Fast SC-Flip Decoding of Polar Codes with Reinforcement Learning

Nghia Doan*, Seyyed Ali Hashemi†, Furkan Ercan*, Warren J. Gross*
*Department of Electrical and Computer Engineering, McGill University, Canada
†Department of Electrical Engineering, Stanford University, USA
nghia.doan@mail.mcgill.ca, ahashemi@stanford.edu, furkan.ercan@mail.mcgill.ca, warren.gross@mcgill.ca

***Abstract*—In this paper, we introduce a novel bit-flipping algorithm for fast successive cancellation (FSC) decoding of polar codes. In particular, we first propose a new bit-flipping strategy tailored to single parity-check (SPC) constituent codes of polar codes. A parameterized bit-flipping model is then developed and reinforcement learning (RL) is used to optimize the parameters. Our experimental results show that for a polar code of length 512 with 256 information bits, the proposed decoder has a better or similar error-correction performance compared to the state-of-the-art fast DSCF (FDSCF) decoding algorithm when the same number of maximum decoding attempts is considered.**

***Index Terms*—5G, polar codes, fast SC flip, machine learning**

I. Introduction

Polar codes are the first type of linear block codes that are proven to achieve the capacity of any binary symmetric channel [1]. With this characteristic, polar codes, concatenated with cyclic redundancy check (CRC) codes, are used in the enhanced mobile broadband (eMBB) control channel of the fifth generation of cellular wireless communication (5G) standard [2]. Successive cancellation (SC) decoding is a low-complexity algorithm introduced in [1] to decode polar codes. However, its serial nature prevents the decoder to reach a high decoding throughput. Fast SC (FSC) decoding algorithm was introduced in [3], [4] to improve the high decoding latency of SC decoding. However, for short to moderate code lengths SC and FSC decoding algorithms do not provide a satisfactory error-correction performance required by the 5G standard [2].

SC-Flip (SCF) decoding is an alternative algorithm used to decode CRC-polar concatenated codes [5]. Given that the initial SC decoding attempt does not satisfy the CRC verification, SCF decoding identifies the first error bit of the initial SC decoding and flips the estimated value of it in the following decoding attempts. The key challenge of SCF decoding is to accurately estimate the first error bit of the initial SC decoding. Dynamic SCF (DSCF) decoding [6] tackles this challenge by introducing a conditional error-probability model, which greatly improves the prediction accuracy of the first error bit, resulting in a significant error-correction performance improvement compared with SCF decoding. However, all SCF-based decoders experience a high decoding latency, which is inherently caused by the serial nature of SC decoding [7].

Several attempts have integrated fast decoding operations to SCF decoding to improve its decoding latency [7]–[10]. It was shown in [7] that using the DSCF-based bit flipping model for FSC decoding results in a better error-correction performance than the decoders in [8]–[10]. In this paper, we propose a novel bit-flipping algorithm for FSC decoding that significantly improves the error-correction performance of the decoder in [7]. In particular, a new bit-flipping strategy tailored to single parity-check (SPC) constituent codes is proposed. Furthermore, a new parameterized bit-flipping model based on [11] is developed to exploit the inherent correlations of the decoded bits under FSC decoding. We formalize the parameter optimization of the proposed bit-flipping model as an on-policy reinforcement learning (RL) problem and use RL techniques [12] to optimize the parameters during the course of decoding. Simulation results show that for a 5G polar code of length 512 with 256 information bits and concatenated with a 24-bit CRC, the proposed decoder has a better or similar error-correction performance compared to the state-of-the-art fast DSCF (FDSCF) decoding algorithm in [7], when the same number of maximum decoding attempts is considered.

II. Preliminaries

A. Polar Encoding

A polar code $\mathcal{P}(N, K)$ of length N with K information bits is constructed by applying a linear transformation to the binary message word $\boldsymbol{u} = \{u_0, u_1, \ldots, u_{N-1}\}$ as $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}^{\otimes n}$ where $\boldsymbol{x} = \{x_0, x_1, \ldots, x_{N-1}\}$ is the codeword, $\boldsymbol{G}^{\otimes n}$ is the n-th Kronecker power of the polarizing matrix $\boldsymbol{G} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$, and $n = \log_2 N$. The vector $\boldsymbol{u}$ contains a set $\mathcal{I}$ of K information bit indices and a set $\mathcal{I}^c$ of $N - K$ frozen bit indices. The positions of the frozen bits are known to the encoder and the decoder, and their values are set to 0. The codeword $\boldsymbol{x}$ is then modulated and sent through the channel. In this paper, binary phase-shift keying (BPSK) modulation and additive white Gaussian noise (AWGN) channel model are considered. Therefore, the soft vector of the transmitted codeword received by the decoder is given as $\boldsymbol{y} = (\mathbf{1} - 2\boldsymbol{x}) + \boldsymbol{z}$, where $\mathbf{1}$ is an all-one vector of size N, and $\boldsymbol{z} \in \mathbb{R}^N$ is a Gaussian noise vector with variance σ^2 and zero mean. In the log-likelihood ratio (LLR) domain, the LLR vector of the transmitted codeword is given as $\boldsymbol{\alpha}_n = \ln \frac{\Pr(\boldsymbol{x}=0|\boldsymbol{y})}{\Pr(\boldsymbol{x}=1|\boldsymbol{y})} = \frac{2\boldsymbol{y}}{\sigma^2}$.

B. Successive Cancellation Decoding

SC decoding can be represented on a polar code factor graph representation. An example of a factor graph for $\mathcal{P}(16, 8)$ is depicted in Fig. 1(a) with the frozen set

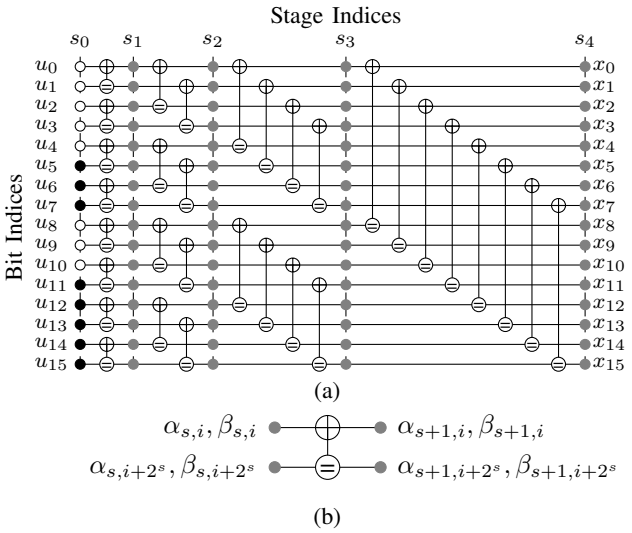

Fig. 1: (a) Factor graph representation of $\mathcal{P}(16, 8)$, and (b) a processing element (PE) of polar codes.

$\mathcal{I}^c = \{0, 1, 2, 3, 4, 8, 9, 10\}$. To obtain the message word, the soft LLR values and the hard bit estimations are propagated through all the processing elements (PEs), which are depicted in Fig. 1(b). Each PE performs the following computations: $\alpha_{s,i} = f(\alpha_{s+1,i}, \alpha_{s+1,i+2^s})$ and $\alpha_{s,i+2^s} = g(\alpha_{s+1,i}, \alpha_{s+1,i+2^s}, \beta_{s,i})$, where $\alpha_{s,i}$ and $\beta_{s,i}$ are the soft LLR value and the hard-bit estimation at the s-th stage and the i-th bit, respectively, and the min-sum approximation formulations of f and g are $f(a, b) = \min(|a|, |b|)\operatorname{sgn}(a)\operatorname{sgn}(b)$, and $g(a, b, c) = b+(1-2c)a$. The hard-bit values of the PE are then computed as $\beta_{s+1,i} = \beta_{s,i} \oplus \beta_{s,i+2^s}$ and $\beta_{s+1,i+2^s} = \beta_{s,i+2^s}$. The soft LLR values at the n-th stage are initialized to $\boldsymbol{\alpha}_n$ and the hard-bit estimation of an information bit at the 0-th stage is obtained as $\hat{u}_i = \beta_{0,i} = \frac{1-\operatorname{sgn}(\alpha_{0,i})}{2}$, $\forall i \in \mathcal{I}$.

C. Fast Successive Cancellation Decoding

SC decoding can also be illustrated in a binary tree representation [3]. Fig. 2(a) shows a full binary tree representation of $\mathcal{P}(16, 8)$, whose factor graph representation is depicted in Fig. 1(a). In [3], [4] the authors identified special constituent nodes of polar codes where maximum likelihood (ML) decoding can be performed to estimate the hard values of the parent nodes without the need to traverse to the left and child nodes. Therefore, the decoding latency of SC decoding can be greatly reduced. In this paper, we consider four types of special nodes, namely Rate-0, Rate-1, repetition (REP), and SPC nodes [3], [4]. A parent node ν located at the s-th stage ($s > 0$) of the binary tree contains N_ν LLR values and N_ν hard decisions associated with this node, where $N_\nu = 2^s$. Let $i_{\min_\nu}$ and $i_{\max_\nu}$ be the minimum and maximum indices of a node ν located at the s-th stage ($s > 0$) of the polar code binary tree, respectively, where $0 \le i_{\min_\nu} < i_{\max_\nu} \le N - 1$ and $i_{\max_\nu} - i_{\min_\nu} = N_\nu - 1$. The LLR and hard values associated with the parent node ν are given as $\boldsymbol{\alpha}_\nu = \{\alpha_{s,i_{\min_\nu}}, \ldots, \alpha_{s,i_{\max_\nu}}\}$ and $\boldsymbol{\beta}_\nu = \{\beta_{s,i_{\min_\nu}}, \ldots, \beta_{s,i_{\max_\nu}}\}$,

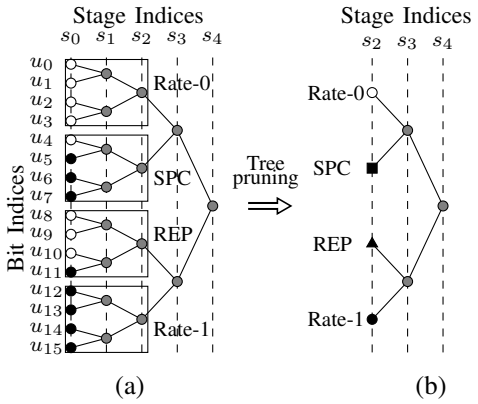

Fig. 2: (a) The full binary tree representation of $\mathcal{P}(16, 8)$ illustrated in Fig. 1(a), and (b) the pruned binary tree representation of the same polar code.

respectively. The definitions and decoding operations of each special node under FSC decoding are given as follows.

1) Rate-0 node: all the leaf nodes of a Rate-0 node are frozen bits. Therefore, all the hard values associated with the parent node are set to 0 [3].

2) Rate-1 node: all the leaf nodes of a Rate-1 node are information bits, the hard decisions associated with ν are [3]

$$\beta_{s,i} = \begin{cases} 0 & \text{if } \alpha_{s,i} > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{1}$$

where $i_{\min_\nu} \le i \le i_{\max_\nu}$.

3) REP node: all the leaf nodes of a REP node are frozen bits, except for $\beta_{0,i_{\max_\nu}}$, which is estimated as [4]

$$\hat{u}_{i_{\max_\nu}} = \beta_{0,i_{\max_\nu}} = \begin{cases} 0 & \text{if } \sum_{i=i_{\min_\nu}}^{i_{\max_\nu}} \alpha_{s,i} > 0, \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

4) SPC node: all the leaf nodes of an SCF node are information bits, except for $\beta_{0,i_{\min_\nu}}$. First, the hard decisions associated with the parent node ν are calculated as [4]

$$\beta_{s,i} = \begin{cases} 0 & \text{if } \alpha_{s,i} > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{3}$$

where $i_{\min_\nu} \le i \le i_{\max_\nu}$. Let $p_\nu = \bigoplus_{i=i_{\min_\nu}}^{i_{\max_\nu}} \beta_{s,i}$ be the parity check sum of the SPC node and $i_\nu = \arg\min_{i_{\min_\nu} \le i \le i_{\max_\nu}} |\alpha_{s,i}|$ be the bit index of the least reliable bit in ν. The value of β_{s,i_ν} is then updated as [4]

$$\beta_{s,i_\nu} := \beta_{s,i_\nu} \oplus p_\nu. \tag{4}$$

Fig. 1(b) shows the pruned binary tree representation of $\mathcal{P}(16, 8)$ depicted in Fig. 1(a), where all the considered special nodes in this paper are illustrated.

D. SCF and Dynamic SCF Decoding Algorithms

The error-correction performance of SC decoding for short to moderate polar codes is not satisfactory. To improve its error-correction performance, a CRC of length C is concatenated to the message word of polar codes to check whether SC decoding succeeded or not. If the estimated message word does

not satisfy the CRC after the initial SC decoding, a secondary SC decoding attempt is made by flipping the estimation of the first information bit that is most likely to be erroneous. In [5], the index of the first error bit is estimated as

$$i_e^* = \underset{\forall i \in \mathcal{I}}{\arg\min} |\alpha_{0,i}|. \tag{5}$$

This process can be performed multiple times by applying a predetermined number of SC decoding attempts, with each attempt flipping the estimation of a different information bit. If the resulting message word after one of the SC decoding attempts satisfies the CRC, the decoding is declared successful. This algorithm is referred to as SCF decoding [5]. However, the main challenge of SCF decoding is the poor prediction accuracy of i_e^* given in (5) [6]. To address this issue the authors in [6] proposed the DSCF decoding algorithm, where a more accurate bit-flipping metric is introduced and is written as [6]

$$Q_i = |\alpha_{0,i}| + \sum_{\substack{\forall j \in \mathcal{I} \\ j \leq i}} \frac{1}{\delta} \ln\left[1 + \exp\left(-\delta |\alpha_{0,j}|\right)\right], \tag{6}$$

where $\delta > 0$ is a perturbation parameter and is set to 0.3 as in [7]. The most probable bit-flipping position i_e^* under the DSCF decoding algorithm is then obtained as

$$i_e^* = \underset{\forall i \in \mathcal{I}}{\arg\min}\, Q_i. \tag{7}$$

The FDSCF decoding algorithm [7] improves the speed of DSCF decoding by incorporating the idea of FSC decoding.

III. Reinforcement-Learning-Aided Fast Successive-Cancellation-Flip Decoding

In this Section, we first introduce the proposed bit-flipping algorithm for FSC decoding that is tailored to the SPC nodes. A parameterized bit-flipping model based on [11] is then developed and RL techniques are used to train the parameters. Finally, experimental results are provided to evaluate the proposed algorithm.

A. Proposed Fast SCF Decoding Algorithm

The proposed decoding algorithm is based on the generation of a specific vector of LLR values. Consider the first FSC decoding attempt does not satisfy the CRC verification and let ν be a node located at the s-th stage ($s \geq 0$) of the polar decoding tree that is visited by FSC decoding. We construct a vector $\boldsymbol{\gamma} = \{\gamma_0, \gamma_1, \ldots\}$ by the following procedure:

- If ν is a leaf node that contains an information bit:

$$\boldsymbol{\gamma} := \text{concat}(\boldsymbol{\gamma}, \alpha_{0,i_{\min_\nu}}). \tag{8}$$

- If ν is a REP node:

$$\boldsymbol{\gamma} := \text{concat}(\boldsymbol{\gamma}, \sum_{i=i_{\min_\nu}}^{i_{\max_\nu}} \alpha_{s,i}). \tag{9}$$

- If ν is a Rate-1 node:

$$\boldsymbol{\gamma} := \text{concat}(\boldsymbol{\gamma}, \alpha_{s,i}), \tag{10}$$

for all $i_{\min_\nu} \leq i \leq i_{\max_\nu}$.

- If ν is an SPC node:

$$\boldsymbol{\gamma} := \text{concat}(\boldsymbol{\gamma}, \alpha_{s,i}), \tag{11}$$

for all $i_{\min_\nu} \leq i \leq i_{\max_\nu}$ and $i \neq i_\nu$, where $i_\nu = \arg\min_{i_{\min_\nu} \leq i \leq i_{\max_\nu}} |\alpha_{s,i}|$.

Note that $\text{concat}(\boldsymbol{\gamma}, a)$ indicates a concatenation of $a \in \mathbb{R}$ to the end of $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma} = \emptyset$ initially. Also note that the size of $\boldsymbol{\gamma}$ at the end of the first FSC decoding attempt is $K + C$. In addition, $\boldsymbol{\gamma}$ remains unchanged if ν does not satisfy any of the above conditions.

The generation of $\boldsymbol{\gamma}$ allows us to directly predict the index of the first erroneous bit in $\boldsymbol{\gamma}$, denoted as i_e^*. In the next FSC decoding attempt, the proposed decoder flips the hard decision associated with the i_e^*-th LLR value in $\boldsymbol{\gamma}$ and continues the FSC decoding operations. One important consequence of using $\boldsymbol{\gamma}$ to predict the bit-flipping positions is for SPC nodes. In fact, there is no need to perform a bit-flipping for the least reliable bit of an SPC node. The value of this bit is calculated from all the other bits in the SPC node to satisfy the parity-check constraint (see (4)). Therefore, the hard decision value of the least reliable bit is automatically adjusted when another bit in an SPC node is flipped. As a result, the proposed algorithm only considers $N_\nu - 1$ possibilities to identify a bit flip that occurs in an SPC node. This is significantly smaller than the maximum search space of size $\binom{N_\nu}{2}$ required to flip a pair of indices, especially as N_ν increases.

To estimate i_e^*, we utilize the approach in [11] and learn the correlations of the LLR values in $\boldsymbol{\gamma}$. Let η_i be the hard decision value of γ_i and let l_i^* be the likelihood ratio that η_i is correctly decoded given $\boldsymbol{y}$ and $\boldsymbol{u}$. l_i^* is calculated as

$$l_i^* = \max\left\{\frac{\Pr(\eta_i = 0|\boldsymbol{y}, \boldsymbol{u})}{\Pr(\eta_i = 1|\boldsymbol{y}, \boldsymbol{u})}, \frac{\Pr(\eta_i = 1|\boldsymbol{y}, \boldsymbol{u})}{\Pr(\eta_i = 0|\boldsymbol{y}, \boldsymbol{u})}\right\}. \tag{12}$$

i_e^* is then obtained as

$$i_e^* = \underset{\forall i,\ 0 \leq i < K+C}{\arg\min}\ l_i^*. \tag{13}$$

Since $\boldsymbol{u}$ is unknown, it is practically impossible to calculate l_i^* during the course of decoding. Thus, we estimate l_i^* as [11]

$$l_i^* \approx \prod_{\forall j,\ 0 \leq j < K} l_j^{\theta_{i,j}}, \tag{14}$$

where

$$l_j = \max\left\{\frac{\Pr(\eta_j = 0|\boldsymbol{y})}{\Pr(\eta_j = 1|\boldsymbol{y})}, \frac{\Pr(\eta_j = 1|\boldsymbol{y})}{\Pr(\eta_j = 0|\boldsymbol{y})}\right\} = \exp(|\gamma_j|) \tag{15}$$

and $\theta_{i,j} \in \mathbb{R}$ are perturbation parameters such that $\theta_{i,j} = \theta_{j,i}$ and $\theta_{i,i} = 1$, for $0 \leq i, j < K + C$. The perturbation parameters are such that if there is a correlation between the i-th and j-th decoded bits, $\theta_{i,j}$ and $\theta_{j,i}$ are nonzero values, otherwise $\theta_{i,j} = \theta_{j,i} = 0$.

To enable numerically stable computations, the likelihood ratio l_i^* can be transformed to the LLR domain as

$$\begin{aligned} M_i = \ln(l_i^*) &\approx \ln\left(\prod_{\forall j, 0 \leq j < K} \exp\left(\theta_{i,j}|\gamma_j|\right)\right) \\ &= \sum_{\forall j, 0 \leq j < K+C} \theta_{i,j}|\gamma_j|. \end{aligned} \tag{16}$$

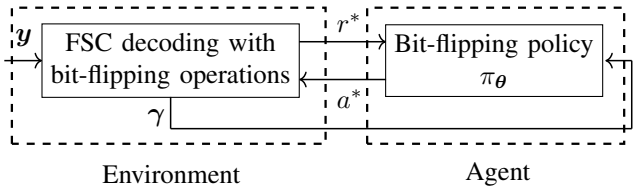

Fig. 3: The training setup of the proposed bit-flipping policy when formalized as a RL problem.

The most probable bit-flipping index i_e^* is then selected as

$$i_e^* = \underset{\forall i, 0 \le i < K+C}{\arg\min} M_i. \tag{17}$$

B. Parameter Optimization

In this section, we formalize the optimization of the matrix of parameters $\boldsymbol{\theta}$, with $\theta_{i,j}$ as the element in its i-th row and j-th column, as an on-policy RL problem. We use policy gradient techniques to train $\boldsymbol{\theta}$ [13, Chapter 13]. Let $\pi_{\boldsymbol{\theta}}$ be a bit-flipping policy characterized by $\boldsymbol{\theta}$. The input of $\pi_{\boldsymbol{\theta}}$ is $\boldsymbol{\gamma}$ and the output of $\pi_{\boldsymbol{\theta}}$ is a probability vector $\boldsymbol{p} = \{p_0, p_1, \ldots, p_{K+C-1}\}$, where $\boldsymbol{p} = \pi_{\boldsymbol{\theta}}(\boldsymbol{\gamma})$ and p_i indicates the probability that $i_e^* = i$. The value of p_i can be obtained from the bit-flipping metric in (16) as

$$p_i = \frac{\exp(-M_i)}{\sum_{j=0}^{K+C-1} \exp(-M_j)}. \tag{18}$$

It can be seen from (16) and (18) that the bit index that has the smallest bit-flipping metric is also the bit index that has the highest probability to be flipped.

Fig. 3 illustrates the parameter optimization framework used in this paper in an RL setup, in which the bit-flipping policy $\pi_{\boldsymbol{\theta}}$ acts as an online learning agent and the FSC decoder with the proposed bit-flipping operations is categorized as part of the environment. Given that the first FSC decoding attempt is not successful, a bit index a^* $(0 \le a^* < K+C)$ is first sampled from the categorical distribution of the bit-flipping policy $\pi_{\boldsymbol{\theta}}$, where the selection probability of the i-th bit is p_i. In the RL terminology, a^* is referred to as the action selected by the agent (with the bit-flipping policy $\pi_{\boldsymbol{\theta}}$). We denote by $r^* \in \{0, 1\}$ a reward value associated with the bit-flipping index a^*. If the resulting message word satisfies the CRC verification after the a^*-th bit of $\boldsymbol{\gamma}$ is flipped in the secondary FSC decoding attempt, a reward of 1 $(r^* = 1)$ is given to the a^*-th output of $\pi_{\boldsymbol{\theta}}$, otherwise the reward is set to 0 $(r^* = 0)$.

In practice, a bit-flipping set $\mathcal{A}$ that contains $T_{\max}$ $(1 \le T_{\max} \le K+C)$ different bit-flipping indices is considered at the secondary decoding attempts. The bit-flipping set $\mathcal{A}$ is first constructed to contain the most probable erroneous indices as

$$\mathcal{A} = \{a_0, a_1, \ldots, a_{T_{\max}-1}\}, \tag{19}$$

where $p_{a_0} \ge p_{a_1} \ge \ldots \ge p_{a_{T_{\max}-1}}$ and $a_0 = \arg\max_{\forall i, 0 \le i < K+C} p_i$. If the sampled bit index a^* is not in $\mathcal{A}$, the bit index $a_{T_{\max}-1}$ is replaced by a^*. The proposed decoder then performs consecutive FSC decoding attempts, each time flipping a different bit index given in $\mathcal{A}$.

Algorithm 1: RL-Aided FSCF Decoding

Input : $T_{\max}$, B, λ
Output: $\hat{\boldsymbol{u}}$

$t \leftarrow 1$, $\bar{r} \leftarrow 0$, $\boldsymbol{\nabla}_{\text{tmp}} \leftarrow \mathbf{0}$ // Initialization
while *True* **do**
  Obtain $\boldsymbol{\alpha_n}$ from the channel output $\boldsymbol{y}$
  $\hat{\boldsymbol{u}}, \boldsymbol{\gamma} \leftarrow \text{FSC}(\boldsymbol{\alpha_n})$ // Initial FSC Decoding
  if $\hat{\boldsymbol{u}}$ *fails the CRC test* **then**
    /* Action Selection */
    Obtain $\boldsymbol{M}$, $\boldsymbol{p}$, $\mathcal{A}$ using (16), (18), and (19)
    $a^* \sim \pi_{\boldsymbol{\theta}}$, $r^* \leftarrow 0$
    if $a^* \notin \mathcal{A}$ **then**
      $a_{T_{\max}-1} \leftarrow a^*$
    /* Proposed FSCF with Bit Flipping */
    for $j \leftarrow 0$ **to** $T_{\max} - 1$ **do**
      $\hat{\boldsymbol{u}}_{\text{tmp}} \leftarrow \text{FSCF}(\boldsymbol{\alpha_n}, a_j)$ // FSCF Decoding
      if $\hat{\boldsymbol{u}}_{tmp}$ *passes the CRC test* **then**
        $a^* \leftarrow a_j$, $r^* \leftarrow 1$, $\hat{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{u}}_{\text{tmp}}$
        break
    /* Parameter Optimization */
    $\boldsymbol{\nabla}_{\text{tmp}} \leftarrow \boldsymbol{\nabla}_{\text{tmp}} + \frac{\partial \ln p_{a^*}}{\partial \boldsymbol{\theta}}(r^* - \bar{r})$
    $\bar{r} \leftarrow \bar{r} + \frac{r^* - \bar{r}}{t}$ // Update baseline reward
    if $(t \mod B) == 0$ **then**
      $\boldsymbol{\nabla}_{\boldsymbol{\theta}} \leftarrow \boldsymbol{\nabla}_{\text{tmp}}/B$
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \lambda \boldsymbol{\nabla}_{\boldsymbol{\theta}}$
      $\boldsymbol{\nabla}_{\text{tmp}} \leftarrow \mathbf{0}$
    $t \leftarrow t + 1$ // Increase the time step
  Output $\hat{\boldsymbol{u}}$

Given a sample of the LLR vector $\boldsymbol{\gamma}$, the objective of the bit-flipping policy $\pi_{\boldsymbol{\theta}}$ is to derive an action a^* that maximizes the expected reward value of r^*, which is given as [12]

$$J(\boldsymbol{\theta}) = E_{a^* \sim \pi_{\boldsymbol{\theta}}}[r^*] \approx \frac{1}{|\mathcal{D}|} \sum_{\forall \boldsymbol{\gamma} \in \mathcal{D}} r^*, \tag{20}$$

where $\mathcal{D}$ is a mini-batch of the dataset that contains B different instances of $\boldsymbol{\gamma}$, and $B = |\mathcal{D}|$. The derivative of the objective function $J(\boldsymbol{\theta})$ with respect to the parameter set $\boldsymbol{\theta}$ can be derived as [12]

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} = \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{|\mathcal{D}|} \sum_{\forall \boldsymbol{\gamma} \in \mathcal{D}} \frac{\partial \ln p_{a^*}}{\partial \boldsymbol{\theta}}(r^* - \bar{r}), \tag{21}$$

where $\bar{r} \in \mathbb{R}$ is a baseline reward [13, Section 13.4]. The parameters in $\boldsymbol{\theta}$ are then updated using a variant of the stochastic-gradient ascent technique, where the parameters are updated as

$$\boldsymbol{\theta} := \boldsymbol{\theta} + \lambda \boldsymbol{\nabla}_{\boldsymbol{\theta}} \tag{22}$$

and $\lambda > 0$ is the learning rate.

We provide the details of the proposed decoder with the optimization of $\boldsymbol{\theta}$ in Algorithm 1. In Algorithm 1, t is the time step, $\boldsymbol{\nabla}_{\text{tmp}}$ is the matrix of size $(K+C) \times (K+C)$ that stores the gradients of $\boldsymbol{\theta}$, and $\mathbf{0}$ is an all-zero matrix of

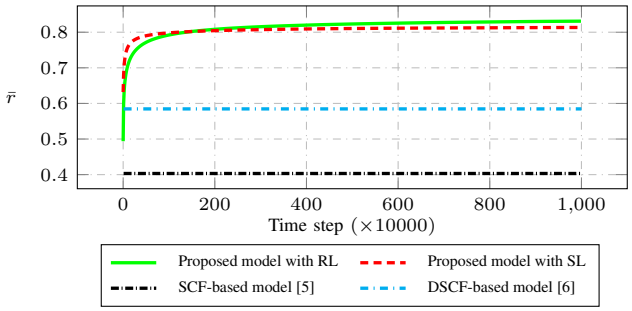

Fig. 4: The cumulative average rewards of various bit-flipping models when applied to the proposed bit-flipping algorithm. The simulation is carried out at $E_b/N_0 = 3$ dB for $\mathcal{P}(512, 256)$ with a 24-bit CRC, and $T_{\max} = 1$.

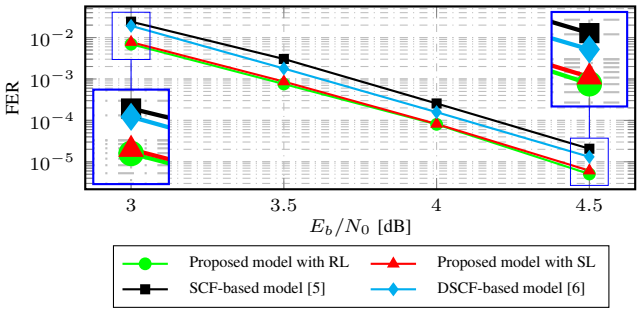

Fig. 5: The error-correction performance of the proposed bit-flipping algorithm with various bit-flipping models in Fig. 4.

size $(K+C) \times (K+C)$. In addition, the cumulative average reward is used as the reward baseline [13, Section 13.4], which is periodically updated at each time step.

C. Evaluation

In this section, we first provide the training results of the proposed bit-flipping model in Algorithm 1. Throughout this section, we use $\mathcal{P}(512, 256)$ concatenated to a 24-bit CRC[1] as stated in the 5G standard [2]. We use the bit-flipping models in [5] and [6] as a benchmark of estimating the first error bit in $\boldsymbol{\gamma}$. Based on [5], [6] the first error bit of $\boldsymbol{\gamma}$ can be obtained as follows.

- SCF-based model [5]: $i_e^* = \arg\min_{\forall i,\, 0 \le i < K+C} |\gamma_i|$.
- DSCF-based model [6]: $i_e^* = \arg\min_{\forall i,\, 0 \le i < K+C} Q_i$, where

$$Q_i = |\gamma_i| + \sum_{\forall j \le i} \frac{1}{0.3} \ln\left[1 + \exp\left(-0.3|\gamma_j|\right)\right].$$

In addition, we also consider an ideal bit-flipping model which can identify i_e^* correctly given $\boldsymbol{\gamma}$ and $\boldsymbol{u}$.

Fig. 4 compares the performance of different bit-flipping models (policies) in terms of the cumulative average reward, denoted as $\bar{r}$, when applied to the proposed bit-flipping algorithm. We also consider the conventional supervised learning (SL) technique as an optimization scheme for the parameter set $\boldsymbol{\theta}$ and compare it with the proposed RL-based optimization scheme. 10^5 training samples are obtained with all-zero codewords to train the parameter set $\boldsymbol{\theta}$ when the SL techniques are used. In fact, increasing the training samples of the SL approach does not improve $\bar{r}$ in our problem setting. For both the RL-based and SL-based approaches, we set $B = 100$, $\lambda = 2 \times 10^{-5}$ and use PyTorch with Adam optimizer [14] as the training framework[2].

It can be observed from Fig. 4 that the cumulative average reward of the proposed bit-flipping model when trained with RL or SL techniques outperforms that of the DSCF-based [6] and SCF-based [5] models. This is because the proposed algorithm utilizes a more powerful predictive model characterized by $\boldsymbol{\theta}$, where $|\boldsymbol{\theta}| = (K+C)^2$, while the models used in [5], [6] only contain up to a single trainable parameter. Moreover, the RL-based optimization approach provides a slight improvement in the cumulative average reward when compared with the SL-based approach, as the objective functions of the RL-based and SL-based approaches are different. The SL-based approach trains the model to make an ML decision given the training dataset. On the other hand, the RL-based approach trains the model to directly maximize the numerical reward $\bar{r}$. Fig. 5 shows the FER of the proposed bit-flipping algorithm with the bit-flipping models shown in Fig. 4 for $T_{\max} = 1$. The experimental results in Fig. 5 show that a direct optimization of $\boldsymbol{\theta}$ to maximize $\bar{r}$ also results in the best error-correction performance of the proposed decoder, when compared with other approaches in Fig. 4. Note that the parameter set $\boldsymbol{\theta}$ is optimized at each SNR value for the SL-based approach and the FERs are simulated using 10^7 frames at each SNR value.

It is worth mentioning that by formalizing the optimization of $\boldsymbol{\theta}$ as an RL problem, the training can be carried out at the decoder side without the need of pilot signals, which is suitable for a pilot-less communication system. Furthermore, unlike the SL approach, the training of $\boldsymbol{\theta}$ does not require a large memory to store the training dataset as observed from Algorithm 1.

Fig. 6 illustrates the FER curves of various fast SCF decoders with different values of $T_{\max}$ at $E_b/N_0 = \{3, 4\}$ dB. With $T_{\max} = 0$, all the decoders in Fig. 6 revert to the FSC decoder [4]. It can be seen from Fig. 6 that at the same value of $T_{\max}$ when $1 \le T_{\max} \le 8$, the proposed decoder has the best error-correction performance when compared to the decoders in [7], [8]. At $T_{\max} = 8$, the proposed decoder only experiences a negligible error-correction performance degradation compared to the ideal bit-flipping algorithm.

Fig. 7 shows the error probabilities of various bit-flipping algorithms of polar codes when $T_{\max} = 8$. The FERs of the state-of-the-art SCL-L decoding algorithm [15] are also plotted for comparison, where $L \in \{2, 4\}$ is the list size. It can be observed that the proposed decoder has a similar error-correction performance when compared with that of the DSCF [6] and FDSCF [7] decoders, respectively. At the target FER of 10^{-4}, the error probability of the proposed decoder is 0.25 dB and 0.2 dB better than that of the FSCF [8] and SCL-

[1]In this paper we use the 24-bit CRC specified as 24C.

[2]The parameters $\boldsymbol{\theta}$ are available at *https://github.com/nghiadt05/RL-FSCF*.

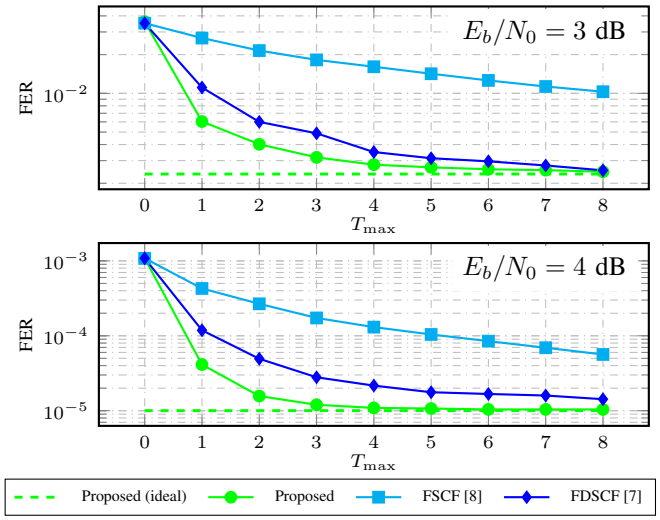

Fig. 6: The FER of various fast SCF decoding algorithms as a function of $T_{\max}$ at $E_b/N_0 = \{3, 4\}$ dB.

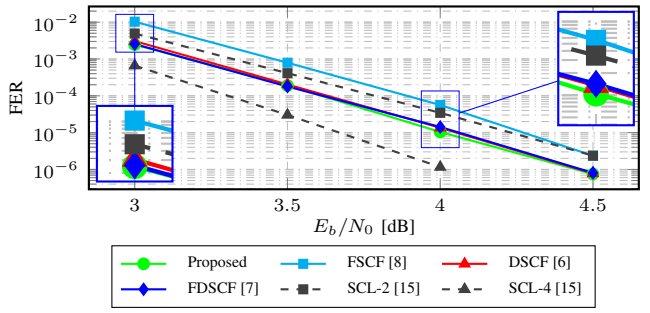

Fig. 7: The error-correction performance of various decoding algorithms. $T_{\max}$ is set to 8 for all the bit-flipping algorithms.

2 [15] decoders. At the same target FER, when compared with SCL-4, the proposed algorithm experiences an error-correction performance loss of 0.3 dB. In Fig. 8, the decoding latency, in terms of the average number of decoding attempts (T_{avg}) of various fast SCF decoders, is plotted for comparison. For all the decoders in Fig. 8, $T_{\max} = 8$. Note that the average number of decoding attempts of all the decoders depicted in Fig. 8 approaches 1 at high E_b/N_0 values. This indicates that at high E_b/N_0 values, the complexity of the decoders approaches the complexity of a single FSC decoder. In addition, the decoding latency of the proposed decoder in terms of T_{avg} is similar to that of the FDSCF decoder [7] at all the SNR values.

IV. Conclusion

In this paper, we first proposed a novel bit-flipping algorithm for fast successive-cancellation (SC) decoding [4] of polar codes. We then proposed a parameterized bit-flipping model, which has a better error-bit prediction accuracy when compared with the bit-flipping model introduced in [6]. Furthermore, the trainable parameters of the proposed decoder are optimized using reinforcement learning (RL) techniques, which are memory-efficient and do not need pilot signals as required by the supervised learning (SL) approach. Finally, we showed that for a 5G polar code of length 512 with

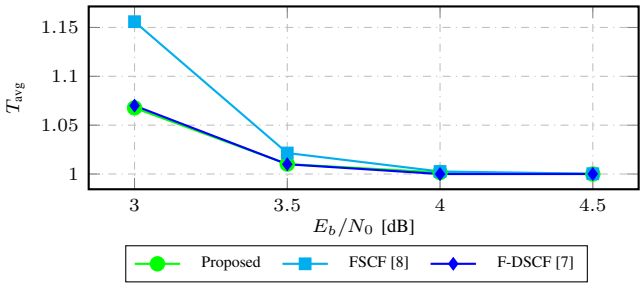

Fig. 8: The average number of decoding iterations of various fast SCF algorithms with $T_{\max} = 8$.

256 information bits, with the same number of maximum additional decoding attempts, the proposed decoder has a better or similar error probability when compared to the state-of-the-art fast dynamic SC-Flip (FDSCF) decoding algorithm in [7].

Acknowledgment

S. A. Hashemi is supported by a Postdoctoral Fellowship from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.

[2] 3GPP, "Multiplexing and channel coding (Release 10) 3GPP TS 21.101 v10.4.0." Oct. 2018. [Online]. Available: http://www.3gpp.org/ftp/Specs/2018-09/Rel-10/21_series/21101-a40.zip

[3] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE Commun. Lett.*, vol. 15, no. 12, pp. 1378–1380, October 2011.

[4] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 946–957, April 2014.

[5] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *48th Asilomar Conf. on Sig., Sys. and Comp.*, Nov 2014, pp. 2116–2120.

[6] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-SCFlip decoding of polar codes," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2333–2345, June 2018.

[7] F. Ercan, T. Tonnellier, N. Doan, and W. J. Gross, "Practical dynamic SC-flip polar decoders: Algorithm and implementation," *IEEE Trans. Signal Process.*, vol. 68, pp. 5441–5456, 2020.

[8] P. Giard and A. Burg, "Fast-SSC-flip decoding of polar codes," in *2018 IEEE Wireless Comm. and Net. Conf. Work.*, 2018, pp. 73–77.

[9] M. H. Ardakani, M. Hanif, M. Ardakani, and C. Tellambura, "Fast successive-cancellation-based decoders of polar codes," *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 4562–4574, 2019.

[10] F. Ercan, T. Tonnellier, and W. J. Gross, "Energy-efficient hardware architectures for fast polar decoders," *IEEE Trans. Circuits Syst. I*, vol. 67, no. 1, pp. 322–335, 2020.

[11] S. A. Hashemi, N. Doan, T. Tonnellier, and W. J. Gross, "Deep-learning-aided successive-cancellation decoding of polar codes," in *53rd Asilomar Conf. on Sig., Sys., and Comp.*, 2019, pp. 532–536.

[12] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015.