Learning to Construct Nested Polar Codes: An Attention-Based Set-to-Element Model

Yang Li, Zhitang Chen, Guochen Liu, Yik-Chung Wu, and Kai-Kit Wong

***Abstract*—As capacity-achieving codes under successive cancellation (SC) decoding, nested polar codes have been adopted in 5G enhanced mobile broadband. To optimize the performance of the code construction under practical decoding, e.g. SC list (SCL) decoding, artificial intelligence based methods have been explored in the literature. However, the structure of nested polar codes has not been fully exploited for code construction. To address this issue, this paper transforms the original combinatorial optimization problem for the construction of nested polar codes into a policy optimization problem for sequential decision, and proposes an attention-based set-to-element model, which incorporates the nested structure into the policy design. Based on the designed architecture for the policy, a gradient based algorithm for code construction and a divide-and-conquer strategy for parallel implementation are further developed. Simulation results demonstrate that the proposed construction outperforms the state-of-the-art nested polar codes for SCL decoding.**

***Index Terms*—Nested polar codes, learning to optimize, neural networks, attention mechanism.**

I. Introduction

With successive cancellation (SC) decoding, polar codes are the first class of capacity-achieving codes. By splitting physical channels into N polarized subchannels, each subchannel will polarize to either purely noiseless or completely noisy as N increases to infinity, and the fraction of noiseless subchannels approaches the Shannon capacity [1].

Due to this property, polar codes select only k ($k < N$) most reliable subchannels to carry information bits. Consequently, the coding performance highly depends on the accurate selection of the reliable subchannels. This subchannel selection is also called code construction, which can be achieved by density evolution [2], Gaussian approximation to density evolution (DE/GA) [3], and upgrading/downgrading of channels [4]. Moreover, a polarization weight method can generate a universal reliability order of the subchannels for all channel conditions [5], [6]. This method has been adopted for the construction of 5G nested polar codes [7], where the subchannels carrying information bits are nested successively according to the reliability order.

Yang Li is with Shenzhen Research Institute of Big Data, Shenzhen 518172, China (e-mail: liyang@sribd.cn).

Zhitang Chen and Guochen Liu are with Noah's Ark Lab, Huawei Technologies Co., Ltd., Shenzhen 518129, China (e-mail: {chenzhitang2, liuguochen1}@huawei.com).

Yik-Chung Wu is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: ycwu@eee.hku.hk).

Kai-Kit Wong is with the Department of Electronic and Electrical Engineering, University College London, London WC1E 6BT, U.K. (e-mail: kai-kit.wong@ucl.ac.uk).

While the above state-of-the-art code constructions achieve good error-correction performance with SC decoding, they are not necessarily competent under variants of SC decoding [8]. To address this issue, the construction of polar codes should be customized for the target decoding algorithm. In particular, the construction of polar codes for SC list (SCL) decoding have drawn a lot of attentions [9]–[11]. However, these heuristic constructions still lack theoretical guarantees, and the optimal construction of polar codes for SCL decoding is still unknown.

Inspired by the success of artificial intelligence (AI) in many application domains [12], researchers have attempted to explore whether the code construction with AI can perform comparably or even better than other heuristic approaches [13]–[16]. In particular, a genetic algorithm and a tabular reinforcement learning algorithm were proposed for code constructions in [13] and [14], respectively. Moreover, neural networks were applied to construct polar codes with a target rate [15] and nested polar codes [16]. It is noteworthy that these AI assisted code constructions provide superior performance to classic constructions for particular decoding algorithms, e.g., SCL decoding.

This paper proposes a learning based approach for nested polar code construction, which is a challenging combinatorial optimization problem. We first transform this problem into a stochastic policy optimization problem for sequential decision, and then represent the policy by a customized neural network. Specifically, the neural network is factorized as a series of set-to-element modules, where the input and output of each module are a set and an element, respectively. To construct a sequence of nested sets, the output element of each module should be out of the input set. We adopt set representations [17] for the input set. In order to determine the output element, we utilize the attention mechanism based on an alignment model, which scores how well the input set and each target element match [18]. Furthermore, we propose a gradient based algorithm to minimize the average loss of the policy, and develop a divide-and-conquer strategy for parallel implementation. Simulation results demonstrate that the proposed construction achieves better performance than the state-of-the-art nested polar codes for SCL decoding.

II. Problem Formulation

A. Nested Polar Codes

A polar code with code length N contains $k \in \{1, 2, \ldots, N\}$ nonfrozen bits and $N - k$ frozen bits, where the

k nonfrozen bits include information bits and cyclic redundancy check (CRC) bits (if CRC is used). We refer to the selection of k nonfrozen bit positions as polar code construction. Let $\{\pi_j\}_{j=1}^k$ denote the set of k nonfrozen bit positions, where $\pi_j \in \{1, 2, \ldots, N\}$ is a position index. For any given code length N, the construction of a sequence of nested polar codes refers to the selection of a permutation $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ of the integer sequence $(1, \ldots, N)$, such that the first k elements of $\boldsymbol{\pi}$ compose the nonfrozen bit positions of the k-th polar code. Define the performance loss (e.g., the block error ratio (BLER)) of the polar code corresponding to $\{\pi_j\}_{j=1}^k$ as $J_k(\{\pi_j\}_{j=1}^k)$, where the function J_k may vary with k since the codes with different k may be decoded under different conditions (e.g., SNRs). Consequently, the construction of nested polar codes can be formulated as

$$\min_{\boldsymbol{\pi}} \sum_{k=1}^{N} c_k J_k(\{\pi_j\}_{j=1}^k), \tag{1}$$

where c_k is a weight providing the trade-off among the codes with different k. However, problem (1) is challenging due to two aspects. First, $J_k(\{\pi_j\}_{j=1}^k)$ usually lacks an explicit expression and hence requires to be evaluated by simulations [16]. Second, the number of feasible solutions is equal to that of possible permutations for a length-N sequence. If $N = 1024$, the permutation number $N!$ would approach 10^{253}, which makes the problem highly intractable.

B. Problem Transformation

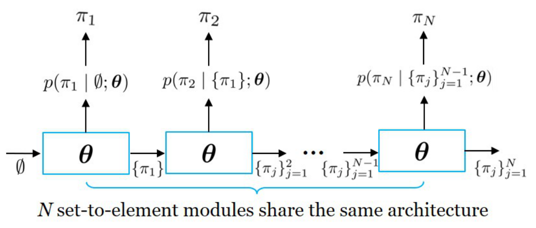
To avoid the brute-force search in such a huge space, we reformulate problem (1) into a stochastic policy optimization problem for sequential decision. In particular, we treat $\boldsymbol{\pi}$ as a random variable and define the policy as the probability distribution of $\boldsymbol{\pi}$, which assigns a probability for each possible permutation. Considering a variable $\boldsymbol{\theta}$ that contains a set of continuous parameters and parameterizing the probability distribution of $\boldsymbol{\pi}$ as $p(\boldsymbol{\pi}; \boldsymbol{\theta})$, we can transform the original combinatorial optimization problem (1) into a continuous optimization problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{k=1}^{N} c_k J_k(\{\pi_j\}_{j=1}^k) \right], \tag{2}$$

where the expectation is over the probability distribution $p(\boldsymbol{\pi}; \boldsymbol{\theta})$. Once $\boldsymbol{\theta}^*$ is obtained, the solution of problem (1) can be selected from a number of permutations sampled from $p(\boldsymbol{\pi}; \boldsymbol{\theta}^*)$. In the following, we will represent the specific expression of $p(\boldsymbol{\pi}; \boldsymbol{\theta})$ by a customized neural network in Section III, and then propose a gradient based algorithm to solve problem (2) in Section IV.

III. Attention-Based Set-to-Element Model

In this section, we design a neural network architecture to represent the specific expression of $p(\boldsymbol{\pi}; \boldsymbol{\theta})$. Instead of representing $p(\boldsymbol{\pi}; \boldsymbol{\theta})$ by the generic multi-layer perceptrons (MLPs), we incorporate the property of $p(\boldsymbol{\pi}; \boldsymbol{\theta})$ and the structure of



π_1
π_2
π_N
$p(\pi_1 \mid \emptyset; \boldsymbol{\theta})$
$p(\pi_2 \mid \{\pi_1\}; \boldsymbol{\theta})$
$p(\pi_N \mid \{\pi_j\}_{j=1}^{N-1}; \boldsymbol{\theta})$
$\emptyset$
$\boldsymbol{\theta}$
$\{\pi_1\}$
$\{\pi_j\}_{j=1}^2$
$\{\pi_j\}_{j=1}^{N-1}$
$\{\pi_j\}_{j=1}^N$
N set-to-element modules share the same architecture

Fig. 1. Represent $p(\boldsymbol{\pi}; \boldsymbol{\theta}) = \prod_{k=1}^N p(\pi_k \mid \{\pi_j\}_{j=1}^{k-1}; \boldsymbol{\theta})$ by an N-module neural network with each module sharing the same architecture.

$\boldsymbol{\pi}$ into the neural network architecture [19]. Specifically, by applying the chain rule, we have

$$p(\boldsymbol{\pi}; \boldsymbol{\theta}) = \prod_{k=1}^{N} p(\pi_k \mid \{\pi_j\}_{j=1}^{k-1}; \boldsymbol{\theta}), \tag{3}$$

where $\{\pi_j\}_{j=1}^0 \triangleq \emptyset$. Based on (3), the neural network representing $p(\boldsymbol{\pi}; \boldsymbol{\theta})$ can be factorized as a series of modules, where the input and output of the k-th module are $\{\pi_j\}_{j=1}^{k-1}$ and π_k, respectively. Since the input $\{\pi_j\}_{j=1}^{k-1}$ and the output π_k are a set and an element, respectively, we refer to this module as a set-to-element module. Based on this observation, we design an N-module neural network as shown in Fig. 1, with each module sharing the same architecture. In this way, the total number of training parameters, i.e., the dimension of $\boldsymbol{\theta}$, can be much reduced.

A. Representations for $\{\pi_j\}_{j=1}^{k-1}$ and π_k

To further build each set-to-element module in Fig. 1, we need to design representations for the input $\{\pi_j\}_{j=1}^{k-1}$ and the output π_k, as well as $p(\pi_k \mid \{\pi_j\}_{j=1}^{k-1}; \boldsymbol{\theta})$. To this end, we propose vector-form representations for π_k and $\{\pi_j\}_{j=1}^{k-1}$, respectively. Such vector-form representations can be fed into the module using vector-space calculations. Specifically, by representing $n \in \{1, \ldots, N\}$ as a vector optimization variable $\mathbf{x}_n \in \mathbb{R}^d$, where the embedding size d is commonly selected as a power of 2 via experiments, we vectorize $\{\pi_j\}_{j=1}^k$ as

$$\mathbf{y}_k = \begin{cases} \mathbf{y}_0, & k = 0, \\ \sum_{j=1}^k \mathbf{x}_{\pi_j}, & k = 1, \ldots, N, \end{cases} \tag{4}$$

where $\mathbf{y}_0 \in \mathbb{R}^d$ is a vector optimization variable representing the initial input $\emptyset$, and $\sum_{j=1}^k \mathbf{x}_{\pi_j}$ represents $\{\pi_j\}_{j=1}^k$ for $k \geq 1$. This representation exploits an important permutation-invariant property [17], i.e., any permutation of the elements in $(\pi_1, \ldots, \pi_k)$ results in the same set representation $\mathbf{y}_k$. This makes the representation of $\{\pi_j\}_{j=1}^k$ invariant regardless of the generated order of its elements.

With $\mathbf{x}_{\pi_k}$ and $\mathbf{y}_{k-1}$ representing π_k and $\{\pi_j\}_{j=1}^{k-1}$, respectively, we can express the conditional probability in (3) as

$$p(\pi_k \mid \{\pi_j\}_{j=1}^{k-1}; \boldsymbol{\theta}) = p(\mathbf{x}_{\pi_k} \mid \mathbf{y}_{k-1}; \{\mathbf{x}_n\}_{n=1}^N, \mathbf{y}_0, \boldsymbol{\phi}), \tag{5}$$

where the original optimization variable $\boldsymbol{\theta}$ is specified as $\{\mathbf{x}_n\}_{n=1}^N$, $\mathbf{y}_0$, and $\boldsymbol{\phi}$. In particular, $\{\mathbf{x}_n\}_{n=1}^N$ and $\mathbf{y}_0$ are used for representing $\{\pi_j\}_{j=1}^{k-1}$ as shown in (4), and $\boldsymbol{\phi}$ is used for representing the conditional probability $p(\mathbf{x}_{\pi_k} \mid$

$\mathbf{y}_{k-1}; \{\mathbf{x}_n\}_{n=1}^N, \mathbf{y}_0, \boldsymbol{\phi})$, which will be represented more specifically later.

B. Representations for $p(\mathbf{x}_{\pi_k} \mid \mathbf{y}_{k-1}; \{\mathbf{x}_n\}_{n=1}^N, \mathbf{y}_0, \boldsymbol{\phi})$

Before we represent $p(\mathbf{x}_{\pi_k} \mid \mathbf{y}_{k-1}; \{\mathbf{x}_n\}_{n=1}^N, \mathbf{y}_0, \boldsymbol{\phi})$, we first briefly review the basic idea of attention mechanism proposed in machine translation [18]. Attention mechanism is a neural network architecture that decides a target word based on a context vector. The context vector is computed as a weighted sum of different source words, where each weight reflects the importance of each source word with respect to the current state. This weighted sum intuitively decides which parts of the source sentence to pay attention to. In [18], the weights are calculated as the normalization of the output of an alignment model, which scores how well the current state and each source word match.

To represent $p(\mathbf{x}_{\pi_k} \mid \mathbf{y}_{k-1}; \{\mathbf{x}_n\}_{n=1}^N, \mathbf{y}_0, \boldsymbol{\phi})$, we adopt the attention mechanism to calculate a context vector $\tilde{\mathbf{y}}_{k-1}$ for deciding the target output $\mathbf{x}_{\pi_k}$. Specifically, we first compute a weight $w_{k-1,n}$ for scoring how well $\mathbf{y}_{k-1}$ and each $\mathbf{x}_n$ match by the alignment model[1] [18]:

$$\begin{cases} \mathbf{v}_1^T \tanh\left(\mathbf{A}_1 \left[\mathbf{x}_n^T, \mathbf{y}_{k-1}^T\right]^T\right), & \forall n \notin \{\pi_j\}_{j=1}^{k-1}, \\ -\infty, & \text{otherwise}, \end{cases} \tag{6}$$

where $\mathbf{A}_1 \in \mathbb{R}^{d\times 2d}$ and $\mathbf{v}_1 \in \mathbb{R}^d$ are parameters to be optimized. In (6), the elements already in $\{\pi_j\}_{j=1}^{k-1}$ are masked by $-\infty$ due to the nested structure, where the elements in $\{\pi_j\}_{j=1}^{k-1}$ are excluded as candidates of π_k. Consequently, the context vector is computed as $\tilde{\mathbf{y}}_{k-1} = \sum_{n=1}^N q_{k-1,n}\mathbf{x}_n$, where $q_{k-1,n} = \frac{e^{w_{k-1,n}}}{\sum_{j=1}^N e^{w_{k-1,j}}}$ is the normalization of the weight $w_{k-1,n}$.

Then we further score how well the context vector $\tilde{\mathbf{y}}_{k-1}$ and each $\mathbf{x}_n$ match by calculating a weight $\tilde{w}_{k-1,n}$ as

$$\begin{cases} \mathbf{v}_2^T \tanh\left(\mathbf{A}_2 \left[\mathbf{x}_n^T, \tilde{\mathbf{y}}_{k-1}^T\right]^T\right), & \forall n \notin \{\pi_j\}_{j=1}^{k-1}, \\ -\infty, & \text{otherwise}, \end{cases} \tag{7}$$

where $\mathbf{A}_2 \in \mathbb{R}^{d\times 2d}$ and $\mathbf{v}_2 \in \mathbb{R}^d$ are parameters to be optimized. To clip $\tilde{w}_{k-1,n}$ in a reasonable range, we use

$$r_{k-1,n} = \begin{cases} C\tanh(\tilde{w}_{k-1,n}), & \forall n \notin \{\pi_j\}_{j=1}^{k-1}, \\ -\infty, & \text{otherwise}, \end{cases} \tag{8}$$

where C is a tuning hyperparameter that controls the range of the weights. Since the clipped weight $r_{k-1,n}$ reflects the importance of each $\mathbf{x}_n$ in the current context, we normalize it in $[0, 1]$ to obtain the probability with respect to $\mathbf{x}_n$:

$$p(\mathbf{x}_{\pi_k} = \mathbf{x}_n \mid \mathbf{y}_{k-1}; \{\mathbf{x}_n\}_{n=1}^N, \mathbf{y}_0, \boldsymbol{\phi}) = \frac{e^{r_{k-1,n}}}{\sum_{j=1}^N e^{r_{k-1,j}}}, \tag{9}$$

where $\boldsymbol{\phi}$ is specified as all the parameters to be optimized as shown in (6)-(9), including $\mathbf{A}_1$, $\mathbf{v}_1$, $\mathbf{A}_2$, and $\mathbf{v}_2$.

[1]Alignment model and dot-product attention are similar in complexity. In our simulations, we find that they are also nearly the same in performance. Thus, we only use alignment model for ease of descriptions.

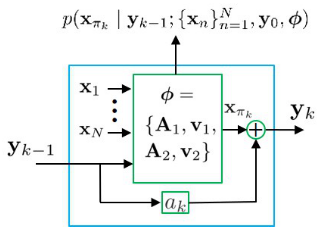

$p(\mathbf{x}_{\pi_k} \mid \mathbf{y}_{k-1}; \{\mathbf{x}_n\}_{n=1}^N, \mathbf{y}_0, \boldsymbol{\phi})$
$\mathbf{x}_1$
$\mathbf{x}_N$
$\mathbf{y}_{k-1}$
$\boldsymbol{\phi} = \{\mathbf{A}_1, \mathbf{v}_1, \mathbf{A}_2, \mathbf{v}_2\}$
$\mathbf{x}_{\pi_k}$
$\mathbf{y}_k$
a_k

Fig. 2. The structure of the k-th attention-based set-to-element module.

Algorithm 1 Code Construction for Nested Polar Codes

```
1:  Input: number of epochs $E$, batch size $B$, decay $\beta$, weights $\{c_k\}_{k=1}^N$
2:  Initialize: $\boldsymbol{\theta}$, $\boldsymbol{\pi}^*$ is sampled from $p(\boldsymbol{\pi};\boldsymbol{\theta})$, $L^* \leftarrow \sum_{k=1}^N c_k J_k(\{\pi_j^*\}_{j=1}^k)$
3:  for epoch $= 1, \ldots, E$
4:      $\boldsymbol{\pi}^{(i)}$ is sampled from $p(\boldsymbol{\pi};\boldsymbol{\theta})$, $\forall i \in \{1, \ldots, B\}$
5:      $L_i \leftarrow \sum_{k=1}^N c_k J_k(\{\pi_j^{(i)}\}_{j=1}^k)$, $\forall i \in \{1, \ldots, B\}$
6:      $j \leftarrow \arg\min(L_1, \ldots, L_B)$
7:      if $L_j < L^*$
8:          $\boldsymbol{\pi}^* \leftarrow \boldsymbol{\pi}^{(j)}$
9:          $L^* \leftarrow L_j$
10:         $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
11:     end
12:     $\mathbf{g}(\boldsymbol{\theta}) \leftarrow \frac{1}{B}\sum_{i=1}^B (L_i - b) \cdot \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\pi}^{(i)};\boldsymbol{\theta})$
13:     $\boldsymbol{\theta} \leftarrow$ Adam$(\boldsymbol{\theta}, \mathbf{g}(\boldsymbol{\theta}))$
14:     $b \leftarrow \beta b + (1-\beta)\frac{1}{B}\sum_{i=1}^B L_i$
15: end
16: Output: $\boldsymbol{\theta}^*, \boldsymbol{\pi}^*$
```

Once $\mathbf{x}_{\pi_k}$ is sampled from (9), according to (4), we can update the representation of $\{\pi_j\}_{j=1}^k$ as

$$\mathbf{y}_k = a_k \mathbf{y}_{k-1} + \mathbf{x}_{\pi_k}, \quad \text{with } a_k = \begin{cases} 0, & k = 1, \\ 1, & k = 2, \ldots, N. \end{cases} \tag{10}$$

Based on (5)-(10), the structure of the proposed attention-based set-to-element module can be illustrated in Fig. 2, which composes the k-th module of the whole model as shown in Fig. 1.

IV. Code Construction with Gradient Based Algorithm

Based on the stochastic policy designed in Section III, we further propose a gradient based algorithm to solve the continuous optimization problem (2). Specifically, we can derive the gradient of the cost function in (2) as

$$\mathbb{E}_{\boldsymbol{\pi}}\left[\left(\sum_{k=1}^N c_k J_k(\{\pi_j\}_{j=1}^k) - b\right) \cdot \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\pi};\boldsymbol{\theta})\right], \tag{11}$$

where b is the baseline to reduce the gradient variance. A simple and popular choice is the exponential moving average of $\sum_{k=1}^N c_k J_k(\{\pi_j\}_{j=1}^k)$. Consequently, we can apply the gradient based solver Adam [20] to solve problem (2).

The overall process is summarized in Algorithm 1. Specifically, in each epoch $= 1, \ldots, E$, a batch of permutations $\{\boldsymbol{\pi}^{(i)}\}_{i=1}^B$ are sampled from $p(\boldsymbol{\pi};\boldsymbol{\theta})$ as shown in line 4. According to the cost function in (2), the loss is calculated as $L_i = \sum_{k=1}^N c_k J_k(\{\pi_j^{(i)}\}_{j=1}^k)$ in line 5. From line 6 to line 11, the permutation with the minimum loss is always kept as $\boldsymbol{\pi}^*$. In line 12, $\mathbf{g}(\boldsymbol{\theta})$ is the mini-batch gradient that

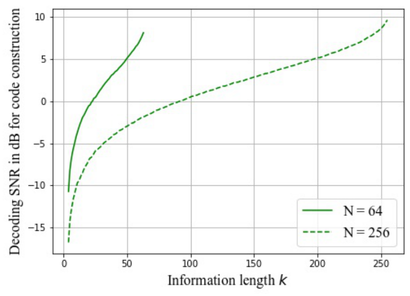


Fig. 3. The decoding SNR for code construction is set to the values under which 5G nested polar codes [7] approach BLER $= 0.01$.

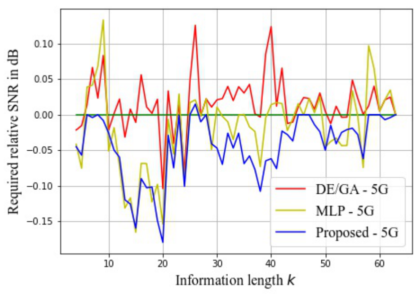


Fig. 4. Required relative SNR between each scheme and 5G nested polar codes when BLER $= 0.01$ and $N = 64$ under SCL-Genie decoder.

approximates the original gradient in (11). Then the variable $\boldsymbol{\theta}$ is updated by a mini-batch gradient descent step in line 13, and the baseline b is updated by an exponential moving average in line 14. By repeating this process, the variable $\boldsymbol{\theta}$ can be constantly optimized, and the corresponding $\boldsymbol{\pi}^*$ contains the final solution of the sequence of nested polar codes.

To well scale up to the long code design, we can implement Algorithm 1 in a divide-and-conquer manner. Specifically, since the cost function $\mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{k=1}^{N} c_k J_k(\{\pi_j\}_{j=1}^{k})\right]$ in (2) is additively separable, we can divide an initial $\boldsymbol{\pi}$ into several segments, where each segment is optimized with Algorithm 1 in parallel. Therefore, after each segment is optimized, the overall $\boldsymbol{\pi}$ can be improved to a better solution. Moreover, by dividing the new permutation in a different way, and optimizing each segment with Algorithm 1 in parallel again, the overall $\boldsymbol{\pi}$ can be further improved. In Section V, we will show that, only by dividing twice, the proposed Algorithm 1 assisted with the divide-and-conquer strategy achieves good performance.

In each epoch of Algorithm 1, we recode the evaluated BLER of each code to a lookup table, so that the computational complexity of Algorithm 1 is dominated by the mini-batch gradient decent updates. Specifically, the dimension of $\boldsymbol{\theta}$ (including $\{\mathbf{x}_n\}_{n=1}^{N}$, $\mathbf{y}_0$, $\mathbf{A}_1$, $\mathbf{v}_1$, $\mathbf{A}_2$, and $\mathbf{v}_2$) is $Nd + d + 2d^2 + d + 2d^2 + d = (N+3)d + 4d^2$. Thus, the complexity per epoch is $\mathcal{O}\left(B\left((N+3)d + 4d^2\right)\right)$ and the total complexity of Algorithm 1 is $\mathcal{O}\left(EB\left((N+3)d + 4d^2\right)\right)$.

V. Simulation Results

In this section, we evaluate the performance of the proposed attention-based set-to-element model and the corresponding Algorithm 1 for constructing nested polar codes with $N = 64$ and 256, respectively.

The simulation setting is as follows. For the proposed attention-based set-to-element model, the embedding size d is set as 128, and the hyperparameter C in (9) is set as 10. Moreover, for Algorithm 1, the epoch number is $E = 20000$, the batch size is $B = 100$, and the decay of the exponential moving average is $\beta = 0.99$. During the iterations, the step size for gradient descent is fixed as 10^{-3}, and the l_2 norm of the gradients are clipped to 1 to avoid the gradient exploding.

To achieve fairness among the polar codes with different information lengths k, the weight c_k in (2) is equally set

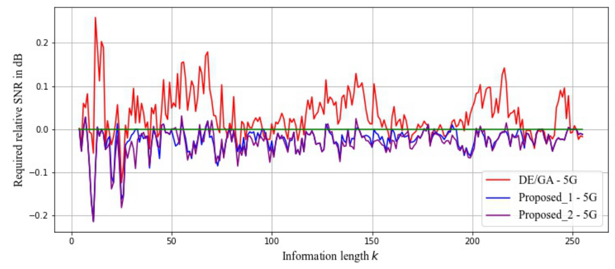
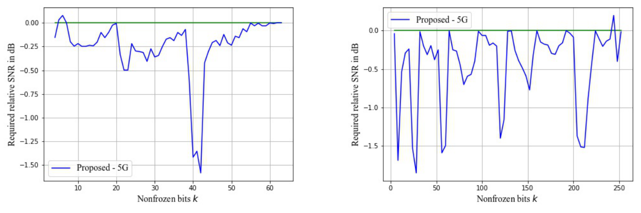


Fig. 5. Required relative SNR between each scheme and 5G nested polar codes when BLER $= 0.01$ and $N = 256$ under SCL-Genie decoder.

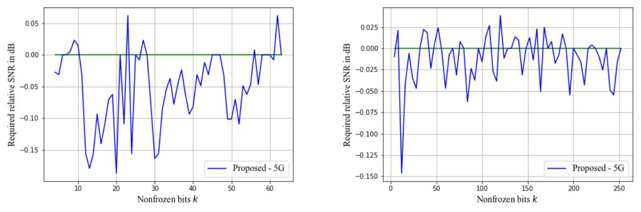
as 1. During the optimization, the individual loss of the k-th polar code, i.e., $J_k(\{\pi_j\}_{j=1}^{k})$ in (2), is set as the BLER on the log scale with the SC list size is $L_{\text{sc}} = 8$ under the additive white gaussian noise channel [21]. To ensure an accurate BLER estimation, at least 1000 block error events are collected for each polar code. Moreover, to ensure that the loss $J_k(\{\pi_j\}_{j=1}^{k})$ with different k are comparable, the decoding SNR for code construction is set to the values under which 5G nested polar codes [7] approach BLER $= 0.01$. In particular, the SNR is given by Es/N0 under QPSK. When the code length $N = 64$ and 256, respectively, the SNR values for different k are illustrated in Fig. 3. It can be seen that, to achieve the same BLER, the required SNR of 5G nested polar codes is increased with the increase of k for both cases.

After the polar codes are constructed under the above setting, we first evaluate the required SNR for approaching the target BLER $= 0.01$ under SCL-Genie decoder, which selects the correct path as long as it is among the surviving paths. For comparison, we also show the performance of DE/GA [3], 5G nested polar codes [7], and the MLP-based design [16], respectively. Both [16] and this work adopt neural networks for code construction. Moreover, both neural networks are trained with the aid of the code construction performance without labels, and hence they both belong to reinforcement learning rather than supervised learning. We term the approach in [16] as MLP to distinguish it from the customized design of neural network based on set representations and attention mechanism. By centralizing the required SNR of 5G nested polar codes (as shown in Fig. 3) as zero, the required relative SNR between each scheme and 5G nested polar codes for $N = 64$ when BLER $= 0.01$ is compared in Fig. 4. As



(a) $N = 64$.   (b) $N = 256$.

Fig. 6. Required relative SNR between the proposed codes and 5G nested polar codes when BLER $= 10^{-5}$ under SCL decoder.



(a) $N = 64$.   (b) $N = 256$.

Fig. 7. Required relative SNR between the proposed codes and 5G nested polar codes with CRC when BLER $= 10^{-5}$ under SCL decoder.

shown in Fig. 4, compared with 5G nested polar codes, DE/GA results in a worse overall performance, with the largest loss 0.1 dB. In contrast, both AI-based constructions achieve better performance than 5G nested polar codes. Moreover, the proposed code construction using attention-based set-to-element model performs better than the MLP-based design for almost all different k. This is because the proposed construction fully incorporates the nested structures of the codes into the neural network architecture by exploiting set representations and attention mechanism.

Next, we compare the performance of different schemes for $N = 256$ in Fig. 5 under SCL-Genie decoder. Due to the extremely long construction time, the result of the MLP-based design is not obtained in reasonable time, and hence omitted. Algorithm 1 is aided with the divide-and-conquer strategy proposed in Section IV. Specifically, we adopt 5G nested polar codes as the initial solution, and equally divide the index sequence into 8 segments for parallel implementation. The corresponding result is termed as Proposed_1. Moreover, the index sequence of Proposed_1 is further divided into 9 segments, where the first and last segments consist of 16 elements, respectively, and each of the middle 7 segments consists of 32 elements. After the optimization of each segment, the result is termed as Proposed_2. As shown in Fig. 5, DE/GA still performs the worst, while both Proposed_1 and Proposed_2 achieve a maximum gain 0.2 dB better performance than 5G nested polar codes, with , and

Finally, we compare the proposed codes with 5G nested polar codes at low BLER under the conventional SCL decoder. Figure 6 shows that, when BLER $= 10^{-5}$, the advantage of the proposed codes is more evident especially for $N = 256$. This demonstrates the superiority of the proposed codes at low BLER. We also compare the performance when CRC is used, where the CRC lengths are 4 and 8 for $N = 64$ and $N = 256$, and the CRC polynomials are selected as 0x3 and 0xD5, respectively. We can see from Fig. 7 that the proposed codes can still achieve better performance than 5G nested polar codes, with a maximum gain over 0.15 dB.

VI. Conclusion

In this paper, we proposed a learning based construction for nested polar codes. We first transformed the original combinatorial optimization problem into a parameterized policy optimization problem for sequential decision. To incorporate the nested structure of the codes into the policy design, we proposed an attention-based set-to-element model by adopting appropriate set representations and attention mechanism. Based on the designed architecture for the policy, a gradient based algorithm for code construction and a divide-and-conquer strategy for parallel implementation were further developed. Simulation results demonstrated that the proposed construction outperforms the state-of-the-art nested polar codes for SCL decoding.

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, Jul. 2009.
[2] R. Mori and T. Tanaka, “Performance of polar codes with the construction using density evolution,” *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 519–521, Jul. 2009.
[3] P. Trifonov, “Efficient design and decoding of polar codes,” *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3221–3227, Nov. 2012.
[4] I. Tal and A. Vardy, “How to construct polar codes,” *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.
[5] *Polar Code Design and Rate Matching*, document R1-167209, 3GPP TSG RAN WG1 Meeting ♯86, Gothenburg, Sweden, 2016.
[6] G. He, J.-C. Belfiore, I. Land, G. Yang, et al., “β-expansion: A theoretical framework for fast and recursive construction of polar codes,” in *IEEE GLOBECOM*, 2017.
[7] 3GPP, “Multiplexing and channel coding,” 3GPP 38.212.Release 15.
[8] S. A. Hashemi, M. Mondelli, S. H. Hassani, C. Condo, et al., “Decoder partitioning: Towards practical list decoding of polar codes,” *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 3749–3759, 2018.
[9] B. Li, H. Shen, and D. Tse, “A RM-polar codes,” *arXiv:1407.5483*, Jul. 2014. [Online]. Available: https://arxiv.org/abs/1407.5483.
[10] M. Mondelli, S. H. Hassani, and R. L. Urbanke, “From polar to Reed-Muller codes: A technique to improve the finite-length performance,” *IEEE Trans. Commun.*, vol. 62, no. 9, pp. 3084–3091, Sep. 2014.
[11] M. Qin, J. Guo, A. Bhatia, A. G. I. Faàbregas, and P. Siegel, “Polar code constructions based on LLR evolution,” *IEEE Commun. Lett.*, vol. 21, no. 6, pp. 1221–1224, Jun. 2017.
[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.
[13] A. Elkelesh, M. Ebada, S. Cammerer, and S. T. Brink, “Decoder-tailored polar code design using the genetic algorithm,” *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 4521–4534, Jul. 2019.
[14] Y. Liao, S. A. Hashemi, J. Cioffi, and A. Goldsmith, “Construction of polar codes with reinforcement learning,” in *IEEE GLOBECOM*, 2020.
[15] M. Ebada, S. Cammerer, A. Elkelesh, and S. ten Brink, “Deep learning-based polar code design,” in *Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2019.
[16] L. Huang, H. Zhang, R. Li, Y. Ge, and J. Wang, “AI coding: Learning to construct error correction codes,” *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 26–39, Jan. 2020.
[17] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J Smola, “Deep sets,” in *Advances in Neural Information Processing Systems (NIPS)*, 2017.
[18] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *International Conference on Learning Representations (ICLR)*, 2015.
[19] Y. Bengio, A. Lodi, and A. Prouvost, “Machine learning for combinatorial optimization: A methodological tour d’horizon,” *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.
[20] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, 2014.
[21] I. Tal and A. Vardy, “List decoding of polar codes,” *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.