting subsequences in the current sequence and restricting modifications to positions covered by these conflicts. Specifically, for each detected conflict, only elements within the conflicting windows are allowed to be modified, while all other positions are temporarily masked and excluded from action selection. This "conflict-cover masking" ensures that the agent focuses on correcting problematic regions without wasting exploration on positions that are already valid.

Extensive experiments demonstrate that this approach is highly effective: for over 99.8% of initial sequences, modifying only the conflict-covered positions is sufficient to recover and become fully valid sequences. By leveraging this targeted masking strategy, MaskPPO significantly improves learning efficiency and success rate, enabling the agent to substantially reduce the number of exploration steps and more efficiently generate valid sequences in large combinatorial spaces.

IV. Experimental Results

A. Experimental Setup

We evaluate the proposed framework on a multiset-constrained sequence generation task. In our experiments, the parameters are set as follows: mask length $m = 5$, the number of value choices per position $C = 3$, and sequence length $L = 15$.

By C^2-SeqRL, we find that there are in total 120 feasible solutions. while the sequence space has a size given by $C^L = 3^{15}$. It is worth noting that we also conduct exhaustive computer search in parallel to find the feasible solutions for the ground truth. Result shows that there are 120 solutions exactly. C^2-SeqRL finds the same.

B. Compared Methods

To evaluate the effectiveness of our proposed framework, we consider several representative baseline and variant methods for constrained combinatorial sequence generation:

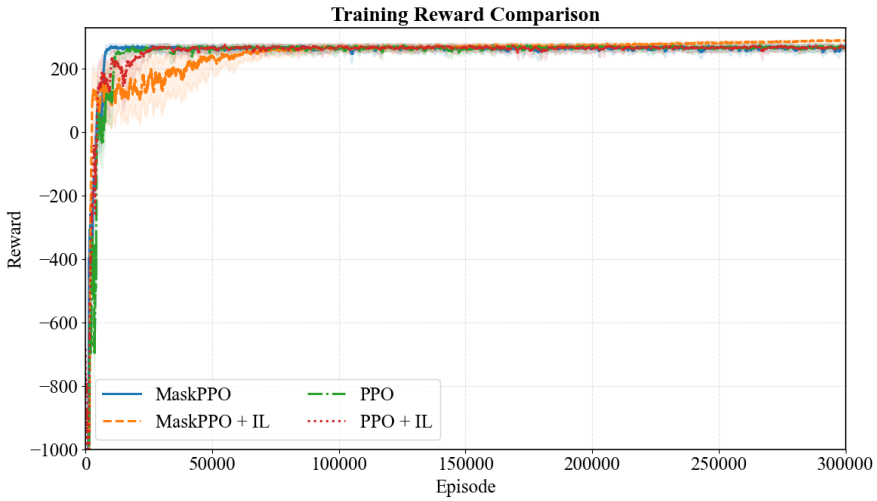

Fig. 2. Training reward curves for different RL-based sequence generation methods. MaskPPO+IL discovers feasible sequences earlier and achieves competitive final performance compared to other baselines.

- Full Enumeration (FE): An exhaustive search baseline that enumerates all possible actions until a valid sequence is obtained.
- Rule-Based: A heuristic approach that iteratively selects actions only from currently feasible (unmasked) positions.
- PPO: Standard Proximal Policy Optimization applied without any explicit constraint enforcement.
- PPO + IL: PPO pretrained with IL using expert demonstration sequences.
- MaskPPO: PPO augmented with dynamic masking to filter out infeasible actions at each decision step.
- MaskPPO + IL: The proposed approach, combining MaskPPO with IL to leverage expert guidance while strictly enforcing feasibility constraints.

C. Evaluation Metrics

We evaluate the performance of different sequence generation methods using the following metrics:

- **Success Rate:** This metric measures the proportion of generated sequences that satisfy all combinatorial constraints. For rule-based algorithms, it reflects whether the method can successfully construct a valid sequence. For RL-based algorithms, to prevent potential infinite loops when the agent fails to find a valid solution, we impose a maximum step limit for exploration. In our experiments, this limit is set to 100 steps. The success rate is then computed as the fraction of sequences successfully generated within this step limit.
- **Explored Actions:** The average number of actions sampled before a valid sequence is obtained. This metric captures the efficiency of each method in navigating the combinatorial space. A smaller number indicates more efficient exploration and faster convergence toward feasible sequences.

D. Parameter Settings

Table I summarizes the main hyperparameters used for BC pretraining and MaskPPO training. The table is split into two parts: the top part corresponds to BC, including the policy network structure, and the bottom part corresponds to MaskPPO. During evaluation, the maximum number of action attempts for generating a valid sequence is limited to 100 to ensure consistent and efficient testing.

TABLE I
TRAINING HYPERPARAMETERS FOR BC PRETRAINING AND MASKABLE PPO

Behavior Cloning Pretraining	
Policy Network	$4\times15 \rightarrow 512 \rightarrow 512 \rightarrow 256 \rightarrow$ num_actions
Observation Shape	(4, 15)
Number of Actions	15×3
Learning Rate	1×10^{-4}
Batch Size	1
Epochs	5000
Maskable PPO Training	
Learning Rate	5×10^{-4}
Batch Size	128
N Epochs	20
Steps per Update	2048
Discount Factor γ	0.99
Max Training Steps	30M

E. Results and Analysis

Table II presents the performance comparison among all evaluated methods. Several key observations can be drawn.

Standard PPO demonstrates stable training behavior but struggles to efficiently explore the large and discrete combinatorial action space, leading to only moderate success rates. In comparison, MaskPPO improves feasibility by explicitly filtering invalid actions, thus achieving better success rates;

TABLE II
PERFORMANCE COMPARISON OF SEQUENCE GENERATION METHODS ($m = 5$, $C = 3$, $L = 15$).

Method	Success Rate (%)	Explored Actions (avg. steps)
FE	100.0	10416.7
Rule-Based (Mask)	99.77	7541.63
PPO	98.88	36.36
PPO + IL	98.84	34.24
MaskPPO	99.20	34.86
MaskPPO + IL	**99.93**	**7.12**

however, it can still suffer from premature convergence to suboptimal solutions.

Introducing IL (PPO+IL) substantially enhances the early learning phase. Expert demonstrations provide high-quality prior knowledge, enabling the agent to more quickly acquire feasible editing strategies and achieve higher success rates with fewer modifications, highlighting the advantage of structurally guided exploration.

The combination of MaskPPO and IL (MaskPPO+IL) achieves the best performance across all metrics. By simultaneously incorporating feasibility constraints and expert guidance, the agent attains a near-perfect success rate and significantly fewer required steps, exhibiting more efficient and robust convergence behavior.

Fig. 2 shows the training reward curves of different methods. Although PPO and MaskPPO converge in fewer training iterations, their purely self-exploratory strategies are prone to local suboptimal solutions. In contrast, the integration of IL provides expert priors and more guided exploration, effectively avoiding premature exploitation and enabling more stable performance improvement. Consequently, MaskPPO+IL achieves higher-quality convergence and superior final performance compared to all other methods.

F. Discussion

Although the experimental scale is relatively moderate ($L = 15$, $m = 5$, $C = 3$), the feasible region remains extremely sparse, making efficient search and feasibility preservation highly challenging. Meanwhile, this setting is still representative for evaluating scalability in constrained combinatorial spaces.Experimental results show that dynamic masking alone may introduce early-stage instability, as each local modification reshapes the feasible action space. By incorporating IL, the agent acquires valid structural patterns more quickly, leading to more stable training and fewer ineffective edits. The synergy of constraint-aware masking and expert guidance enables efficient conflict resolution and significantly improves convergence efficiency. This approach also shows strong potential for high-dimensional discrete optimization, while more advanced masking or hierarchical policies may be required for problems with non-local structural constraints. Overall, these findings highlight the importance of guided exploration in RL for combinatorial sequence generation, improving both training efficiency and solution quality.

V. CONCLUSION AND FUTURE WORK

This paper presents a RL framework for constrained combinatorial sequence construction, integrating dynamic action masking with imitation learning. By combining expert-guided priors and constraint-aware exploration, the proposed method achieves efficient generation of conflict-free sequences in the large search space, yielding superior performance in both the solution optimality and sample efficiency. Although the experiments are conducted on moderate problem size ($L = 15$), the underlying search space is not small (equal to 3^{15}) and can reflect the effectiveness of the proposed framework in tackling challenging combinatorial optimization problems.

As future work, we plan to extend the approach to construct sequences of large L and also large symbol set size. It is also suitable to consider other sequence structure requirements and to consider two-dimensional codeword construction problem. In addition, incorporating multi-position editing actions and hierarchical policy architectures may improve convergence speed and enhance the ability to handle non-local structural constraints. Curriculum learning and symbolic or heuristic constraint reasoning may also offer promising techniques to generalize the framework and improve performance robustness of of constraint-aware RL for various applications in solving real-world combinatorial optimization problems.

ACKNOWLEDGMENT

We would like to thank Cedric Adjih, Pierre Escamilla, Elie De Panafieu and many colleagues for their support and help.

REFERENCES

[1] A. Nagda, P. Raghavan, and A. Thakurta, "Reinforced generation of combinatorial structures: Applications to complexity theory," *arXiv Preprint arXiv:2509.18057*, 2025.

[2] F. W. Sinden, "Sliding window codes," *AT&T Bell Laboratories Technical Memorandum*, pp. 1–19, 1985.

[3] T. Etzion and A. Lempel, "Algorithms for the generation of full-length shift-register sequences," *IEEE Transactions on Information Theory*, vol. 30, no. 3, pp. 480–484, 1984.

[4] C. S. Chen, P. Keevash, S. Kennedy, E. de Panafieu, and A. Vetta, "Robot positioning using torus packing for multisets," in *51st International Colloquium on Automata, Languages, and Programming (ICALP)*, 2024.

[5] S. Habib, A. Beemer, and J. Kliewer, "Learning to decode: Reinforcement learning for decoding of sparse graph-based channel codes," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[6] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Computers & Operations Research*, vol. 134, p. 105400, 2021.

[7] C. S. Chen, Y.-H. Lo, W. S. Wong, and Y. Zhang, "Object tracking using multiset color coding," in *International Symposium on Information Theory and Its Applications (ISITA)*, 2024.

[8] W. D. Blizard, "Multiset theory," *Notre Dame Journal of Formal Logic*, vol. 30, no. 1, pp. 36–66, 1989.

[9] C. S. Chen, W. S. Wong, Y.-H. Lo, and T.-L. Wong, "Multiset combinatorial gray codes with application to proximity sensor networks," 2025. [Online]. Available: https://arxiv.org/abs/2410.15428

[10] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: Algorithms, recent developments, and challenges," *IEEE Transactions on Cybernetics*, 2024.

[11] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich, "Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments," 2020. [Online]. Available: https://arxiv.org/abs/1910.04281

[12] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv Preprint arXiv:2006.14171*, 2020.