An Area-Efficient Routing Solution for Automorphism Ensemble Decoding of Polar Codes

Jiajie Li, Huayi Zhou, Ryan Seah, Marwan Jalaleddine and Warren J. Gross

***Abstract*—Implementing automorphisms requires hardware for routing, and it leads to a large area overhead, especially when the number of automorphisms is large. We propose methods for selecting automorphisms in successive cancellation (AED-SC) decoders to reduce the number of routes. We established the equivalence between the automorphism selection and the NP-hard minimum K-union (MKU) problem. To maintain decoding performance under the selected automorphisms, the equivalent class property of the AED-SC is formulated as a quadratic constraint. Also, by incorporating the Hamming distance-based selection method, the problem size of the MKU problem with the quadratic constraint is reduced while enhancing decoding performance. Two heuristics are proposed: a sequential algorithm based on the least expanding set problem, and a greedy algorithm with a complexity that grows linearly with the problem size. Up to a 65% reduction in the number of routes is returned by proposed methods when compared to the state-of-the-art result, while the returned automorphism sets yield comparable decoding performance to the randomly selected set. Synthesis is performed for the routing unit for polar codes with a code length $n = 128$, a dimension $k = 60$, and 63 automorphisms that require routing, and these synthesis results verify that a 59% reduction in the number of routes reduces 57% of the logic area required by the routes, which leads to an overall area reduction of 25%.**

***Index Terms*—automorphism ensemble decoding, optimization, polar codes, routing, successive cancellation decoder**

I. Introduction

Polar codes with the successive cancellation (SC) decoder are a low-complexity encoding and decoding scheme with the proven asymptotic capacity achieving property (i.e., achieving the channel capacity when the code length goes to infinity) [1]. However, for short- to medium-length polar codes, the SC decoder suffers from a loss in error-correction performance, and this loss can be compensated by using a successive cancellation list (SCL) decoder [2]. Polar codes have been adopted by the 5th generation communication standard [3].

Although the SCL decoder and the CRC-aided (CA) SCL decoder achieve superior decoding performance for short- to medium-length polar codes, extra hardware, such as the sorting unit, is required by the SCL decoder [2], which increases the area for the hardware implementation [4]. In addition, the serial nature and the sorting operations performed during the decoding of SCL decoders contribute to the high decoding latency and the low throughput [4], [5].

Permutation-based decoding, which does not require sorting during the decoding, is another way to improve the decoding performance of SC decoders. It is shown in [6] that the decoding can be performed on trellises produced by the shuffled encoding stages, these trellises can also be realized by permuting messages and codewords in the original factor graph [7], and factor graph permutations can be generated by multiplying the permutation matrix to binary representations of all indices [7]. These permutations are called factor graph permutations, and they have been shown to improve the error correction performance of polar codes [7]–[9]. However, factor graph permutations might change information bit positions, and decoders with factor graph permutations are observed to have relatively inferior error correction performance compared to the SCL decoder [7]–[9].

A new permutation-based decoding, automorphism ensembles decoding (AED), was recently proposed for polar codes [10]. It achieves decoding performance comparable to the SCL decoders when decoding polar codes with short-to-medium code lengths [10]. In AED, the decoder permutes the (received) codewords based on the code automorphisms. Automorphisms are permutations that map a codeword to another codeword in the same codebook, and they are described by the affine transformation; all such automorphisms constitute the code's affine automorphism group. Consequently, the positions of the information bits remain unchanged under automorphisms. It is first shown in [11] that the affine automorphism group is at least the lower triangular affine group. A larger affine automorphism group, block lower triangular affine group (BLTA), is found in [10], which has been later proved as the complete affine automorphism group for polar codes [12].

Multiple permuted codewords are decoded independently and in parallel by the constituent decoder. For example, using the SC decoder as the constituent decoder (automorphism ensembles decoding with successive cancellation constituent decoder (AED-SC)) results in multiple SC decoders operating in parallel. A hardware implementation of AED-SC [13] demonstrated lower latency, higher throughput, and better energy and area efficiency than a state-of-the-art SCL decoder implementation [5]. Also, in [13], the AED-SC has a similar frame error rate (FER) to the SCL decoder when decoding a length $n = 128$ and a dimension $k = 60$ polar codes. However, the AED-SC is targeting polar codes with short-to-medium code lengths because polar codes asymptotically do not have many automorphisms [14]. Moreover, the reliability sequence of polar codes is modified to make polar codes suitable for the

Jiajie Li, Ryan Seah, Marwan Jalaleddine and Warren J. Gross are with the Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, H3A 0E9, Canada (e-mail: jiajie.li@mail.mcgill.ca; ryan.seah@mail.mcgill.ca; marwan.jalaleddine@mail.mcgill.ca; warren.gross@mcgill.ca).

Huayi Zhou was in the Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, H3A 0E9, Canada. He is now with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210018, China (e-mail: huayi.zhou@mail.mcgill.ca).

AED-SC, and these modifications might degrade the decoding performance under SC and SCL decoding [10], [15].

To further improve AED-SC performance, the automorphisms should be carefully selected [16], [17]. It has been shown in [18] that the output of the SC decoder is invariant among certain automorphisms, such as automorphisms in the low triangular affine group (the absorption group). The complete absorption group for the SC decoder is also a BLTA [19]. Beyond the absorption group, there are groups of automorphisms that return the same result under the SC decoder, these groups, which are the right coset of the absorption group, are called the equivalence class (EC) [16], [17].

II. Related work and Contributions

On the implementation side, routes are used to implement automorphisms, and implementing a large number of automorphisms can incur a significant area overhead. It was shown in [20] that factor graph permutations can be implemented using a set of basic routes, where these routes are used to generate all required routing configurations for factor graph permutations. It is shown in [20] that a set of routes is enough to implement all possible factor graph permutations for polar codes; hence, the number of required routing blocks is fewer than the number of permutations that are going to be implemented, which translates to a significant reduction in the area of the implementation. Additionally, only a constant overhead is needed regardless of whether the number of implemented factor graph permutations is 8 or 32 [20]. For a length-1024 polar code and the 28-nm technology, the permutation unit with 9 routes takes an area of 0.076 mm^2, and the implementation of the belief propagation decoder for polar codes takes an area of 0.87 mm^2 [20]. Using basic route optimizations still consumes 9% of the hardware area [20], and the area will be larger if no optimization - such as using the Beneš network [20], [21] - is used.

A routing unit with route-sharing enabled by independent sub-matrices in the BLTA is proposed for AED-SC [22], but it searches in the automorphism subsets with specific structures. The authors did not prove that these structures exist in a large set of ECs for AED-SC, hence the applicability of this method on a wide range of codes with different lengths and rates is not yet verified. Also, an efficient construction method for the routing network is not provided [22], and efficiently finding a sharing solution will be a problem when the number of possible automorphisms is large. Besides, in [22], it is also shown that the direct implementation of the routing unit uses about $2\times$ more area than the AED-SC with an optimized routing solution under the 3-nm technology, so the direct implementation has more than $2\times$ the area of the SC decoder implementation. The hardware synthesis results in [22] show that it is necessary to reduce the area used by the routing unit for area-sensitive applications.

Our work proposes an efficient method to construct a novel routing solution to support all possible automorphisms of polar codes with SC decoding. We extend the area-efficient implementation [20] to the affine automorphism group of polar codes with SC decoding, and we create a scheme that uses fewer routes than the naive implementation. In this work, we are targeting to reduce the hardware area needed to implement the AED-SC inspired by the area reduction due to the reduced number of routes in [20].

Part of this work has been submitted to the 2025 International Symposium on Topics in Coding [23], where the idea of basic routes for the factor graph permutations is generalized to the affine automorphism group, and the mapping of the PUL decomposition of an affine transformation matrix to these routes is proposed. Then, we demonstrate in [23] that selecting an automorphism set that can be realized with a small number of routes is equivalent to solving a minimum K-union (MKU) problem, and we introduce a quadratic constraint to ensure that the selected automorphisms come from different ECs in the MKU formulation. This work extends on [23] and proposes the following novel contributions:

1) We propose an algorithm for generating basic routes and a control algorithm for scheduling basic routes. Additionally, we propose a scheme that combines the Hamming distance (HD)-based EC selection to reduce the problem size of the MKU problem with the quadratic constraint while further enhancing the decoding performance.
2) We propose a sequential quadratic constraint linear programming algorithm to improve the solvability of the MKU problem with the quadratic constraint. The proposed sequential algorithm can find automorphism sets that require the same number of routes as the sets generated by the commercial solver while requiring orders of magnitude fewer numerical evaluations. Additionally, we propose a greedy heuristic with a linear complexity relative to the problem size. The solutions obtained using this heuristic are identical to those produced by the commercial solver.
3) We achieve up to a 65% reduction in the number of routes from our proposed methods compared to the state-of-the-art results in [13] while having similar decoding performance to the randomly selected automorphism set. We also present hardware synthesis results on polar codes with $n = 128$ and $k = 60$, which show a 59% reduction in the number of routes. This translates to a 57% reduction in the logic area required by the routes leading to an overall area reduction of 25%.

This work is structured as follows. Section II explains the motivation of this work. Section III provides background knowledge of polar codes, AED-SC, and the MKU problem. Section IV shows how to construct routes for automorphisms for SC decoders. Section V explains the method to find a set of automorphisms that can be realized by a small number of routes. Section V also explains the combined scheme that selects a set of automorphisms that preserves the superior decoding performance of the AED-SC while it can be implemented by a small number of routes. Section VI shows the proposed sequential and greedy algorithms for solving the MKU with the quadratic constraint. Section VII shows the experiment results, and Section VIII concludes this work.

III. Preliminaries

In this section, we will present the notations and the background knowledge related to this work, where polar codes, AED-SC and the MKU problem are introduced.

A. Notations

The matrix transpose operation is denoted by $\top$. The Boolean complement is denoted by $\overline{\ }$. The Kronecker product of a matrix is denoted by $\otimes$. The cardinality of a set and the absolute value are denoted by $|\cdot|$, and they will be specified in the text. Bold upper case letters (e.g., $\boldsymbol{M}$) and bold lower case letters (e.g., $\boldsymbol{m}$) denote matrices and vectors unless specified explicitly. The index in binary vector representation is denoted as $\boldsymbol{z}$ with the right most-significant bit (most-significant bit-last when counting from left to right).

B. Polar Codes

The generator matrix of length-n polar codes $\boldsymbol{c}$ is constructed through two steps: I), construct a $n \times n$ matrix $\boldsymbol{G}$ through Kronecker power,

$$\boldsymbol{G} = \boldsymbol{F}^{\otimes m},\ \boldsymbol{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \tag{1}$$

where $m = \log_2(n)$; II), selecting k most reliable channels in $\boldsymbol{G}$ to place the length-k message vector, positions of these k rows are called information bit positions ($\mathcal{I}$), and all other positions, frozen bit positions ($\mathcal{F}$), will be placed with the bit zero. Different types of measures are used to construct polar codes for different types of channels.

Instead of measuring the channel reliability, polar codes can be viewed as monomial codes like RM codes [11]. There is a universial partial order (UPO) (i.e., regardless of the channel types) [11], [24] on the row indices of $\boldsymbol{G}$/monomials, and polar codes, which have the information set that is equivalent to the set of monomials ordered by the UPO [11], are also called decreasing monomial codes.

Similar to prior work [10], [16], [17], [19], instead of computing reliability measures for all n synthetic channels and selecting the k synthetic channels with k highest reliability, we firstly select one or more monomials, which are called generating monomials $\mathcal{I}_{min}$, and constructing the information set by including all monomials that are more reliable than $\mathcal{I}_{min}$ in the UPO. In this work, the construction of the UPO is implemented according to the method explained in [17].

C. AED with SC Constituent Decoders

An automorphism is a permutation (π) that maps a codeword to another codeword in the same codebook, and the set of all these permutations is called the automorphism group [10], [16]. The row of the generator matrix can be indexed by an integer in the binary representation $\boldsymbol{z}$ or its monomial form

$$\mathcal{M} = \prod_{\{i|z_i=0, i\in\{0,\dots,n-1\}\}} e_i,$$

where $z_i \in \boldsymbol{z}$ and e_i denotes a monomial.

For monomial codes like polar codes, the encoding process can be described by evaluating an m-variate polynomial

$$f(e_0, \dots, e_{m-1}),$$

and the codeword $\boldsymbol{c}$ can be computed by evaluating f for all elements in $\mathbb{F}_2^m$

$$\boldsymbol{c} = (f(n-1), \dots, f(0)) = (c_0, \dots, c_{n-1}),$$

where $c_i \in \boldsymbol{c}$ is a code bit.

The polar codes' automorphisms, which can be represented by the affine transformation, have the following form

$$\boldsymbol{z}' = \overline{(\boldsymbol{A}\boldsymbol{e} + \boldsymbol{b})}, \tag{2}$$

where $\boldsymbol{e} = \overline{\boldsymbol{z}}$ is a m-variate element of the monomial, $\boldsymbol{z}$ is the index of the code bit in binary, and $\boldsymbol{z}'$ is the permuted code index. All of these automorphisms form the affine automorphism group. The binary transformation matrix $\boldsymbol{A}$ has a size of $m \times m$, and the offset $\boldsymbol{b}$ is an arbitrary length-m binary vector. The affine automorphism group of polar codes is in the BLTA that can be represented by matrices $\boldsymbol{A}$ with the BLTA structure $\boldsymbol{s} = (s_1, \dots, s_t)$ [12], where s_i is the size of the square matrix in the diagonal of the affine transformation matrix $\boldsymbol{A}$, and $i \in \{1, ..., t\}$. Related work [10], [13], [25] performs the affine transformation in the index for the code bit, while this work follows the convention, which uses the monomial in the affine transformation, of the related work [12], [16], [17], [19].

It is first proven that outputs of the SC decoder are invariant under the low triangular affine group [18], where $\boldsymbol{A}$ has a low triangular structure. Later, the complete affine automorphisms (described by a BLTA structure $\boldsymbol{s}_{\mathbb{1}}$), which are SC-invariant, are found in [19] and called the absorption group. Besides the absorption group, automorphisms can be classified into ECs where the same SC decoding results are yielded for each class [16], [17].

The decoding performance of the SC decoder can be improved by independently decoding M permuted codewords according to automorphisms π_i that are from different ECs

$$\hat{\boldsymbol{c}}_i = \pi_i^{-1}(\mathrm{SC}(\pi_i(\boldsymbol{l}))),$$

and then select the one that has the highest posterior probability with respect to the received log-likelihood ratio (LLR) vector $\boldsymbol{l}$ [25]:

$$\hat{\boldsymbol{c}} = \underset{i\in\{1,2,\dots,M\}}{\arg\max}\ P(\boldsymbol{l}|\hat{\boldsymbol{c}}_i).$$

According to Dumer's explanation in [26], permuting received codewords might generate evenly spread error patterns, and some patterns might be easily decoded.

D. Minimum K-Union Problem

Let a set be $\mathcal{W}$, and each element in $\mathcal{W}$ contains some elements from a set $\mathcal{V}$. It is interesting to know how to choose a subset with a size of $K \leq |\mathcal{W}|$ out of $|\mathcal{W}|$ possible elements from $\mathcal{W}$ such that the size of the union of elements in these K subsets of $\mathcal{V}$ is reduced, where $|\cdot|$ denotes the cardinality of a set. This problem is called the MKU problem, which

is NP-hard [27], and a bipartite graph representation can be generated for the MKU problem. The bipartite graph consists of two groups of nodes, one group has $|\mathcal{W}|$ nodes, which represent elements $w \in \mathcal{W}$ and are named as the hyperedges (i.e., subset of vertexes) in the literature, and the other group has $|\mathcal{V}|$ nodes, which represent elements $v \in \mathcal{V}$ and are referred as the vertex in the literature. Connections ($\mathcal{E}$) are only formed by nodes from different groups

$$(w, v) \in \mathcal{E}, w \in \mathcal{W}, v \in \mathcal{V}.$$

The MKU problem can be formulated as an integer programming problem as following [28]:

$$\min \sum_{v \in \mathcal{V}} x_v, \tag{3a}$$

$$\text{s.t.} \sum_{w \in \mathcal{W}} x_w = K, \tag{3b}$$

$$x_v \geq x_w \ \forall (w, v) \in \mathcal{E}, w \in \mathcal{W}, v \in \mathcal{V}, \tag{3c}$$

$$x_w,\, x_v \in \{0, 1\} \ \forall w,\, v, \tag{3d}$$

where x_w and $x_v = 1$ in (3d) means the element $w \in \mathcal{W}$ and the element $v \in \mathcal{V}$ are chosen, and x_w and $x_v = 0$ otherwise. The objective function (3a) measures how many elements v are included in the union. The equality constraint (3b) is usually written as an inequality constraint $\sum_{w \in \mathcal{W}} x_w \geq K$ in literature [28] because the objective function (3a) is non-decreasing and the solution with a size of K usually has the smallest union size. We choose to use this equality constraint (3b) because we want to ensure only K elements $w \in \mathcal{W}$ are chosen for the ease of the implementation. The inequality constraints (3c) ensure all indicator variables of v in a chosen w are set to 1.

IV. Routes for automorphisms under the SC decoder

In this section, we show the mapping of the PUL decomposition of the affine transformation matrix to routes. Then, we established the link between the basic routes for factor graph permutations and the permutation matrix derived from the PUL decomposition.

A. PUL Decomposition of Affine Transformations

It is shown in [16], [17] that there exists non-unique PUL decompositions for the transformation matrix $\boldsymbol{A}$, and the affine transformation function (2) can be rewritten as

$$\boldsymbol{A}\boldsymbol{e} + \boldsymbol{b} = \boldsymbol{P} \cdot \boldsymbol{U} \cdot (\boldsymbol{L}\boldsymbol{e} + \boldsymbol{b}_0) \tag{4}$$

where $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{U}\boldsymbol{L}$, $\boldsymbol{b} = \boldsymbol{P}\boldsymbol{U}\boldsymbol{b}_0$, $\boldsymbol{P}$ is the permutation matrix, $\boldsymbol{U}$ is the binary upper triangular matrix, and $\boldsymbol{L}$ is the binary lower triangular matrix. This PUL decomposition implies that the automorphism described by the affine transformation can be viewed as the concatenation of permutations

$$\pi_{(\boldsymbol{A}, \boldsymbol{b})} = \pi_{\boldsymbol{L}, \boldsymbol{b}_0} \circ \pi_{\boldsymbol{U}} \circ \pi_{\boldsymbol{P}} \tag{5}$$

from right to left [17], where $\circ$ denotes the concatenation of the permutations, $\pi_{\boldsymbol{L}, \boldsymbol{b}_0}$ denotes the permutation formed by the transformation matrix $\boldsymbol{L}$ and the offset $\boldsymbol{b}_0$, $\pi_{\boldsymbol{U}}$ denotes the permutation formed by the transformation matrix $\boldsymbol{U}$ and a zero offset, and $\pi_{\boldsymbol{P}}$ denotes the permutation formed by the transformation matrix $\boldsymbol{P}$ and a zero offset. Since the SC decoder is invariant under the lower triangular affine transformation $\pi_{\boldsymbol{L}, \boldsymbol{b}_0}$ [18], the affine transformation can be reduced to

$$\boldsymbol{P} \cdot \boldsymbol{U} \cdot \boldsymbol{e}. \tag{6}$$

It is shown in [17] that mixing $\boldsymbol{P}$ and $\boldsymbol{U}$ with the block diagonal structure is sufficient to find all ECs. Hence, finding routes for $\boldsymbol{P}$ and $\boldsymbol{U}$ with the block diagonal structure is sufficient to generate routing for automorphisms formed by the affine transformation (4).

B. Routes for $\boldsymbol{U}$

By associativity, equation (6) can be rewritten to

$$\boldsymbol{P} \cdot \boldsymbol{U} \cdot \boldsymbol{e} = \boldsymbol{P}\,(\boldsymbol{U} \cdot \boldsymbol{e}) = \boldsymbol{P} \cdot \boldsymbol{e}^{'}. \tag{7}$$

where $\boldsymbol{U} \cdot \boldsymbol{e} = \boldsymbol{e}^{'} \in \{0, 1\}^m$. Hence, let $\boldsymbol{E}$ be a $m \times n$ matrix with $\boldsymbol{e}_i^{\top}$ as the column vector where $\boldsymbol{e}_i \in \mathbb{F}_2^m$ is the m-variate element of the monomial. Define $\boldsymbol{E}^{'}$ as

$$\boldsymbol{E}^{'} = \boldsymbol{U} \cdot \boldsymbol{E}, \tag{8}$$

where the set of column vectors produced by the affine transformation

$$\left\{\boldsymbol{e}_i^{'} | \boldsymbol{e}_i^{'} \in \{0, 1\}^m,\, \boldsymbol{e}_i^{'\top} \in \boldsymbol{E}^{'}\right\} = \left\{\boldsymbol{e}_i | \boldsymbol{e}_i \in \{0, 1\}^m,\, \boldsymbol{e}_i^{\top} \in \boldsymbol{E}\right\},$$

and only the ordering of column vectors in $\boldsymbol{E}^{'}$ is different from the ordering in $\boldsymbol{E}$ [17]. Hence, $\boldsymbol{E}^{'}$ is equivalent to $\boldsymbol{E}$ after a column permutation according to a permutation vector $\pi_{\boldsymbol{U}}$. Based on this fact, the corresponding routing matrix for $\boldsymbol{U}$ is a $n \times n$ matrix with entries ones in the rows $\boldsymbol{z} = \overline{\boldsymbol{e}}$ and the columns $\boldsymbol{z}^{'} = \overline{(\boldsymbol{U}\boldsymbol{e})}$ for all $\boldsymbol{z} \in \{0, 1\}^m$. All other entries in the routing matrix are zeros.

An example of computing the automorphism for $\boldsymbol{U}$ is given in the following.

Example 1: Let the affine transformation be $\boldsymbol{Z}^{'} = \overline{(\boldsymbol{U} \cdot \boldsymbol{E})}$, where $\boldsymbol{U} = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$ and

$$\boldsymbol{E} = \overline{\boldsymbol{Z}} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

Then, we have

$$\overline{(\boldsymbol{U} \cdot \boldsymbol{E})} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

and the permutation in the integer form is $[1, 0, 2, 3]$.

C. Routes for $\boldsymbol{P}$

There are $m!$ possible factor graph permutations for polar codes [6], and a change of the factor graph order is equivalent to a change in the order of the bit index [7]. Since the affine transformation for the permutation matrix can be viewed as changing the order of bits in $\boldsymbol{e}$, it is sufficient to work on the binary index $\boldsymbol{z}$ and perform the affine transformation. An example of computing the automorphism for $\boldsymbol{P}$ is given in the following.

Example 2: Let the affine transformation be $\boldsymbol{Z}^{'} = \overline{(\boldsymbol{P} \cdot \boldsymbol{E})}$, where $\boldsymbol{P} = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ and

$$\boldsymbol{E} = \overline{\boldsymbol{Z}} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

Then, we have

$$\overline{(\boldsymbol{P} \cdot \boldsymbol{E})} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \boldsymbol{P} \cdot \boldsymbol{Z},$$

and the permutation in the integer form is $[0, 2, 1, 3]$.

It is also proved in [20] that routes for all factor graph permutations can be realized by combining $m-1$ basic routes. Hence, we conjecture that the routing (i.e., grouping several routes together) method proposed in [20] can be used to realize all affine transformations with the transformation matrices $\boldsymbol{A}$ being the permutation matrix and a zero offset. The following theorem shows that automorphisms derived from the affine transformation using the transformation matrix $\boldsymbol{P}$ with block diagonal structure ($\boldsymbol{z}^{'} = \boldsymbol{P}\boldsymbol{z}$) are a subset of the factor graph permutations.

Theorem 1. *Record indices of entries* 1 *in a* $m \times m$ *permutation matrix* $\boldsymbol{P}$ *with a block diagonal structure* $\boldsymbol{s}$*. Let row indices* j*s of entries* 1 *be indices for a permutation vector* π*, and column indices of entries* 1 *be elements in corresponding position* j*s of* π*. Denote the set of permutations corresponding to* $\boldsymbol{P}$ *as*

$$\mathcal{D} = \{\pi | \pi\left(\mathcal{I}_i^{\pi}(s_i)\right) = \mathcal{I}_i^{\pi}(s_i)\ \forall s_i \in \boldsymbol{s}\}, \tag{9}$$

where $\mathcal{I}_i^{\pi}(s_i) = \{\gamma_i, \ldots, \gamma_i + s_i - 1\}$*, and* γ_i *is the starting index for the block* s_i*.* $\mathcal{D}$ *is a subgroup of the permutation group* $\mathcal{P}$ *on the set of elements* $\{1, 2, \ldots, m\}$*.*

Proof. (I) The square matrix corresponding to a $s_i \in \boldsymbol{s}$ defines a permutation on the index set $\mathcal{I}_i^{\pi}(s_i)$ within the block defined by the indices $\gamma_i, \ldots, \gamma_i + s_i - 1$, indices in $\{1, 2, \ldots, m\} \backslash \mathcal{I}_i^{\pi}(s_i)$ are randomly shuffled outside the block, and we denote this permutation as π_i. Hence, the set of permutations π_i corresponding to the block s_i can be written as

$$\mathcal{D}_i = \{\pi_i | \pi\left(\mathcal{I}_i^{\pi}(s_i)\right) = \mathcal{I}_i^{\pi}(s_i)\}.$$

(II) Since the set $\mathcal{I}_i^{\pi}(s_i)$ is shuffled within the block according to π_i, and the group $\mathcal{P}$ is transitive because there exists a $\boldsymbol{P}$ such that $\pi_{\boldsymbol{P}}(x) = y$ for every x, $y \in \{1, 2, \ldots, m\}$, it can be concluded that $\mathcal{D}_i$ is a subgroup of $\mathcal{P}$ [29].
(III) For the given block diagonal structure $\boldsymbol{s}$, the corresponding set of permutations should have

$$\pi\left(\mathcal{I}_i^{\pi}(s_i)\right) = \mathcal{I}_i^{\pi}(s_i),\ \forall s_i \in \boldsymbol{s},$$

which can be formed by the intersection of all the sets, $\mathcal{D}_i$,

$$\bigcap_{\forall s_i \in \boldsymbol{s}} \mathcal{D}_i = \{\pi | \pi\left(\mathcal{I}_i^{\pi}(s_i)\right) = \mathcal{I}_i^{\pi}(s_i),\ \forall s_i \in \boldsymbol{s}\} = \mathcal{D}.$$

(IV) As $\mathcal{D}_i \subseteq \mathcal{P}$ and $\mathcal{D}$ is the intersection of all $\mathcal{D}_i$, $\mathcal{D} \subseteq \mathcal{P}$. □

Algorithm 1: FindSubroutingBLTA

Input: BLTA structure $\boldsymbol{s}$, The set of basic routes $\mathcal{V}_{set}$
Output: The subset of selected basic routes $\mathcal{V}_{subset}$, Indexes for basic routes $subroute\_idx$

```
num_blocks ← Size(s, 2)        /* Find the number of blocks in the BLTA. */
γ_vec ← FindBlockStartIdxes(s)
num_subroute ← Sum(s − 1) /* Find starting indexes for each block. */
subroute_idx ← 0
counter_subset ← 1
for i = 1 : num_blocks do
    s ← s(i)     /* Fetch the size of one block. */
    if s ≠ 1 then
        γ ← γ_vec(i)
        start_idx ← γ
        end_idx ← γ + s − 2
        for j = start_idx : end_idx do
            V_subset(counter_subset) ← V_set(j)  /* Record the routes for corresponding stages in the factor graph. */
            subroute_idx(j) ← counter_subset    /* Index the recorded route. */
            counter_subset = counter_subset + 1
return V_subset, counter
```

Based on Theorem 1 and [20, Thm. 1], we can conclude that automorphisms derived from the affine transformation using the transformation matrix $\boldsymbol{P}$ with block diagonal structure can be realized by a subset of $m-1$ routes [20]. Only basic routes corresponding to change indices within each block are needed to realize all $\boldsymbol{P} \in \text{BLTA}(\boldsymbol{s})$. We propose Algorithm 1 to select the subset $\mathcal{V}_{subset}$ of the routes corresponding to $\boldsymbol{P} \in \text{BLTA}(\boldsymbol{s})$ given the block diagonal structure $\boldsymbol{s}$ and the $m-1$ routes for implementing all possible factor graph permutations.

In [20], routes are scheduled by [20, Alg. 2] and [20, Alg. 4] to achieve the target routing. In this work, we modify the [20, Alg. 2] and [20, Alg. 4] algorithms to generate the control signals of the routing schedule for the $\boldsymbol{P} \in \text{BLTA}(\boldsymbol{s})$, and the pseudo-code is shown in Algorithm 2 and 3. The permutation algorithm [20, Alg. 3] is included for being self-contained, and the pseudo-code is shown in Algorithm 4. The set of indices $\boldsymbol{R}_0$ is routed to their appropriate places using Algorithm 3.

D. Routes for $\boldsymbol{P} \cdot \boldsymbol{U}$

According to [17], the corresponding concatenation of permutations of equation (6) is

$$\pi = \pi_{\boldsymbol{U}} \circ \pi_{\boldsymbol{P}}. \tag{10}$$

Hence, the corresponding route is equivalent to first passing the input through the routes of $\boldsymbol{P}$ and then passing through the routes of $\boldsymbol{U}$. An example of this concatenated routing is shown in Fig. 1 and example 3 with the following affine transformation matrix

$$\boldsymbol{P} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \text{ and } \boldsymbol{U} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Example 3: Let the affine transformation be $\boldsymbol{Z}^{'} =$

Algorithm 2: SubRoutingBLTA

Input: Input indexes $\boldsymbol{R}_0$, The subset of selected basic routes $\mathcal{V}_{subset}$, The index of the permuted stage in the factor graph p, The index of the stage in the factor graph e, Indexes for basic routes $subroute\_idx$
Output: Permuted indexes $\boldsymbol{R}_0$

if $p < e$ **then**
  for $j = p : 1 : (e-1)$ **do**
    $i \leftarrow subroute\_idx(j)$ /* Get the index of the corresponding route. */
    $\boldsymbol{R}_0 \leftarrow \boldsymbol{R}_0 * \mathcal{V}_{subset}(i)$ /* Permute indexes corresponding to the route. */
else if $p > e$ **then**
  for $j = (p-1) : -1 : e$ **do**
    $i \leftarrow subroute\_idx(j)$ /* Get the index of the corresponding route. */
    $\boldsymbol{R}_0 \leftarrow \boldsymbol{R}_0 * \mathcal{V}_{subset}(i)$ /* Permute indexes corresponding to the route. */
return $\boldsymbol{R}_0$

Algorithm 3: Permutation Generation by A Matrix Decomposition

Input: Input indexes $\boldsymbol{R}_0$, Indexes of permuted stages in the factor graph PFG $= [\pi^0, \pi^1, ..., \pi^{m-1}]$, Indexes of stages in the factor graph OFG $= [0, 1, ..., m-1]$, BLTA structure $\boldsymbol{s}$, The subset of selected basic routes $\mathcal{V}_{subset}$, Indexes for basic routes $\boldsymbol{i}$
Output: Permuted indexes $\boldsymbol{R}_0$

$\boldsymbol{p} \leftarrow \boldsymbol{0}$ /* Initialize vector for storing indexes of stages in the factor graph. */
for $i = 1 : m$ **do**
  $p \leftarrow$ PFG(i) /* The index of the stage in the permuted factor graph. */
  $e \leftarrow$ OFG(i) /* The index of the stage in the original factor graph. */
  for $j = i : m$ **do**
    PFG$(j) =$ updateStage(PFG$(j), p, e)$ /* Update the index of the stage for the newly permuted factor graph. */
  $\boldsymbol{p}(i) = p$ /* Record index of the stage in the permuted factor graph. */
$Blocks \leftarrow \{i | s_i \in \boldsymbol{s} \wedge s_i > 1\}$ /* Find indexes of BLTA blocks that require routing. */
$num\_blocks \leftarrow |Blocks|$ /* Compute how many BLTA blocks that require routing. */
$\gamma_{vec} \leftarrow$ FindBlockStartIdxes$(\boldsymbol{s})$
for $i = 1 : num\_blocks$ **do**
  $Block \leftarrow Blocks(i)$ /* Get the index of a block in the BLTA that requires routing. */
  $s \leftarrow \boldsymbol{s}(Block)$
  $start\_idx \leftarrow \gamma$
  $end\_idx \leftarrow \gamma + s - 1$
  for $j = start\_idx : end\_idx$ **do**
    $\boldsymbol{R}_0 \leftarrow$ SubRoutingBLTA$(\boldsymbol{R}_0, \mathcal{V}_{subset}, \boldsymbol{p}(j), \text{OFG}(j), \boldsymbol{i})$ /* Permute indexes according to the selected route. */
return $\boldsymbol{R}_0$

$\overline{(\boldsymbol{P} \cdot \boldsymbol{U} \cdot \boldsymbol{E})}$, and $\boldsymbol{U} \cdot \boldsymbol{E} = \left[\begin{smallmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{smallmatrix}\right]$. Then

$$\boldsymbol{Z}^{'} = \overline{\boldsymbol{P} \cdot \boldsymbol{U} \cdot \boldsymbol{E}} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

and the permutation in the integer form is $[2, 0, 1, 3]$.

Algorithm 4: updateStage [20]

Input: The initial index of the permuted stage in the factor graph π^{in}, The index of the permuted stage in the factor graph p, The index of the stage in the factor graph e
Output: The new index of the permuted stage in the factor graph π^{out}

if $\pi^{in} == p$ **then**
  $\pi^{\text{out}} \leftarrow e$ /* [20, Lemma 2] */
else if $\pi^{in} \in [\min(p, e), \max(p, e)]$ ***and*** $p \neq e$ **then**
  $\pi^{\text{out}} \leftarrow \pi^{\text{in}} + \text{sign}(p - e)$ /* [20, Lemma 3] */
else
  $\pi^{\text{out}} \leftarrow \pi^{\text{in}}$ /* Indexes that are not affected by the permutation are kept constant. */
return π^{out}

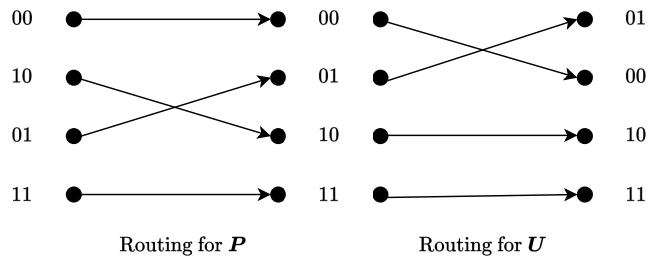


Fig. 1. An example of the implementation of the routing for $\overline{(\boldsymbol{P} \cdot \boldsymbol{U} \cdot \boldsymbol{e})}$.

V. Reduced the number of required routes

In this section, we first show the equivalence between reducing the number of routes and the MKU problem, then the constraint for the EC, and lastly the HD-based heuristic to reduce the problem size of the MKU problem.

A. MKU Problem and Reducing the Number of Routes

The equivalence between the problem of selecting the automorphism set whose implementation requires a small number of routes and the MKU problem is shown in the following.

Initialization: Since the identity permutation does not require any routes, the identity permutation is always included in the permutation set in this work.

Nodes $\mathcal{W}$: Given a polar code with a block diagonal structure $\boldsymbol{s} = (s_1, s_2, ..., s_t)$ and the block structure of the absorption group $\boldsymbol{s}_{\mathbb{1}}$, we first remove all automorphisms in the absorption group because the SC decoder will output the same result as the decoding codeword without permutations under these automorphisms. Let $\mathcal{W}$ denote the set of automorphisms after excluding automorphisms from the absorption group.

Nodes $\mathcal{V}$: Given a polar code with a block diagonal structure $\boldsymbol{s} = (s_1, s_2, \ldots, s_t)$, there are $|\mathcal{A}_{\boldsymbol{U}}| = \prod_{i=1}^{t} \sqrt{2^{s_i(s_i-1)}}$ $\boldsymbol{U}$s with the block diagonal structure $\boldsymbol{s}$ [17], where $\mathcal{A}_{\boldsymbol{U}}$ denotes the set of valid $\boldsymbol{U}$ and $|\cdot|$ denotes the size of the set. Hence, all $|\mathcal{W}|$ automorphisms can be realized by at most $|\mathcal{V}| = |\mathcal{A}_{\boldsymbol{U}}| + (m-1)$ routes. We use $\mathcal{V}$ to denote all routes required to implement all automorphisms of the given polar codes.

Edges $\mathcal{E}$: For every automorphism $w \in \mathcal{W}$, an edge $(w, v) \in \mathcal{E}$ is constructed if the route v is required to implement the automorphism w.

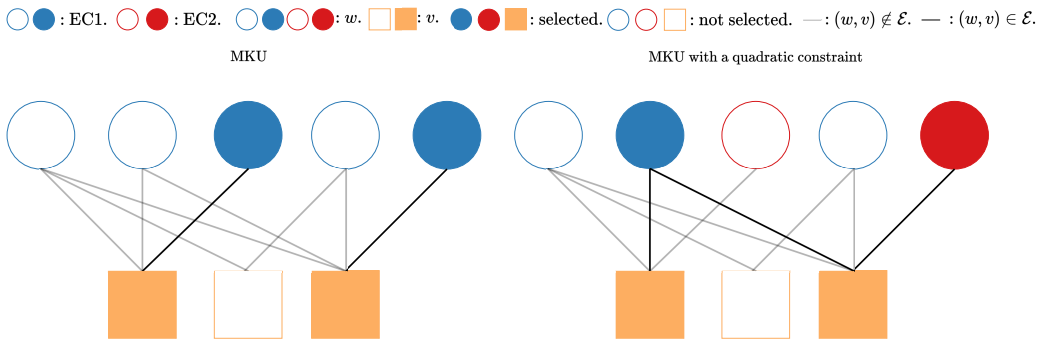


Fig. 2. The bipartite graph representation of the MKU problem (left figure) and the MKU problem with the quadratic constraint (right figure). Circle nodes in the same color are from the same EC. Filled circles and squares denote selected nodes in $\mathcal{W}$ and $\mathcal{V}$, respectively, and circles and squares are not filled otherwise. Black lines denote $(w, v) \in \mathcal{E}$ of selected w and v, and lines are grey otherwise.

MKU problem: Define an indicator vector $\boldsymbol{x}$, the first $|\mathcal{V}|$ elements $\{x_v|\forall v \in \mathcal{V}\} = \{x_1, x_2, \ldots, x_{|\mathcal{V}|}\}$ denotes routes that are used to realize a given set of automorphisms, and the last $|W|$ elements $\{x_w|\forall w \in \mathcal{W}\} = \{x_{|\mathcal{V}|+1}, x_{|\mathcal{V}|+2}, \ldots, x_{|\mathcal{V}|+|\mathcal{W}|}\}$ denotes which automorphism is selected into the set of permutations. Assume we want to select M automorphisms that can be realized by a reduced number of routes. Selecting $K = M - 1$ automorphisms that can be realized by a reduced number of routes takes the same form as equation (3).

Fig. 2 shows a bipartite graph example. In Fig. 2, the circle nodes represent the automorphisms, the square nodes represent routes, and the edges between the square nodes and circle nodes link the automorphism and its required routes together. The goal of the MKU problem is to select K automorphisms (circle nodes) such that a reduced number of routes (square nodes) is required. Selected automorphisms and required routes are filled. Fig. 2 shows an example of the MKU problem with $K = 2$. The minimum number of required routes is 2 for this MKU problem, there exist multiple solutions, and the filled nodes and black lines show one of the solutions.

B. The Quadratic Constraint for Equivalent Classes

We first remove all permutations in the absorption group, including the identity permutation, and denote the number of remaining automorphisms as $\mathcal{W}$. To remove automorphisms in the absorption group, all automorphisms whose affine transformation matrix has the BLTA structure $\boldsymbol{s}_{\mathbb{1}}$ are removed. Then, to ensure each selected automorphisms from different ECs, we define a $|\mathcal{W}| \times |\mathcal{W}|$ correlation matrix $\boldsymbol{Q}$, and the element $q_{i,j}$ denotes whether two automorphisms π_i and π_j are in the same EC or not:

$$q_{i,j} = \begin{cases} 1 & \text{if } \pi_i \text{ and } \pi_j \in \text{the same EC;} \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

We use $q_{i,j}$ as the element in the ith row and the jth column of $\boldsymbol{Q}$. A quadratic constraint that ensures selected permutations belong to different ECs can be formulated as

$$\tilde{\boldsymbol{x}}\boldsymbol{Q}\tilde{\boldsymbol{x}}^\top = K, \tag{12}$$

where $\tilde{\boldsymbol{x}} = (x_{|\mathcal{V}|+1}, x_{|\mathcal{V}|+2}, ..., x_{|\mathcal{V}|+|\mathcal{W}|})$. For ease of implementing the problem formulation in our simulation tool (i.e., MATLAB), the correlation matrix $\boldsymbol{Q}$ is padded with 0 to ensure all constraints have the same dimension:

$$\boldsymbol{Q}' = \begin{bmatrix} \boldsymbol{0}^{|\mathcal{V}|\times|\mathcal{V}|} & \boldsymbol{0}^{|\mathcal{V}|\times|\mathcal{W}|} \\ \boldsymbol{0}^{|\mathcal{W}|\times|\mathcal{V}|} & \boldsymbol{Q} \end{bmatrix}, \tag{13}$$

where $\boldsymbol{0}^{N_1\times N_2}$ denotes an all-zeros matrix with a size of $N_1 \times N_2$, and $|\cdot|$ denotes the cardinality of the set. The problem of finding M permutations, which can be realized by a reduced number of routes, from different ECs can be formulated by adding the following quadratic constraint to (3):

$$\boldsymbol{x}\boldsymbol{Q}'\boldsymbol{x}^\top = K, \tag{14}$$

where $K = M - 1$.

We implement our problem formulation in the CVX modeling framework for MATLAB with the MOSEK solver, the equality constraint (14) is written as

$$\boldsymbol{x}\boldsymbol{Q}'\boldsymbol{x}^\top \leq K \tag{15}$$

because the CVX with MOSEK only accepts quadratic inequality constraints, which can be transformed into a convex constraint instead of a concave constraint, in the standard form (i.e., $\leq$) instead of quadratic equality constraints. This reformulation is valid because the solution of

$$\min_{\boldsymbol{x}} \boldsymbol{x}\boldsymbol{Q}'\boldsymbol{x}^\top \text{ s.t. (3b)}$$

is K, and using the constraint (15) enforces $\boldsymbol{x}\boldsymbol{Q}'\boldsymbol{x}^\top$ to take value K.

Fig. 2 shows an example of the bipartite graph representation of the MKU formulation shown in Equation (3) with the quadratic constraint (14). Compared to the left figure in Fig. 2, automorphisms in different ECs are denoted by different colors. The goal of the MKU problem with the quadratic constraint is changed to select K automorphisms from different ECs such that the number of required routes is reduced. Filled nodes denote selected automorphisms and their required routes. Fig. 2 shows an example of the MKU problem with the quadratic constraint and $K = 2$. The minimum number of required routes is 2, there exist multiple solutions, and filled nodes and black lines show one of the solutions.

C. Problem Size Reduction Based on Hamming Distance

It is shown in [16] that, for some polar codes, the AED-SC yields poor decoding performance when randomly selected automorphisms from different ECs. A HD-based heuristic, which is first proposed in [30] (the arXiv version of [9]), is introduced in [16] to construct a group of automorphisms with good decoding performance. Since the automorphism can be represented by $\boldsymbol{P}$ and $\boldsymbol{U}$, matrices $\boldsymbol{P}$s and $\boldsymbol{U}$s are given vector representations in [16], the HD is measured between vector representations for different automorphisms. A group of automorphisms that are from different ECs and the HD between each other are above the given thresholds can be constructed using the HD heuristic proposed in [16]. The group of automorphisms selected by the HD heuristic yields good decoding performance [16]. Given the non-uniqueness of the PUL decomposition [17], it is likely that automorphisms selected by the HD heuristic can be swapped by automorphisms from the same ECs such that the decoding performance is preserved and the implementation requires a reduced number of routes.

To ensure the selected automorphisms have good decoding performance, come from different ECs, and can be realized by a reduced number of routes, this work proposes a two-step construction method. This work uses the HD heuristic proposed in [16] to find automorphisms that can yield good decoding performance. The two-step construction method is as follows:

1) Given the pair of HD thresholds $D = (d_U, d_P)$, initialize the group with the identity permutation, and then find all automorphisms that have a larger Hamming distance than D from each other and come from different ECs; Denotes selected permutations (excluding the identity permutation) as $\mathcal{W}^{'}$.
2) Find all automorphisms that are from these $|\mathcal{W}^{'}|$ ECs and denotes selected automorphisms as $\mathcal{W}$; Find all routes that are required to realize these $|\mathcal{W}|$ automorphisms; Construct the problem of minimizing the required routes as Section V-A and V-B.

This two-step construction not only minimizes the number of required routes while preserving the decoding performance but also reduces the size of our MKU problem formulation (Equation (3)) with the quadratic constraint (14). The reduction is due to the exclusion of automorphisms that are not in the ECs selected by the HD heuristic.

VI. Heuristics for solving MKU with a quadratic constraint

Inspired by the approximation algorithm [28], a sequential algorithm based on solving the least expansion set problem is proposed for solving the MKU problem with the quadratic constraint. Besides, a greedy heuristic is also proposed, which has a complexity that grows linearly with the problem size.

A. Least Expansion Set with the Quadratic Constraint

The least expanding set problem aims to find a set $\mathcal{W}$ of $w \in \mathcal{W}$ with a set $\mathcal{V}$ of neighbor nodes $v \in \mathcal{V}$ such that the ratio of the size of the set's neighbor to the size of the set ($|\mathcal{V}|/|\mathcal{W}|$) is minimized. This problem is equivalent to the problem formulation in (3) without specifying the set size (3b) because minimizing the ratio also minimizes the objective function (3a). The least expanding set problem has a linear programming problem formulation [31], [32], which is convex, and a rounding algorithm combo that can solve the problem in polynomial time [32].

In this work, we want to modify the least expanding set problem formulation and the rounding algorithm in [32] such that $w \in \mathcal{W}$ in the selected set are from different ECs. The modified problem formulation is

$$\min \sum_{v \in \mathcal{V}} x_v, \tag{16a}$$

$$\text{s.t.} \sum_{w \in \mathcal{W}} x_w = 1 \tag{16b}$$

$$x_v \geq x_w \ \forall (w,v) \in \mathcal{E}, w \in \mathcal{W}, v \in \mathcal{V}, \tag{16c}$$

$$x_w,\, x_v \geq 0 \ \forall w,\, v, \tag{16d}$$

$$\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}^{''}\boldsymbol{x}^{\top} = 0, \tag{16e}$$

where

$$\boldsymbol{Q}^{''} = \begin{bmatrix} \boldsymbol{0}^{|V|\times|V|} & \boldsymbol{0}^{|V|\times|W|} \\ \boldsymbol{0}^{|W|\times|V|} & \boldsymbol{Q}_0 \end{bmatrix},$$

$\boldsymbol{Q}_0$ has the same entries as $\boldsymbol{Q}$ except for entries in the diagonal, and entries in the diagonal are set to 0. The modified rounding algorithm for (16) works as the following:

1) Find all unique values of x_w, sort these values in ascending order, and set the returned value of (16a) as the minimum ratio;
2) Create an all-zeros indicator vector $\boldsymbol{x}^{'}$ for $\boldsymbol{x}$ to record x_w and x_v selected by this rounding algorithm;
3) Enumerating all unique values;
4) Create an all-zeros indicator vector $\boldsymbol{x}^{''}$ of $\boldsymbol{x}$;
5) For a unique value in this sorted array, set indicators $x_w^{''}$s of x_ws whose value is larger or equal to the unique value to 1 in $\boldsymbol{x}^{''}$ and set all indicators $x_v^{''}$ of x_v with $((w,v) \in \mathcal{E}) \wedge (x_w = 1)$ to 1 in $\boldsymbol{x}^{''}$;
6) Compute the ratio

$$y_r = \frac{\left|\{x_v | x_v^{''} = 1\}\right|}{|\{x_w | x_w^{''} = 1\}|},$$

and the corresponding quadratic constraint

$$y_c = \frac{1}{2}\boldsymbol{x}^{''}\boldsymbol{Q}^{''}\boldsymbol{x}^{''\top}; \tag{17}$$

7) If $y_r \leq$ the minimum ratio and $y_c = 0$, set indicators $x_w^{'}$ in $\{x_w | x_w = 1\}$ and indicators $x_v^{'}$ in

$$\{x_v |\, ((w,v) \in E) \wedge (x_w = 1)\}$$

to 1, and 0 otherwise. Then, set the minimum ratio to y_r;
8) If all unique values are tested, end the enumeration. If not, go back to Step 4.

Given the above procedures, the least expanding set problem with the quadratic constraint is still hard to solve, as the quadratic constraint appeared in the equality constraint breaks

the convexity of the original least expanding set problem (16) because the equality constraint should be affine to preserve the convexity of the problem formulation [33].

B. Convexification of the Least Expanding Set Problem with the Quadratic Constraint

To convexify the problem formulation (16), the quadratic equality constraint (16e) can be converted to the following two inequality constraints [34]:

$$\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}^{''}\boldsymbol{x}^{\top} \leq 0, \tag{18a}$$

$$-\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}^{''}\boldsymbol{x}^{\top} \leq 0. \tag{18b}$$

To preserve the convexity of the problem formulation, the inequality constraints should be convex [33]. As the matrix $\boldsymbol{Q}^{''}$ might not be positive semi-definite (PSD), this work uses a convexification method proposed in [35], and the procedures are the following.

Assume the matrix $\boldsymbol{Q}^{''}$ is not PSD, and $\boldsymbol{Q}^{''}$ can be converted to the following subtraction form:

$$\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}^{''}\boldsymbol{x}^{\top} = \frac{1}{2}\boldsymbol{x}\boldsymbol{Q}_{+}^{''}\boldsymbol{x}^{\top} - \frac{1}{2}\boldsymbol{x}\boldsymbol{Q}_{-}^{''}\boldsymbol{x}^{\top} \leq 0, \tag{19a}$$

$$-\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}^{''}\boldsymbol{x}^{\top} = -\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}_{+}^{''}\boldsymbol{x}^{\top} + \frac{1}{2}\boldsymbol{x}\boldsymbol{Q}_{-}^{''}\boldsymbol{x}^{\top} \leq 0, \tag{19b}$$

where $\boldsymbol{Q}_{+}^{''}$ and $\boldsymbol{Q}_{-}^{''}$ are PSD.

As $\boldsymbol{Q}_{+}^{''}$ and $\boldsymbol{Q}_{-}^{''}$ are PSD, the first-order Taylor expansion of quadratic terms involved $\boldsymbol{Q}_{-}^{''}$ can serve as an affine lower bound for these quadratic terms, and the equation (19) after the Taylor expansion becomes

$$\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}_{+}^{''}\boldsymbol{x}^{\top} - \boldsymbol{x}\boldsymbol{Q}_{-}^{''}\boldsymbol{x}_0^{\top} - \frac{1}{2}\boldsymbol{x}_0\boldsymbol{Q}_{-}^{''}\boldsymbol{x}_0^{\top} \leq 0, \tag{20a}$$

$$-\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}_{+}^{''}\boldsymbol{x}^{\top} + \boldsymbol{x}\boldsymbol{Q}_{-}^{''}\boldsymbol{x}_0^{\top} + \frac{1}{2}\boldsymbol{x}_0\boldsymbol{Q}_{-}^{''}\boldsymbol{x}_0^{\top} \leq 0. \tag{20b}$$

The Taylor expansion is taken at the point $\boldsymbol{x}_0$, and (20a) and (20b) are in the standard quadratic form, and they are convex.

The quadratic term $\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}_{-}^{''}\boldsymbol{x}^{\top}$ is equivalent to the second-order Taylor expansion because its higher-order derivatives are zeros, and the corresponding second-order Taylor expansion is

$$\frac{1}{2}\boldsymbol{x}_0\boldsymbol{Q}_{-}^{''}\boldsymbol{x}_0^{\top} + \boldsymbol{x}\boldsymbol{Q}_{-}^{''}\boldsymbol{x}_0^{\top} + \frac{1}{2}\boldsymbol{x}\boldsymbol{Q}_{-}^{''}\boldsymbol{x}^{\top}. \tag{21}$$

Since a Taylor expansion is taken, errors incurred by the approximation should be taken into account to preserve the solvability of the problem formulation. The error of the approximation is the difference between the first-order Taylor expansion of the quadratic term and the second-order Taylor expansion, which is

$$\frac{1}{2}\boldsymbol{x}\boldsymbol{Q}_{-}^{''}\boldsymbol{x}^{\top} \leq \sum_{i=1}^{|W|+|V|}\sum_{j=1}^{|W|+|V|}|q_{i,j}^{''}|, \tag{22}$$

where $q_{i,j}^{''} \in \boldsymbol{Q}_{-}^{''}$, and this inequality holds because $x_w \in [0,1]$. Thus, the error induced by the first-order Taylor expansion can be controlled by controlling the magnitude of entries in $\boldsymbol{Q}_{-}^{''}$.

The following theorem shows that the eigenvalue of matrices with the block diagonal structure (e.g., $\boldsymbol{Q}^{'}$ and $\boldsymbol{Q}^{''}$) can be controlled by controlling the eigenvalues of square matrices in the matrix diagonal.

Theorem 2. *The eigenvalues of a block diagonal matrix are the union of the eigenvalues of all square matrices on the diagonal.*

Proof. I), Let the block diagonal matrix $\boldsymbol{A}$ with 2 blocks takes the following form:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_2 \end{bmatrix},$$

where $\boldsymbol{A}_1$ and A_2 are two square matrices. The determinant of $\boldsymbol{A}$ is [36]:

$$\det(\boldsymbol{A}) = \det(\boldsymbol{A}_1)\det(\boldsymbol{A}_2);$$

II), Let λ be the eigenvalue of $\boldsymbol{A}$, the characteristic polynomial is $\det(\boldsymbol{A}-\lambda\boldsymbol{I}) = 0$. Since $\boldsymbol{A}-\lambda\boldsymbol{I}$ is also a 2×2 block diagonal matrix,

$$\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = \det(\boldsymbol{A}_1 - \lambda\boldsymbol{I}_1)\det(\boldsymbol{A}_2 - \lambda\boldsymbol{I}_2) = 0,$$

where $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are identity matrices with the same size as $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ respectively;
III), Because $\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = 0$, either $\det(\boldsymbol{A}_1 - \lambda\boldsymbol{I}_1)$ or $\det(\boldsymbol{A}_2 - \lambda\boldsymbol{I}_2)$ is 0. Hence, λ is also the eigenvalue of $\boldsymbol{A}_1$ or $\boldsymbol{A}_2$. It can be concluded that the set of eigenvalues of $\boldsymbol{A}$ are the union of eigenvalues of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ □

Based on Theorem 2, it can be concluded that if eigenvalues of matrices on the diagonal are non-negative, then the eigenvalues of the 2×2 block diagonal matrix are non-negative.

In this work, we proposed to set the matrix $\boldsymbol{Q}_{+}^{''}$ and $\boldsymbol{Q}_{-}^{''}$ as the following to minimize the approximation error:

1) According to Theorem 2, the non-PSD matrix $\boldsymbol{Q}^{''}$ has a negative minimum eigenvalue λ_{min}, and the minimum eigenvalue is equal to the minimum eigenvalue of $\boldsymbol{Q}_0$.
2) We set $\boldsymbol{Q}_{-}^{''}$ as

$$\boldsymbol{Q}_{-}^{''} = \begin{bmatrix} \boldsymbol{0}^{|\mathcal{V}|\times|\mathcal{V}|} & \boldsymbol{0}^{|\mathcal{V}|\times|\mathcal{W}|} \\ \boldsymbol{0}^{|\mathcal{W}|\times|\mathcal{V}|} & |\lambda_{min}|\boldsymbol{I} \end{bmatrix},$$

where $\boldsymbol{I}$ is a $|\mathcal{W}| \times |\mathcal{W}|$ identity matrix, and, by Theorem 2, the minimum eigenvalue of $\boldsymbol{Q}_{-}^{''}$ is 0.
3) The error induced by the first-order Taylor expansion is upper-bounded by

$$\sum_{i=1}^{|\mathcal{W}|+|\mathcal{V}|}\sum_{j=1}^{|\mathcal{W}|+|\mathcal{V}|}|q_{i,j}^{''}| = |\mathcal{W}| * |\lambda_{min}|,$$

as entries in the other three sub-matrices are zeros.
4) The other PSD matrix $\boldsymbol{Q}_{+}^{''}$ is set as

$$\boldsymbol{Q}_{+}^{''} = \boldsymbol{Q}_{-}^{''} + \boldsymbol{Q}^{''},$$

and $\boldsymbol{Q}_{+}^{''}$ takes the following form:

$$\boldsymbol{Q}_{+}^{''} = \begin{bmatrix} \boldsymbol{0}^{|\mathcal{V}|\times|\mathcal{V}|} & \boldsymbol{0}^{|\mathcal{V}|\times|\mathcal{W}|} \\ \boldsymbol{0}^{|\mathcal{W}|\times|\mathcal{V}|} & \boldsymbol{Q}_0 + |\lambda_{min}|\boldsymbol{I} \end{bmatrix},$$

5) We know the minimum eigenvalue of $\boldsymbol{Q}_0 + |\lambda_{min}|\boldsymbol{I}$ is $\lambda_{min} + |\lambda_{min}| = 0$, then, by Theorem 2, the minimum eigenvalue of $\boldsymbol{Q}_+^{''}$ is 0.

C. Selecting K Automorphisms from the Least Expanding Set

Once we have this convex problem formulation, this problem is solved iteratively [35] to find a "good" feasible point with a small objective value (16a) and a least expanding set returned from our modified rounding algorithm.

The solving process starts with an initial point $\boldsymbol{x}_0$, takes the first-order Taylor expansion at this point, and solves the least expanding set problem with the quadratic constraint (16) but replacing the quadratic constraint (16e) with (20). The returned results are checked against constraints (16b), (16c), (16d), and (16e), and see if these constraints are satisfied within an acceptable tolerance. If constraints are satisfied, the return point is considered feasible, and the optimization process stops. If not, the return point is set as $\boldsymbol{x}_0$, and repeat the solving process. In this paper, this solving technique is referred to as sequential quadratic constrained linear programming (SQCLP) because of its iterative nature.

From our experiments, our proposed methods tend to return a least-expanding set with a size that is much larger than K. To select K automorphisms from the returned least expanding set, either solve the problem formulation (3) using solvers or use the approximation algorithm proposed in [28]. As reproducing the approximation algorithm is out of the scope of this work, we use the existing commercial solver MOSEK to solve the MKU problem (3) to demonstrate the effectiveness.

D. A Greedy Heuristic

The proposed reformulation SQCLP of the MKU problem with the quadratic constraint provides an alternative path to reach a solution, but it is still constrained by the memory limit of the simulation platform and the unknown number of iterations to reach a solution. A greedy heuristic is proposed in this work to return a good solution within a finite time step.

The greedy heuristic starts with the creation of an empty set $\mathcal{S}$ for the automorphism, an empty set $\mathcal{U}$ for the routes, and a copy $\mathcal{W}^{'}$ of $\mathcal{W}$. Every automorphism $w \in \mathcal{W}$ has corresponding basic routes, and the indicator of all routes is stored in the matrix $\boldsymbol{B}$. For every $w_i \neq w_j$, the $|\mathcal{W}| \times |\mathcal{W}|$ matrix $\boldsymbol{Q}$ stores the information of whether w_i and w_j are in the same EC or not.

The sets $\mathcal{S}$ and $\mathcal{U}$ are initialized with the automorphism $w^{'}$, which requires the minimum number of routes, and the indicators of associated routes. All automorphisms ws that belong to the same EC as the $w^{'}$ are removed from the set $\mathcal{W}^{'}$ of unselected automorphisms. Then, the union between the basic routes set of selected automorphisms and the basic routes for each unselected automorphism is computed. The unselected automorphism that returns the union with the smallest size among all unselected automorphisms

$$w^{'} = \underset{w \in \mathcal{W}^{'}}{\arg\min} |\boldsymbol{B}[w] \cup \mathcal{U}| \tag{23}$$

is included in the set. Automorphisms $w^{'}$s that belong to the same EC as the selected automorphism are removed from $\mathcal{W}^{'}$.

Algorithm 5: Greedy Heuristic

Input: The indicator matrix of routes $\boldsymbol{B}$, The correlation matrix $\boldsymbol{Q}$, The set of automorphisms $\mathcal{W}$, The target number of selected automorphisms K

Output: The set of selected automorphisms $\mathcal{S}$, The set of routes for selected automorphisms $\mathcal{U}$

$\mathcal{S}, \mathcal{U} \leftarrow [\,], [\,]$ /* Initialize sets for automorphisms and routes. */

$\mathcal{W}^{'} \leftarrow \mathcal{W}$ /* Copy the set of automorphisms. */

$\boldsymbol{a} \leftarrow \mathtt{Sum}(\boldsymbol{B}[i,:])\ \forall i \in \{1, 2, ..., |\mathcal{W}|\}$ /* Compute number of routes required by each automorphism. */

$i \leftarrow \mathtt{Argmin}(\boldsymbol{a})$ /* Find the index of the automorphism required the least number of routes. */

$w^{'} \leftarrow \mathcal{W}^{'}[i]$ /* Select the automorphism required the least number of routes. */

$\mathcal{S} \leftarrow w^{'} \cup \mathcal{S}$ /* Include the selected automorphism into the set. */

$\mathcal{U} \leftarrow \mathcal{U} \cup \boldsymbol{B}[w^{'}]$ /* Include the corresponding routes into the set. */

$\mathcal{W}^{'} \leftarrow \mathcal{W}^{'} \setminus \{w | (w = \mathcal{W}[j]) \ \wedge \ \left(w^{'} = \mathcal{W}[i]\right) \ \wedge \ (\ \boldsymbol{Q}[i,j] = 1)\ \forall j \in \{1, 2, ..., |\mathcal{W}|\}\}$ /* Exclude automorphisms that are in the same EC as the selected automorphism. */

for $i = 1 : K - 1$ **do**

  $\boldsymbol{a}^{'} \leftarrow \boldsymbol{0}$ /* Initialize the vector for recording the size of the union set. */

  for $j = 1 : |\mathcal{W}^{'}|$ **do**

    $w \leftarrow \mathcal{W}^{'}[j]$

    $\boldsymbol{a}^{'}[j] \leftarrow \mathtt{Size}\,(\boldsymbol{B}[w] \cup \mathcal{U})$ /* Compute the size of the union set. */

  $i^{'} \leftarrow \mathtt{Argmin}\left(\boldsymbol{a}^{'}\right)$ /* Find the index of the union set with the smallest size. */

  $w^{'} \leftarrow \mathcal{W}^{'}[i^{'}]$

  $\mathcal{S} \leftarrow w^{'} \cup \mathcal{S}$

  $\mathcal{U} \leftarrow \mathcal{U} \cup \boldsymbol{B}[w^{'}]$

  $\mathcal{W}^{'} \leftarrow \mathcal{W}^{'} \setminus \{w | (w \in \mathcal{W}[j]) \ \wedge \ \left(w^{'} = \mathcal{W}[i^{'}]\right) \ \wedge \ \left(\boldsymbol{Q}[i^{'}, j] = 1\right)\ \forall j \in \{1, 2, ..., |\mathcal{W}|\}\}$

return $\mathcal{S}, \mathcal{U}$

This process is repeated until the cardinality $|\mathcal{S}|$ is equal to K. The pseudo-code of the greedy heuristic is shown in Algorithm 5. From Algorithm 5, the number of union computations required by the greedy heuristic is upper bound by $K|\mathcal{W}|$. Hence, it can be concluded that the proposed heuristic has a complexity of $O(K|\mathcal{W}|)$, which grows linearly with the size of the set of automorphisms.

VII. Experimental Results

Experiments are performed using the binary phase shift keying modulation and the additive white Gaussian noise channel. Polar codes are constructed using the UPO, and the construction method is implemented according to [17]. The fast simplified SC decoder [37] is used as the constituent decoder in the AED.

The benchmark solutions for the MKU problems in this work are solved by the CVX modeling framework for MATLAB 2023a with the MOSEK 10.2.13 solver [38]. The proposed sequential algorithm is implemented by the nonlinear optimization solver in MATLAB 2023a. The greedy heuristic is also implemented in MATLAB 2023a. The simulation

platform uses 12 AMD Ryzen 5 7600x 6-core processors, and it has 16 GB memory.

Experiments are performed on three polar codes used in prior work [17], and the information is detailed as follows:

1) polar codes with $n = 128$ and $k = 85$ $((128, 85))$, $\mathcal{I}_{min} = \{23, 25\}$, 21 ECs, the block diagonal structure $\boldsymbol{s} = [3, 1, 3]$, and the block diagonal structure for the absorption group $\boldsymbol{s}_{\mathbb{1}} = [3, 1, 1, 1, 1]$;
2) polar codes with $n = 256$ and $k = 95$ $((256, 95))$, $\mathcal{I}_{min} = \{55, 120, 228\}$, 21 ECs, the block diagonal structure $\boldsymbol{s} = [2, 1, 1, 1, 3]$, and the block diagonal structure for the absorption group $\boldsymbol{s}_{\mathbb{1}} = [2, 1, 1, 1, 1, 1, 1]$;
3) polar codes with $n = 128$ and $k = 60$ $((128, 60))$, $\mathcal{I}_{min} = \{27\}$, 2205 ECs, the block diagonal structure $\boldsymbol{s} = [3, 4]$, and the block diagonal structure of the absorption group $\boldsymbol{s}_{\mathbb{1}} = [2, 1, 1, 1, 1, 1]$.

Table I shows the number of automorphisms, the number of routes, and the number of ECs (# EC) involved in the formulated MKU problems. We pick these three different polar codes because their MKU problems have a small, a medium, and a large size, respectively. For the selection of the parameter K for the MKU problem, we will pick Ks such that either $K + 1$ is a power of 2, which is commonly adopted by actual implementations [13] and simulations [17], or $K + 1$ is equal to the maximum number of ECs, which investigates the best possible decoding performance. Hence, parameters $K = 15$, $K = 20$, $K = 63$, and $K = 127$ are selected in this work.

Since an initialization point is required by solving the SQCLP, three different initializations are experimented with in this work, and they are:

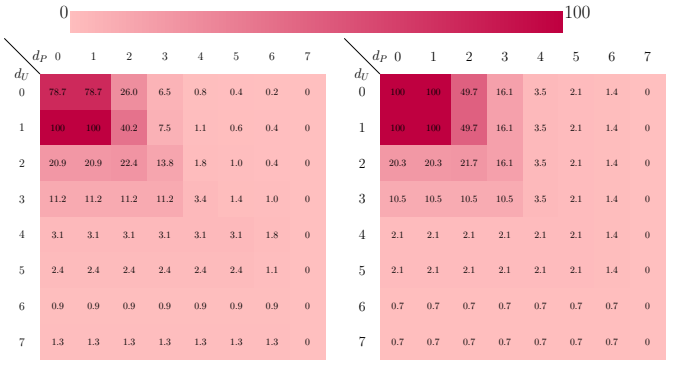
1) An all-zeros vector is used as the initialization of x (ini. 1);
2) A uniform initialization $1/|\mathcal{W}|$ is set as the initialization for x, which satisfies the constraints (16b), (16c), and (16d) (ini. 2);
3) The indicator for the first automorphism w in $\mathcal{W}$ and the indicator for the corresponding routes for this w are set to 1, and it is a feasible point for the problem (16), which is the suggested initialization from [35] (ini. 3).

To demonstrate the effectiveness of the two-step construction method shown in Section V-C, experiments are performed over polar codes $(128, 60)$. From Fig. 3, the number of automorphisms returned from the HD-based two-step method decreases as the threshold increases. Also, it can be observed from Fig. 3 (b) that the same number of ECs is returned from the HD-based method, while, from Fig. 3, the numbers of returned automorphisms are different.

The reason is that different sets of ECs with the same set size are returned. However, the number of automorphisms, which can be described by the equation (6), is different for different sets of ECs, and the number of returned automorphisms is different. In this work, the purpose of using the HD threshold is to reduce the number of automorphisms such that the size of the problem is reduced. Hence, in this work, the result of the HD threshold $(d_U = 1, d_P = 0)$ is shown, which reduces the number of ECs of polar codes $(128, 60)$ from 2204 to 143, a 93.5% reduction, and the number of automorphisms from 73724 to 7236, a 90.2% reduction. The experiment on polar codes $(128, 60)$ with $(d_U = 1, d_P = 0)$ is the largest experiment that can run on our simulation platform.

TABLE I
INFORMATION OF FORMULATED MKU PROBLEMS FOR POLAR CODES

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{W}\rvert$	# EC
$(128, 85)$	68	2,256	20
$(256, 95)$	19	188	20
$(128, 60)$ full problem	517	73,724	2,204
$(128, 60)$ HD $(d_U = 1, d_P = 0)$	517	7,236	143



(a) The normalized number of automorphisms in the percentage. (b) The normalized number of ECs in the percentage.

Fig. 3. The percentage of automorphisms and ECs sampled from the HD-based two-step method with respect to polar codes $(128, 60)$ with $(d_U = 1, d_P = 0)$.

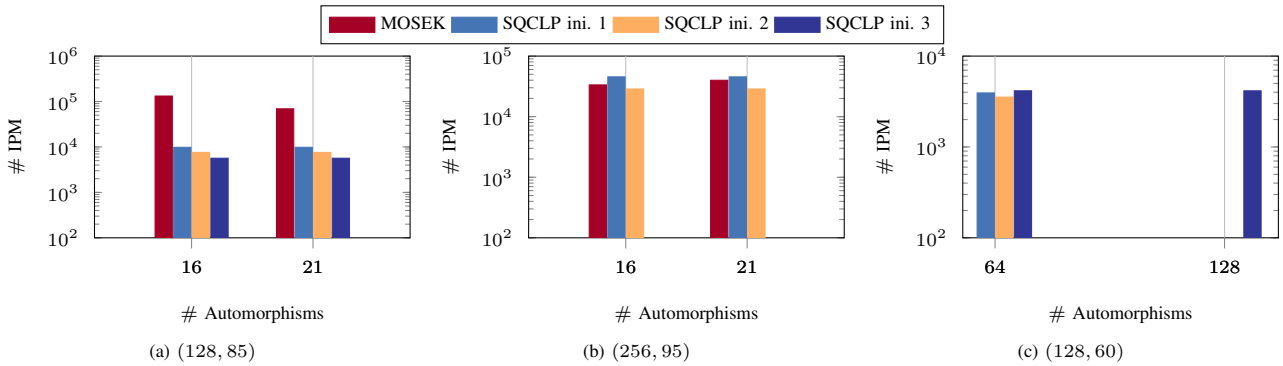
A. Comparisons of the Number of Required Routes

The number of routes returned from different methods is shown in Table II. In Table II, the size of the automorphism set is denoted by a number after - in the code name. For example, the automorphism set with a size of 16 for polar codes $(128, 85)$ is denoted as $(128, 85)$-16. Cases, which are not solvable, are indicated by $\setminus$ in Table II. From Table II, all proposed methods can find an automorphism set that requires fewer routes than random selection, and all proposed methods can return the same result as the MOSEK solver for polar codes $(128, 85)$ and $(256, 95)$. Previous work [13] proposes a parallel implementation for the AED-SC decoder, so the number of required routes is equal to the number of automorphisms implemented. In this comparison, we assume the identity permutation, which does not require any route, is always included in the automorphism set, and the number of routes used by [13] is equal to the number of automorphisms minus one. From Table II, we can see that our method requires fewer routes than [13], and a reduction of up to 65% is observed for polar codes $(128, 85)$ and $(256, 95)$ with an automorphism set size of 21.

The all-zeros initialization (ini. 1) requires more interior point method (IPM) iterations to find a solution than the uniform initialization (ini. 2) for polar codes $(128, 85)$ and $(256, 95)$. When using ini. 3, the SQCLP finds a worse solution than ini. 1 and 2 for polar codes $(128, 85)$, and it fails to find a feasible solution for polar codes $(256, 95)$.

TABLE II
NUMBER OF ROUTES RETURNED BY DIFFERENT METHODS AND DIFFERENT POLAR CODES. THE BEST RESULT IS SHOWN IN BOLD FONT.

	$(128, 85)$-16	$(128, 85)$-21	$(256, 95)$-16	$(256, 95)$-21	$(128, 60)$-64	$(128, 60)$-128
MOSEK	**6**	**7**	**6**	**7**	\	\
SQCLP ini. 1	**6**	**7**	**6**	**7**	51	\
SQCLP ini. 2	**6**	**7**	**6**	**7**	53	\
SQCLP ini. 3	9	14	\	\	48	112
Greedy	**6**	**7**	**6**	**7**	**26**	**56**
Random	17	22	13	14	65	124
[13]	15	20	15	20	63	127



(a) $(128, 85)$ (b) $(256, 95)$ (c) $(128, 60)$

Fig. 4. The number of routes returned from the proposed methods (lower is better), and the number of interior point iterations for solving the MKU problem with the quadratic constraint (higher is better). Missing bars imply the algorithm cannot solve the problem.

When using ini. 3 for polar codes $(256, 95)$, we found that the SQCLP is stuck at a point that is feasible under the SQCLP constraints (16b), (16c), (16d), and (20), and the algorithm stops as the step size is small, but the original quadratic constraint (16e) is not satisfied. Hence, it can be concluded that the initialization for the SQCLP plays an important role in the solvability and the quality of the results. Also, from the result of polar codes $(128, 60)$ in Table II, the automorphism pool selected by the two-step construction method has an automorphism set that can be implemented by a smaller number of routes than the random selection. This result shows that the two-step construction method is an effective way to reduce the problem size while retaining the quality of the solution.

The MOSEK solver fails to solve the MKU problem with the quadratic constraint for polar codes $(128, 60)$ with 64 automorphisms, while the SQCLP algorithm can find a solution under three different initializations. Hence, based on the results of Table II, it can be concluded that the SQCLP method improves the solvability of the MKU problem with the quadratic constraint. However, when solving the problem for polar codes $(128, 60)$, the results returned from the SQCLP method are worse than the greedy algorithm, and the results vary when different initializations are used. When solving the problem for polar codes $(128, 60)$ with 128 automorphisms, the SQCLP can only solve the problem with the ini. 3 as the returned least expanding set has a size of $140 > 127$, and the returned solution is still worse than the solution returned from the greedy algorithm. The SQCLP fails to solve the problem with the ini. 1 and 2 because the returned least expanding set has a size (a size of 121 and 117, respectively) that is smaller than $K = 127$, so the automorphism set with the predefined size $K = 127$ cannot be generated. It can be concluded that the SQCLP algorithm significantly improves the solvability of problems with a medium size (i.e., polar codes $(128, 85)$).

The proposed greedy heuristic can return the same result as the MOSEK solver while having a complexity that scales linearly with the problem size. Generally, the greedy heuristic might not perform well when solving combinatorial optimization problems because the greedy heuristic does not take the different combinations into account while only focusing on the current best solution. The reduced choices after selecting an automorphism may contribute to the superior performance of the greedy heuristic as fewer possible combinations survive. Since the MKU problem with the quadratic constraint is hard to solve, the MOSEK solver and the SQCLP method might not return the globally optimal solution, which makes the greedy heuristic look like having comparable performance.

B. Complexity Comparisons of Solver-based Approach

In Fig. 4, the automorphism set size is denoted by # Automorphisms. The proposed SQCLP and the MOSEK solver use the IPM to solve the MKU problem with the quadratic constraint, so the number of IPM iterations (# IPM) used by the SQCLP and the MOSEK is used to measure the complexity of solving the MKU problem with the quadratic constraint.

From Fig. 4, the proposed SQCLP methods require 86% to 96% fewer IPM iterations than the MOSEK solver when solving the MKU problem with the quadratic constraint for polar codes $(128, 85)$. For problems with a small size (e.g., for polar codes $(256, 95)$), the proposed SQCLP algorithm required a slightly smaller amount of IPM (i.e., 14% to 28% fewer IPMs) than the MOSEK solver with ini. 2.

It can also be observed from Fig. 4 that the MOSEK solver fails to solve the MKU problem with the quadratic constraint for polar codes $(128, 60)$ due to exceeding the time limit

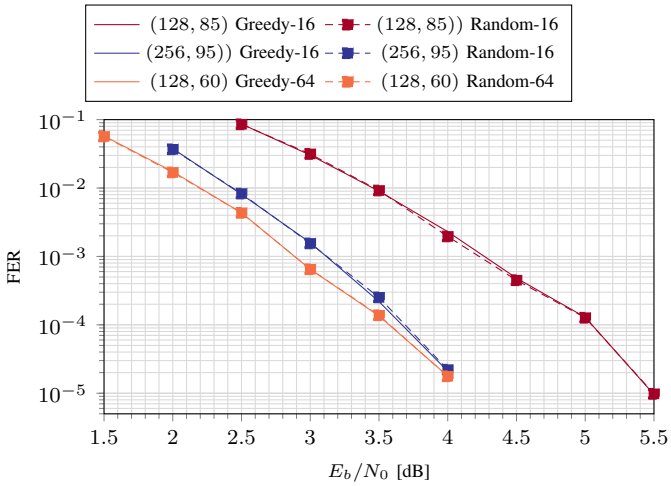


Fig. 5. FERs of polar codes under AED-SC.

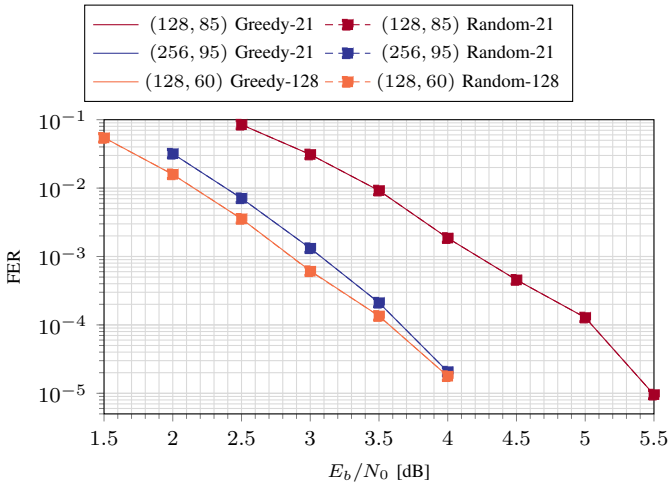


Fig. 6. FERs of polar codes under AED-SC.

(one week). Hence, based on the results of Fig. 4, it can be concluded that the SQCLP method improves the solvability of the MKU problem with the quadratic constraint.

It is also worth noticing that the MOSEK solver needs more IPM iterations to solve polar codes $(128, 85)$ than the number of IPM iterations to solve polar codes $(256, 95)$, while the SQCLP is the opposite. When solving the largest problem, polar codes $(128, 60)$, the SQCLP also uses fewer IPM iterations than the number of IPM iterations for solving polar codes $(256, 95)$ using SQCLP. From these experimental results in Fig. 4, we can see the number of IPM iterations used by the SQCLP depends on the initialization point and varies from problem to problem, hence it is possible that the SQCLP needs more IPM iterations to solve polar codes $(256, 95)$ than polar codes $(128, 85)$.

C. Comparisons of the Decoding Performance

In this work, the automorphism sets generated by the greedy heuristic and the random selection are used to compare the decoding performance measured by the FER. From Fig. 5 and 6, the automorphism sets generated by the greedy heuristic

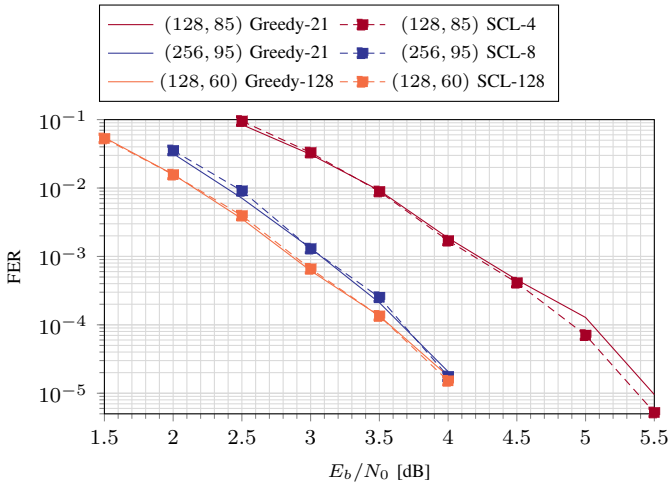


Fig. 7. FERs of polar codes under AED-SC and SCL decoders.

and the random selection return similar decoding performance when automorphism sets of different sizes are used. For polar codes $(128, 85)$ and $(256, 95)$, the automorphism set will return the same decoding performance as the random selection when the size of the set is equal to the number of ECs in the affine automorphism group (e.g., Greedy-21 and Random-21).

We also compare the best possible decoding performance of the AED-SC with the SCL decoders to check how well the AED-SC performs compared to other decoders for polar codes. We use the open-source fast SCL decoder in [39] where the exact path metric (PM), which is also used in [40], is used in the single parity-check (SPC) node instead of using the approximated list sphere metric in [41]. The exact PM is equivalent to the Chase decoding PM [42], [43], and decoding the SPC node with the exact PM requires the same complexity as decoding the SPC node with the approximate PM [43].

Comparisons are plotted in Fig.7. For polar codes $(128, 60)$, the AED-SC with 128 ensembles return a similar FER to the SCL decoder with a list size of 128, which coincides with the observation from [13] where the AED-SC and the SCL decoder return similar decoding performance under the same list/ensemble size. For polar codes $(128, 85)$ and polar codes $(256, 95)$, the AED-SC with 21 ensembles can only return a similar decoding performance to the SCL decoder with list sizes of 4 and 8 respectively. We can conclude that the construction of polar codes significantly affects the decoding performance, but the code construction problem is out of the scope of this work. We want to reemphasize that we choose polar codes $(256, 95)$, $(128, 85)$, and $(128, 60)$ because their MKU problems have a small, a medium, and a large size, respectively, so that we can test the effectiveness of our proposed route selection methods under different problem sizes.

D. Implementation Results of the Routing Unit

Fig. 8 shows the architecture of the proposed routing unit, and this proposed architecture is the same as the architecture of the routing unit in [20]. Compared to the routing unit in [20], we create a fixed state transition in the controller. We

TABLE III
SYNTHESIS RESULTS OF THE ROUTING UNIT ON TSMC 65 NM TECHNOLOGY, THE CLOCK FREQUENCY IS 500 MHZ. W DENOTES THE ROUTING UNIT WITH SHARED ROUTES, AND W/O DENOTES THE ROUTING UNIT WITHOUT SHARED ROUTES.

	$(128, 60)$ w/o	$(128, 60)$ w	$(128, 60)$ w/o	$(128, 60)$ w	$(128, 85)$ w/o	$(128, 85)$ w	$(256, 95)$ w/o	$(256, 95)$ w
(K, q)	$(127, 127)$	$(127, 56)$	$(63, 63)$	$(63, 26)$	$(20, 7)$	$(20, 7)$	$(20, 7)$	$(20, 7)$
Total Area $[\mu m^2]$	$93,683.52$	$21,581.64$	$25,018.92$	$18,651.24$	$5,144.76$	$10,507.68$	$9,926.28$	$21,244.68$
Controller	97.20	$1,817.64$	87.84	$1,044.72$	64.44	189.36	64.44	189.36
Routes and MUXs	$93,586.32$	$12,846.60$	$24,931.08$	$10,689.12$	$5,080.32$	$3,400.92$	$9,861.84$	$7,217.28$
2-to-1 MUX	-	$1,847.52$	-	$1,847.52$	-	$1,847.52$	-	$3,699.36$
Register	-	$5,069.88$	-	$5,069.88$	-	$5,069.88$	-	$10,138.68$
Latency [CCs]	127	929	63	447	20	56	20	56
Power $[mW]$	17.47	13.43	3.04	14.28	0.72	8.54	1.34	17.37

TABLE IV
SYNTHESIS RESULTS OF THE INVERSE ROUTING UNIT ON TSMC 65 NM TECHNOLOGY, THE CLOCK FREQUENCY IS 500 MHZ. W DENOTES THE INVERSE ROUTING UNIT WITH SHARED ROUTES, AND W/O DENOTES THE INVERSE ROUTING UNIT WITHOUT SHARED ROUTES.

	$(128, 60)$ w/o	$(128, 60)$ w	$(128, 60)$ w/o	$(128, 60)$ w	$(128, 85)$ w/o	$(128, 85)$ w	$(256, 95)$ w/o	$(256, 95)$ w
(K, q)	$(127, 127)$	$(127, 56)$	$(63, 63)$	$(63, 26)$	$(20, 7)$	$(20, 7)$	$(20, 7)$	$(20, 7)$
Total Area $[\mu m^2]$	$9,685.80$	$6,464.88$	$6,408.36$	$4,932.36$	$1,208.52$	$2,273.76$	$2,191.68$	$4,323.24$
Controller	97.20	$1,896.48$	86.76	$1,058.76$	64.44	178.56	64.44	176.40
Routes and MUXs	$9,588.60$	$3,183.48$	$6,321.60$	$2,488.68$	$1,144.08$	708.48	$2,127.24$	$1,378.80$
2-to-1 MUX	-	370.08	-	370.08	-	370.80	-	739.44
Register	-	$1,014.84$	-	$1,014.84$	-	$1,014.84$	-	$2,028.60$
Latency [CCs]	127	929	63	447	20	56	20	56
Power $[mW]$	0.69	3.35	0.67	3.17	0.16	1.78	0.30	3.59

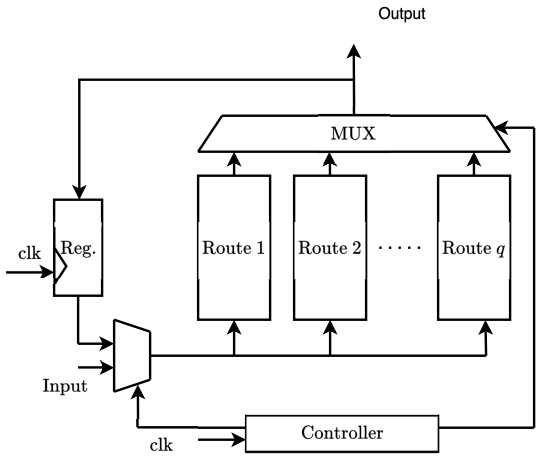


Fig. 8. Proposed hardware architecture of the proposed routing unit with hardware sharing among q basic routes.

also implement a routing network without hardware sharing with q routes for comparisons, where the register (Reg.), the two-input MUX, and their corresponding control signals are removed. The LLR is quantized to 5 bits. This bit width for the channel LLR is selected because we found that the degradation in the decoding performance of the SC decoder is negligible when the binary phase shift keying modulation and the additive white Gaussian noise channel are used.

The routing unit is implemented in VHDL and synthesized by Cadence Genus Synthesis Solution with general-purpose TSMC 65-nm CMOS technology. The synthesis area results of the routing unit with (w) and without (w/o) sharing and their breakdowns are shown in Table III. For polar codes (128,60) with 63 ensembles, the area occupied by the routes is reduced by 57%, which matches the observation of the 59% reduction in the number of routes. Combining all components used in our architecture, the area is reduced by 25% compared to the routing unit without hardware sharing. The area reduction for the routes is 86%, and the overall area reduction increases to 77% when increasing K to 127. However, for problems (e.g., polar codes $(128, 85)$ and $(256, 95)$) with a small $K = 20$, an increase in the area is observed as the reduction in the area required by routing is smaller than the peripheral circuits (i.e., the 2-to-1 multiplexer (MUX) and registers). In future work, new architectural designs should be considered for the routing unit with sharing and a small number of target permutations.

We also present results of other metrics like the latency in the clock cycle (CC) and the power. The latency required by the routing increases as multiple basic routes have to be called in a sequential order to achieve the target automorphism. The existing hardware implementation of the AED-SC decoder has a smaller area and a lower latency than the SCL decoder [13], our proposed routing solution further reduces the area required by the AED-SC, and this reduced area is favorable for applications like the narrowband internet-of-things (IoT), where a low-cost device is needed [44]. Moreover, the increase in the latency is acceptable for the narrowband IoT because the latency requirement is modest [44], and this area and latency trade-off proposed in our work is useful for the narrowband IoT. The routing unit with sharing uses more power than the routing unit without sharing when $K = 20$ and 63. However, the routing unit with sharing uses less power than the routing unit without sharing when $K = 127$.

E. Implementation Results of the Inverse Routing Unit

Besides the synthesis results of the routing unit, we also provide the synthesis results of the inverse routing unit. The inverse routing unit with shared routes follows the same architecture, which is shown in Fig. 8, as the routing unit, but with the following changes. First, the inputs of the basic route i in the routing unit become the outputs of the basic route i in the inverse routing unit. The outputs of the basic route i in the routing unit become the inputs of the basic route i in the inverse routing unit. Secondly, the order of the control signals for basic routes is reversed. For example, assume the inputs should be permuted by basic routes 1, 2, and 3 to achieve the targeted automorphism, and the order of passing through these basic routes is fixed. The control signals in the inverse routing unit will become 3, 2, and 1.

However, instead of routing LLRs, the AED-SC only requires the inverse routing unit to inversely permute the decoded codeword. Compared to the LLRs that require five bits to represent their values, the decoded code bits only require one bit to represent their values. Given that the inverse routing unit has a similar architecture and mechanism to the routing unit for LLRs, we could synthesis the inverse routing unit to check what characteristic the actual hardware has when a low quantization bit width is used.

The synthesis results of the inverse routing unit are shown in Table IV. A 33% and a 23% reduction in the area are observed for polar codes $(128, 60)$ with 127 and 63 automorphisms, respectively. No reduction in the area is observed for polar codes $(128, 85)$ and polar codes $(256, 95)$ with 20 automorphisms. However, the area required by the inverse routing unit is much smaller than the routing unit because of the reduced bit width.

VIII. Conclusion

In this work, we addressed the implementation challenges of routing in the AED-SC. We extended the concept of the basic routes used in factor graph permutations [20] to the affine automorphism group of polar codes, significantly reducing the hardware resources required to implement AED-SC. We showed that finding an automorphism set implementable with a small number of routes is equivalent to solving the MKU problem. To ensure the chosen automorphism set maintains strong decoding performance under AED-SC, we incorporated the relationship of ECs as an additional quadratic constraint in the MKU formulation.

To reduce the problem size of the MKU problem with the quadratic constraint and further enhance the decoding performance, we adopt the HD-based heuristic to reduce the size of the pool of automorphisms involved in the MKU problem with the quadratic constraint. To improve the solvability of the MKU problem with the quadratic constraint, we propose a SQCLP algorithm to reduce the number of IPM iterations that the solver is required to solve the problem. We also propose a greedy heuristic that has a complexity that increases linearly with the size of the automorphism set.

Proposed methods reduce the number of routes by up to 65% compared to the state-of-the-art results. The decoding performance of the automorphism set generated by the proposed method can return similar error correction performance to the automorphism set generated by the random selection of automorphisms from different ECs, while requiring fewer routes. Hardware synthesis results are provided to verify the reduced logic area induced by the proposed method.

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, 2009.

[2] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[3] "TS 38.212 NR; multiplexing and channel coding V17.1.0," 3GPP, Technical Specification (TS), Mar. 2022.

[4] A. Balatsoukas-Stimming, M. Bastani Parizi, and A. Burg, "On metric sorting for successive cancellation list decoding of polar codes," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 1993–1996.

[5] Y. Ren, A. T. Kristensen, Y. Shen, A. Balatsoukas-Stimming, C. Zhang, and A. Burg, "A sequence repetition node-based successive cancellation list decoder for 5G polar codes: Algorithm and implementation," *IEEE Trans. Signal Process.*, vol. 70, pp. 5592–5607, 2022.

[6] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *IEEE International Symposium on Information Theory (ISIT)*, 2009, pp. 1488–1492.

[7] N. Doan, S. A. Hashemi, M. Mondelli, and W. J. Gross, "On the decoding of polar codes on permuted factor graphs," in *IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[8] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation decoding of polar codes on permuted factor graphs," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[9] M. Kamenev, Y. Kameneva, O. Kurmaev, and A. Maevskiy, "Permutation decoding of polar codes," in *XVI International Symposium Problems of Redundancy in Information and Control Systems (REDUNDANCY)*, 2019, pp. 1–6.

[10] M. Geiselhart, A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "On the automorphism group of polar codes," in *IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 1230–1235.

[11] M. Bardet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Algebraic properties of polar codes from a new polynomial formalism," in *IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 230–234.

[12] Y. Li, H. Zhang, R. Li, J. Wang, W. Tong, G. Yan, and Z. Ma, "The complete affine automorphism group of polar codes," in *IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 01–06.

[13] C. Kestel, M. Geiselhart, L. Johannsen, S. ten Brink, and N. Wehn, "Automorphism ensemble polar code decoders for 6g URLLC," in *26th International ITG Workshop on Smart Antennas and 13th Conference on Systems, Communications, and Coding (WSA & SCC)*, 2023, pp. 1–6.

[14] K. Ivanov and R. Urbanke, "Polar codes do not have many affine automorphisms," in *IEEE International Symposium on Information Theory (ISIT)*, 2022, pp. 2374–2378.

[15] K. Ivanov and R. L. Urbanke, "On the efficiency of polar-like decoding for symmetric codes," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 163–170, 2022.

[16] C. Pillet, V. Bioglio, and I. Land, "Classification of automorphisms for the decoding of polar codes," in *IEEE International Conference on Communications (ICC)*, 2022, pp. 110–115.

[17] V. Bioglio, I. Land, and C. Pillet, "Group properties of polar codes for automorphism ensemble decoding," *IEEE Trans. Inf. Theory*, vol. 69, no. 6, pp. 3731–3747, 2023.

[18] M. Geiselhart, A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Automorphism ensemble decoding of Reed–Muller codes," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 6424–6438, 2021.

[19] Z. Ye, Y. Li, H. Zhang, R. Li, J. Wang, G. Yan, and Z. Ma, "The complete SC-invariant affine automorphisms of polar codes," in *IEEE International Symposium on Information Theory (ISIT)*, 2022, pp. 2368–2373.

[20] Y. Ren, Y. Shen, L. Zhang, A. T. Kristensen, A. Balatsoukas-Stimming, E. Boutillon, A. Burg, and C. Zhang, "High-throughput and flexible belief propagation list decoder for polar codes," *IEEE Trans. Signal Process.*, vol. 72, pp. 1158–1174, 2024.

[21] V. E. Beneš, "Optimal rearrangeable multistage connecting networks," *Bell system technical journal*, vol. 43, no. 4, pp. 1641–1656, 1964.

[22] M. Rübenacke, S. Cammerer, M. Sullivan, and A. Keller, "Serial polar automorphism ensemble decoders for physical unclonable functions," *arXiv preprint arXiv:2510.09220*, 2025.

[23] J. Li, H. Zhou, R. Seah, and W. J. Gross, "Automorphism ensemble decoding of polar codes with reduced number of routes," in *13th International Symposium on Topics in Coding (ISTC)*, 2025, pp. 1–5.

[24] C. Schürch, "A partial order for the synthesized channels of a polar code," in *IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 220–224.

[25] M. Geiselhart, J. Clausius, and S. T. Brink, "Rate-compatible polar codes for automorphism ensemble decoding," in *12th International Symposium on Topics in Coding (ISTC)*, 2023, pp. 1–5.

[26] I. Dumer and K. Shabunov, "Soft-decision decoding of Reed-Muller codes: recursive lists," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1260–1266, 2006.

[27] S. A. Vinterbo, "A note on the hardness of the k-ambiguity problem," *Harvard Med. School, Boston, MA, USA, Tech. Rep. DSG*, 2002.

[28] E. Chlamtáč, M. Dinitz, and Y. Makarychev, "Minimizing the union: Tight approximations for small set bipartite vertex expansion," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2017, pp. 881–899.

[29] D. S. Dummit and R. M. Foote, *Abstract algebra*. Wiley Hoboken, 2004, vol. 3.

[30] M. Kamenev, Y. Kameneva, O. Kurmaev, and A. Maevskiy, "Permutation decoding of polar codes," *arXiv preprint arXiv:1901.05459*, 2019.

[31] M. Charikar, "Greedy approximation algorithms for finding dense components in a graph," in *International workshop on approximation algorithms for combinatorial optimization*. Springer, 2000, pp. 84–95.

[32] E. Chlamtáč, M. Dinitz, C. Konrad, G. Kortsarz, and G. Rabanca, "The densest k-subhypergraph problem," *SIAM Journal on Discrete Mathematics*, vol. 32, no. 2, pp. 1458–1477, 2018.

[33] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[34] J. LoVetri, "Inequality constraints," *Lecture Notes in Optimization Methods (ECE 7670), University of Manitoba*, 2005.

[35] A. d'Aspremont and S. Boyd, "Relaxations and randomized methods for nonconvex QCQPs," *EE392o Class Notes, Stanford University*, vol. 1, pp. 1–16, 2003.

[36] J. R. Silvester, "Determinants of block matrices," *The Mathematical Gazette*, vol. 84, no. 501, pp. 460–467, 2000.

[37] 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, 2014.

[38] M. ApS, "Mosek optimization toolbox for matlab," *User's Guide and Reference Manual, Version*, vol. 4, no. 1, p. 116, 2019.

[39] Polar code decoders in Matlab. Accessed on Mar. 26, 2025. [Online]. Available: https://github.com/YuYongRun/PolarCodeDecodersInMatlab

[40] Y. Zhao, Z. Yin, Z. Yang, Z. Wu, and R. Zhang, "Reliability-design of ordered tree-based single-parity-check decoder for polar codes fast list decoding," *IEEE Trans. Rel.*, vol. 72, no. 2, pp. 445–458, 2023.

[41] S. A. Hashemi, C. Condo, and W. J. Gross, "Fast and flexible successive-cancellation list decoders for polar codes," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5756–5769, 2017.

[42] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast list decoders for polar codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 318–328, 2016.

[43] J. Li, S. Shen, and W. J. Gross, "Enhanced successive cancellation list decoder for long polar codes targeting 6G air interface," *arXiv preprint arXiv:2508.16498*, 2025.

[44] S. Landström, J. Bergström, E. Westerberg, and D. Hammarwall, "NB-IoT: A sustainable technology for connecting billions of devices," *Ericsson Technology Review*, vol. 4, pp. 2–11, 2016.