REINFORCEMENT LEARNING ASSISTED DECODING

Milad Taghipour, Asit Kumar Pradhan, Bane Vasić
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ, 85719
{miladt, asitpradhan, vasic}@arizona.edu

Faculty Advisor: Bane Vasic

ABSTRACT

This paper explores the application of reinforcement learning techniques in the context of the performance improvement of bit-flipping based decoders. We begin with a concise overview of bit-flipping based decoders and reinforcement learning algorithms. We then outline the methodology involved in mapping these iterative decoders into Markov Decision Processes and propose a method to decrease the number of states to make the Q-learning algorithm feasible for low-rate and long-length codes. This enables us to obtain an optimal decision rule and improve the decoding performance through the utilization of reinforcement learning algorithms. Subsequently, we conduct an analysis of the reinforcement aided bit-flipping based decoder and investigate a number of potential optimal solutions achievable through reinforcement learning algorithm. We provide a comparative examination of efficiency and complexity trade-offs between data-driven algorithms and traditional methods across the Binary Symmetric Channel and Additive White Gaussian Noise Channel.

INTRODUCTION

Error correcting codes (ECC) are essential for ensuring reliable communication across noisy channels. The design and implementation of low-complexity decoders for error correcting codes are of significant interest. However, low-complexity decoders, such as message-passing based algorithms, struggle to achieve low error rates for dense codes and codes with short girths. Therefore, the primary goal is to find a balance between complexity and performance in the decoding of error correction codes. The problem of decoding of linear block codes can be viewed as a supervised learning task, where the goal is to learn the mapping induced by a linear block code in the code space. This problem can be understood as learning the structure of the code space through input words and their associated codewords. One approach is to use parameterized functions, such as neural networks, to learn good parameters for the network to achieve their prespecified goal (e.g., minimizing the bit error rate). Without considering structures on the code, this approach works only for short block length codes [1]. The problem of learning decision regions is simplified for linear block codes due to the symmetry of the code, as it is sufficient to learn the decision region around just one codeword. In [2], the authors parameterized and optimized the message-passing algorithm. Additionally, the syndrome-based decoding of linear block codes is investigated in [3].

Another approach is to view the problem of decoding as a decision-making process, with the goal of learning optimal decisions. In [4], the authors propose a decoding approach that sequentially employs individual bit-flipping decisions rather than directly mapping observations to estimated codewords via supervised learning. This methodology is structured as a Markov decision process (MDP), utilizing reinforcement learning (RL) algorithms to refine decision strategies. Their method is based on the syndrome-based approach, where the MDP's state space contains all possible binary syndromes. In [5], the authors explores the scheduling problem in the belief propagation algorithm, framing it as a Markov decision process (MDP). Using reinforcement learning (RL), they optimize check node (CN) scheduling policies to improve sequential decoding performance compared to traditional flooding scheduling methods. These methods are useful for short-length and high-rate codes. We propose a modification to the MDP of the linear block codes to make it applicable to a wider range of codes.

CHANNEL CODING BACKGROUND

Let us assume $\mathcal{C}$ to be a binary linear code with parameters (n, k, d_{min}), where n is the code length, k is the code dimension, and d_{min} is the minimum distance of the code. We define a binary linear code $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_2^n \mid \mathbf{c}H^T = 0\}$, where H is the parity-check matrix, which is an $m \times n$ matrix where $m/n \geq 1 - R$, $R = k/n$ is the code rate. The error correction capability of a code can be defined on the basis of its minimum distance as $t = \lfloor \frac{d_{\text{min}} - 1}{2} \rfloor$. A codeword $\mathbf{c}$ is transmitted through the Binary Symmetric Channel (BSC) as $\mathbf{y} = (1-2\mathbf{c}) + \mathbf{z}$, where $\mathbf{y}$ is the received row vector of length n, and $\mathbf{z}$ is a row vector of length n generated by i.i.d. Bernoulli random variable with parameter ρ which is the cross-over probability of the channel. Similarly, a codeword can be transmitted through the Additive White Gaussian Noise (AWGN) channel defined as $\mathbf{y}' = (1 - 2\mathbf{c}) + \mathbf{n}$ where $\mathbf{n}$ is an n-length row vector generated by i.i.d. zero mean Gaussian random variable with variance $(2R \cdot \text{SNR})^{-1}$ where SNR is the signal-to-noise ratio. In addition, there are correspondences $\mathbf{y} = \text{sign}(\mathbf{y}')$ between the BSC and AWGN channel.

A. *Bit-Flipping Decoder*

The bit-flipping algorithm is a fast, simple yet effective algorithm in the decoding of error correcting codes such as Low Density Parity Check (LDPC) codes. The idea is to flip bits that are the least reliable to maximize the number of satisfied checks [6]. The Bit-Flipping algorithm is provided in Algorithm 1, where $\mathbf{s}$ is a syndrome row vector of length m, $\mathbf{q} = (q_1, \cdots, q_n)$ is an n-length row vector representing the number of unsatisfied checks connected to each variable, $\mathcal{B}$ is the set of candidates for flipping based on the majority update rule, and $\oplus$ denotes modulo-2 addition.

Markov Decision Process

Markov Decision Processes (MDPs) provides a mathematical framework for modeling decision-making problems where outcomes are random or deterministic under the control of a decision maker. An MDP can be defined based on (S, A, P, R, γ), where S is the set of finite states repre-

Algorithm 1 Bit-Flipping Decoder

Initialization: $\mathbf{x} = \text{sign}(\mathbf{y})$, H is the parity-check matrix, and set maximum iteration number
Output: estimated codeword $\hat{c}$

$\hat{\mathbf{c}} = \mathbf{x}$
$\mathbf{s} = \mod(\hat{\mathbf{c}}H^T, 2)$ ▷ Syndrome Computation
while $\mathbf{s} \neq 0$ and maximum number of iteration is not reached **do**
  $\mathbf{q} = \mathbf{s}H$ ▷ Computing violated parity checks
  $\mathcal{B} = \{i \in [n] \mid q_i \geq \lfloor \frac{\deg(v_i)}{2} \rfloor + 1\}$ ▷ Majority Update Rule
  $\hat{c}_i \leftarrow \hat{c}_i \oplus 1; \quad \forall i \in \mathcal{B}$
  $\mathbf{s} = \mod(\hat{\mathbf{c}}H^T, 2)$
end while

senting the environment, A is the set of finite actions that the decision maker, or agent, can take, $P^a_{ss'} = P(s' \mid s, a) \triangleq P(S_{t+1} = s' \mid S_t = s, A_t = a)$ is the probability transition function $P : S \times A \times S \to [0, 1]$ which represents the probability of going to state s' from state s after taking action a, $R(s, a)$ is the reward function $R : S \times A \to \mathbb{R}$ which represents the immediate reward received after taking action a in state s, and $\gamma \in [0, 1]$ is the discount factor which denotes the importance of future rewards. γ close to 1 implies that future rewards are almost as important as immediate rewards, whereas γ close to 0 implies that only immediate rewards matter. A deterministic policy π is a mapping $\pi : S \to A$, where $\pi(s)$ denotes the action to take in state s. The goal in an MDP is to find an optimal policy $\pi^\star$ that maximizes the expected cumulative reward $\pi^\star = \arg\max_\pi \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s]$, where $\mathbb{E}_\pi$ is the expectation by taking actions under policy π.

B. Q-LEARNING

Q-learning is a model-free reinforcement learning algorithm that is used to solve Markov Decision Processes (MDPs). It has the ability to learn the optimal policy $\pi^\star$ directly from interactions with the environment without requiring the model of environment. The action-value function $Q : S \times A \to \mathbb{R}$ is defined to be the expected cumulative rewards starting from state s, taking action a, and using policy π to select actions afterwards

$$Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid S_0 = s, A_0 = a]. \tag{1}$$

Q-learning algorithm aims to learn the optimal action-value function

$$Q^\star(s, a) = \max_\pi Q_\pi(s, a) \tag{2}$$

that find the expected reward of taking action a in state s and following the optimal policy afterwards. The Q-learning algorithm iteratively updates the Q-values using the following rule

$$Q(s, a) \leftarrow Q(s, a) + \alpha[R(a, s) + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{3}$$

updating the Q-values based on the immediate rewards received and the estimated value of subsequent maximum action-value estimate where $\alpha \in (0, 1]$ is the step size for updating the Q values. The Q-learning algorithm is shown in Algorithm 2 [7], where s_E is the terminal state. The selec-

Algorithm 2 Q-learning Algorithm

Initialization: Initializing $Q(s, a)$ table randomly except $Q(s_E, a) = 0$, set epo to be the number of episodes.

Output: Q-table $Q(s, a) \quad \forall s \in S, a \in A$

for $i = 1, 2, \cdots,$ epo **do**

  while s_E `is not reached` **do**

    Using an exploration-exploitation strategy to select action a. ▷ Action selection

    Taking action a, observing the reward $R(s, a)$ and next state s'.

    updating Q-value: $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[R(a, s) + \gamma \max_{a'} Q(s', a')]$

    $s \leftarrow s'$

  end while

end for

tion of actions in the Q-learning algorithm is an important step that needs to be balanced between exploration, testing new actions, and exploitation, choosing the best known action. The ϵ-greedy is a strategy used for this purpose that selects action based on probability distribution determines by the parameter ϵ as follow

$$a_t = \begin{cases} \text{random action uniformly from } A, & \text{w.p. } \epsilon \\ \arg\max_{a \in A} Q(s_t, a), & \text{w.p. } 1 - \epsilon \end{cases} \tag{4}$$

A simple and effective way to reduce exploration in favor of exploitation as the learning process progresses is the use of linear decay in ϵ-greedy algorithm. By controlling the decay rate, the agent can effectively balance between discovering new actions and leveraging learned knowledge to maximize rewards. The linear decaying $\epsilon(t)$ at the episode t is given by

$$\epsilon(t) = \max(\epsilon_{min}, \epsilon_{max} - \frac{\epsilon_{max} - \epsilon_{min}}{N} \cdot t) \tag{5}$$

where ϵ_{max}/ϵ_{min} is the maximum/minimum value of ϵ and N is the total number of episodes.

BIT-FLIPPING STRATEGY

In this section, we explain how the decoding problem using bit flipping can be mapped into an MDP. We use maximum-likelihood decoding for a binary linear code to define the reward function for the MDP. Considering n independent memoryless channels described by its density function $P(y_i \mid c_i)$, the maximum likelihood problem can be defined as an optimization problem

$$\arg\max_{c \in \mathcal{C}} \prod_{i=1}^{n} P(y_i \mid c_i) = \arg\max_{c \in \mathcal{C}} \sum_{i=1}^{n} (1 - 2c_i) l_i \tag{6}$$

where

$$l_i \triangleq \ln \frac{P(y_n \mid 0)}{P(y_n \mid 1)} \tag{7}$$

is the channel log-likelihood ratio (LLR). The optimization problem can be modified as explained in [4] into a multi-stage process

$$\underset{\sum_{i=1}^{T'} \mathbf{h}_{a_i}^c = \mathbf{s}}{\arg\max} \sum_{i=1}^{T'} - \mid l_{a_i} \mid \tag{8}$$

where T' is the length of an episode to reach the terminal state, a_i-th bit is flipped in the time i, $\mathbf{h}_i^c$ is the i-th column of the parity-check matrix, $\mathbf{s} = cH^T$ is the syndrome of the received word. LLR of the i-bit, l_i, is the measure of reliability of that bit which can be interpreted as a negative reward $- \mid l_i \mid$ if the i-th bit is flipped. In other words, the goal is to find an error pattern that find the smallest summation of LLRs of bits that matches the syndrome.

C. State and Action Spaces

Given that the code dimension is n, it is likely that each of these n bits be erroneous. In other words, the action space A should be designed to allow the flipping of each bit at any time step. Therefore, we can state $A = \{1, 2, \cdots, n\}$. The decoding process can be terminated based on the value of the syndrome vector, which indicates the positions of unsatisfied checks in the code. Consequently, the state space can be defined by all possible binary syndromes of length m. The initial state $\mathbf{s}_0 = \mathbf{y}H^T$ represents the syndrome of the received vector. Next, we need to define the transition probability function $P(\mathbf{s}' \mid \mathbf{s}, a)$ for an MDP. By taking action a from state $\mathbf{s}$, we transition to state $\mathbf{s}' = \mathbf{s} + \mathbf{h}_a^c$ with probability 1, indicating that it is a deterministic MDP. Terminal state is an all-zero syndrome, indicating that a codeword is obtained and the decoding process should be terminated. We also impose a limit of L steps, which is the maximum length of each episode. If an episode is not terminated within L steps, a new episode begins.

D. Reward Strategy

Choosing a reward strategy is crucial in Reinforcement Learning algorithms because it significantly impacts the learning efficiency and the resulting policy. Since the goal of the decoding problem primarily relies on finding a codeword, we define a positive reward of 1 if an agent reaches the terminal state. We also need to penalize the action of flipping a bit based on its reliability. This helps identify the set of least reliable bits to find a codeword, thereby minimizing the probability of word error rate. Thus, we define a reward function for the Binary Symmetric channel based on Equation (8) to be

$$R(s, a, s') = \begin{cases} -\frac{1}{L} + 1, & \text{if } s' = 0 \\ -\frac{1}{L}, & \text{otherwise.} \end{cases} \tag{9}$$

Since in the BSC the reliability of all bits are equal, We use a constant negative reward of $-1/L$ to ensure it is small compared to the reward of 1 for reaching the terminal state.

E. Number of optimal policies

The Q-learning algorithm aims to find the optimal policy $\pi^\star$. In this section, we explore the existence and number of optimal policies that the Q-learning will eventually try to find.

Theorem 1. *The number of different optimal policies, or decoders, for the Binary Symmetric channels is greater than or equal to*

$$\prod_{i=2}^{t} \binom{n}{i} i \tag{10}$$

where n is the codelength and t is the error correction capability of the code.

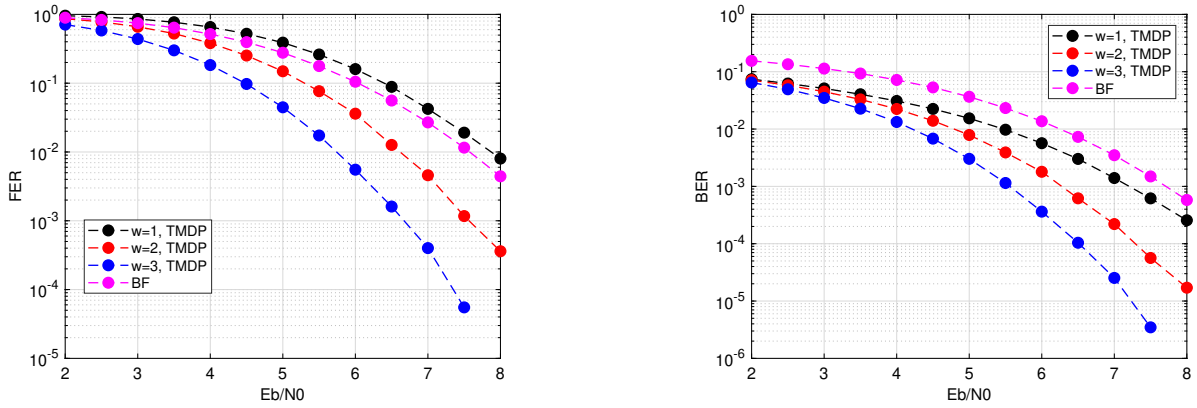
Proof. The total number of distinct syndromes is 2^{n-k}. Let we assume the code has the error correction capability of t, then, all erroneous patterns with weight less than or equal t are correctable. There is a one-to-one correspondence between syndromes and their associated lowest weight error patterns that satisfy $\sum_{i=1}^{t} \binom{n}{i} \leq 2^{n-k}$, which states the existence of correction of some of higher than weight t error patterns. There are i positions in the weight-i error pattern that can be flipped to correctly reach the weight-$(i-1)$ error pattern, where $2 \leq i \leq t$. Additionally, the multiplicity of weight-i error pattern is $\binom{n}{i}$. Thus, based on multiplication axiom, we can state that the number of possible optimal actions from weight-i error to weight-$(i-1)$ error is $\binom{n}{i}i$, where $2 \leq i \leq t$. Since the correction of error patterns can be viewed as a sequential procedure, we again can use the multiplication axiom. We can state that the total number of distinct optimal policies up to correction of weight-t error patterns is associated with the multiplication of possibilities of correcting weight-i error pattern to weight-$(i-1)$ error pattern for $2 \leq i \leq t$. Therefore, the number of distinct optimal policies up to correction of weight-t erroneous patterns are $\prod_{i=2}^{t} \binom{n}{i} i$. As mentioned earlier, it is possible for the code to correct some of higher than weight-t error patterns that makes the number of possible optimal actions more than that of found in Equation 10. Hence, the found number is a lower bound on the number possible optimal policies. □

TRUNCATED MDP

The proposed Markov Decision Process (MDP) faces challenges related to the number of states, which corresponds to the number of syndromes of the code, given by 2^{n-k}.This number grows exponentially, making Q-Learning methods infeasible for long codes and low-rate codes. In this section, we propose a truncated MDP model aimed at handling low-rate codes to learn their optimal policies. Binary linear block codes, specifically LDPC codes, may have complex decision regions that make finding an optimal policy difficult. It is known that the error correction capability of the code is a good estimate for its probability of error in a high SNR regime. Therefore, we consider only the states corresponding to syndromes related to erroneous patterns up to weight w. With this modification, we dramatically decrease the number of states to

$$\mid S \mid = \sum_{i=0}^{w} \binom{n}{i} \tag{11}$$

which is significantly smaller than the total number of syndromes. The action space can be modified for the states on the boundary (weight-w error patterns) to be a specific set related to their



(a) (b)

Figure 1: Simulation results for RM$(64, 45, 8)$ of Bit-Flipping decoder and learned Bit-Flipping based on Q-table for truncated MDPs with weight $w = \{1, 2, 3\}$. (a) Frame Error Rate, (b) Bit Error Rate.

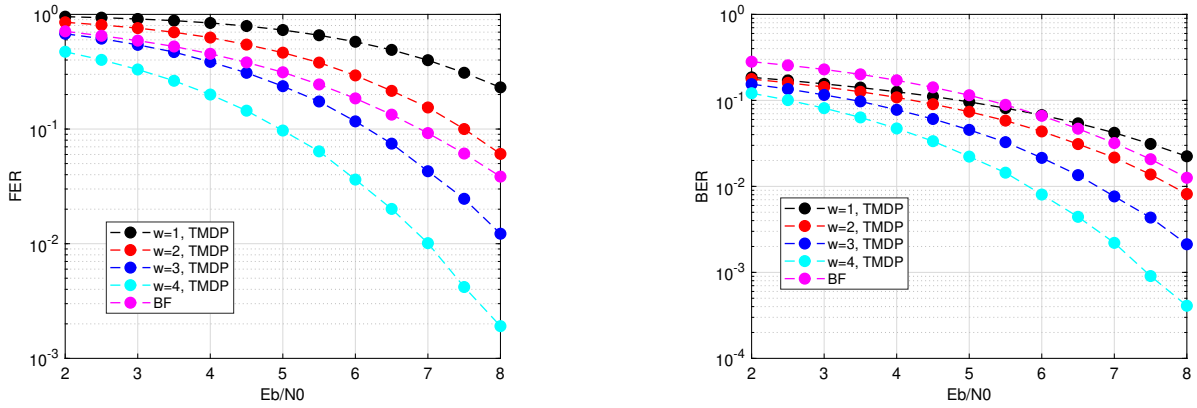
syndrome. Alternatively, the action space can remain as before, with a new reward of -1 assigned for actions that transition to states outside our defined state space, along with the end of episode.

NUMERICAL RESULTS

In this section, we utilize the learned bit-flipping strategy based on the Q-learning algorithm over the Binary Symmetric Channel with a truncated MDP up to a specific weight w. We compare the performance of this approach with the Bit-Flipping decoder as a baseline to demonstrate improvements in terms of Bit Error Rate (BER) and Frame Error Rate (FER). In Figure 1, we show the performance of the Reed-Muller (RM) code with parameters $(64, 45, 8)$. The performance of the Bit-Flipping decoder is sub-optimal in terms of both FER and BER. By using the proposed truncated MDP, we can accelerate the training process and make feasible the use of Q-learning algorithm for long codes to improve both BER and FER. In Figure 2, we select a random low-rate binary linear code code with parameters $(24, 6, 10)$ and rate $R = 0.25$. We improve the performance better than that of the bit-flipping decoder for the truncated MDP with weights $w = \{3, 4\}$.

F. Training Hyper-Parameters

We set the maximum number of decoding iterations to be $T' = 10$ and the discount factor to be $\gamma = 0.9$. For standard table Q-learning, we use the epsilon-greedy algorithm with a linear decay, setting the parameters to $\epsilon_{max} = 1$ and $\epsilon_{min} = 0.1$ over 10^7 episodes. The assumed learning rate is $\alpha = 0.7$.



(a) (b)

Figure 2: Simulation results for a binary random linear code with parameters $(24, 4, 10)$ of Bit-Flipping decoder and learned Bit-Flipping based on Q-table for truncated MDPs with weight $w = \{1, 2, 3, 4\}$. (a) Frame Error Rate, (b) Bit Error Rate.

CONCLUSIONS

In this paper, we propose a novel method to reduce the number of states of the MDP associated with binary linear codes. By reducing the number of states, we are able to accelerate the training procedure. In other words, we only focus on learning more probable region instead of the entire region around a codeword. With this modification, We are also able to effectively learn policies for both low-rate codes and long-length codes in addition to high-rate and short-length codes.

Acknowledgments

This work is supported by the NSF under grants CIF-2106189, CCF-2100013, ECCS/CCSS-2027844, ECCS/CCSS-2052751, and in part by the CoQREATE program under grant ERC-1941583, and Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded through JPL's Strategic University Research Partnerships (SURP) program. Bane Vasić has disclosed an outside interest in his startup company, Codelucida to The University of Arizona. Conflicts of interest resulting from this interest are being managed by The University of Arizona in accordance with its policies.

The authors are with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721, USA (e-mail: miladt@arizona.edu; asitpradhan@arizona.edu; vasic@ece.arizona.edu).

REFERENCES

[1] T. Gruber, S. Cammerer, J. Hoydis, and S. Ten Brink, "On deep learning-based channel decoding," in *2017 51st annual conference on information sciences and systems (CISS)*, pp. 1–6, IEEE, 2017.

[2] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 341–346, 2016.

[3] A. Bennatan, Y. Choukroun, and P. Kisilev, "Deep learning for decoding of linear codes - a syndrome-based approach," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1595–1599, 2018.

[4] F. Carpi, C. Häger, M. Martalò, R. Raheli, and H. D. Pfister, "Reinforcement learning for channel coding: Learned bit-flipping decoding," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 922–929, IEEE, 2019.

[5] S. Habib, A. Beemer, and J. Kliewer, "Belief propagation decoding of short graph-based channel codes via reinforcement learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 2, pp. 627–640, 2021.

[6] W. Ryan and S. Lin, *Channel codes: classical and modern*. Cambridge university press, 2009.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.