Decoder-Tailored Polar Code Design Using the Genetic Algorithm

Ahmed Elkelesh, Moustafa Ebada, Sebastian Cammerer, *Student Member, IEEE*, and Stephan ten Brink, *Senior Member, IEEE*

***Abstract*— We present a new framework for constructing polar codes (i.e., selecting the frozen bit positions) for arbitrary channels, tailored to a given decoding algorithm rather than assuming the (not necessarily optimal) successive cancellation (SC) decoding. The proposed framework is based on the genetic algorithm (GenAlg), where populations (i.e., collections) of information sets evolve via evolutionary transformations based on their individual error-rate performance. These populations converge toward an information set that fits both the decoding behavior and the defined channel. We construct polar codes, without the CRC-aid, tailored to *plain* successive cancellation list (SCL) decoding, achieving the same error-rate performance as the CRC-aided SCL decoding over both the AWGN channel and the Rayleigh channel, respectively. Furthermore, a proposed belief propagation (BP)-tailored construction approaches the SCL error-rate performance without any modifications in the decoding algorithm itself. The performance gains can be attributed to the significant reduction in the number of low-weight codewords. We show that, when required, the GenAlg can also be set up to find codes that reduce the decoding complexity. This way, the SCL list size or the number of BP iterations can be reduced while maintaining the same error-rate performance.**

***Index Terms*— Polar codes, channel polarization, polar code construction, Reed–Muller codes, genetic algorithm, evolutionary algorithms, artificial intelligence.**

I. Introduction

POLAR codes [2] are the first family of codes proven to be capacity achieving for any Binary Input Discrete Memoryless Channel (BI-DMC) and with an explicit construction method under low complexity successive cancellation (SC) decoding. However, for finite lengths, both the SC decoder and the polar code itself (i.e., its maximum likelihood (ML) performance) are shown to be sub-optimal when compared to other state-of-the-art coding schemes such as low-density parity-check (LDPC) codes [3]. Later, Tal and Vardy introduced a successive cancellation list (SCL) decoder [4] enabling a decoding performance close to the ML bound for sufficiently large list sizes. A complexity reduced version was proposed in [5] with a slight negligible performance degradation. The concatenation with an additional high-rate Cyclic Redundancy Check (CRC) code [4] or parity-check (PC) code [6] further improves the code performance itself, as it increases the minimum distance and, thus, improves the weight spectrum of the code. This simple concatenation renders polar codes into a powerful coding scheme. For short length codes [7], polar codes were recently selected by the 3GPP group as the channel codes for the upcoming *5th generation mobile communication standard (5G)* uplink/downlink control channel [8].

On the other hand, some drawbacks can be seen in their high decoding complexity and large latency for long block lengths under SCL decoding due to their inherent sequential decoding manner (see [9] and references therein), which currently limits the usage of polar codes in several practical applications.

Although several other decoding schemes such as the iterative belief propagation (BP) decoding [10] and the iterative belief propagation list (BPL) decoder [11] exist, they are currently not competitive with CRC-aided SCL decoding in terms of error-rate performance [12]. However, they promise more efficient decoder architectures in terms of decoding latency and parallelism. The partitioned successive cancellation list (PSCL) decoder [13] was introduced achieving almost the same performance of the SCL decoder with a significantly lower memory requirement. Therefore, a good polar code design tailored to an explicit decoder may change this situation as it promises either an improved error-rate performance or savings in terms of decoding complexity for a specific type of decoder. In this work, we show that both can be achieved by considering the decoding algorithm throughout the code construction phase instead of constructing the code based on the typical assumption of SC decoding. An intuitive example why a design for SC is sub-optimal for other decoding schemes can be given by considering the girth of the graph under BP decoding, which strongly depends on the frozen positions of the code. Thus, freezing additional nodes can even result in a degraded decoding performance under BP decoding although the ML performance indeed gets better (see Fig. 9 in [14]).

In a strict sense, one may argue that polar codes are inherently connected with SC decoding and only a design based on the concept of channel polarization results in a *true* "polar" code (cf. Reed–Muller (RM) codes). However, from a

Manuscript received September 20, 2018; revised January 28, 2019 and March 22, 2019; accepted March 22, 2019. Date of publication April 2, 2019; date of current version July 13, 2019. This work has been supported by DFG, Germany, under grant BR 3205/5-1. This paper was presented in part at the International ITG Conference on Systems, Communications and Coding (SCC), February 2019 [1]. The associate editor coordinating the review of this paper and approving it for publication was P. Trifonov. *(Corresponding author: Ahmed Elkelesh.)*

The authors are with the Institute of Telecommunications, University of Stuttgart, 70569 Stuttgart, Germany (e-mail: elkelesh@inue.uni-stuttgart.de; ebada@inue.uni-stuttgart.de; cammerer@inue.uni-stuttgart.de; tenbrink@inue.uni-stuttgart.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2019.2908870

more practical point of view, we seek to find the most efficient coding scheme for finite length constraints and, with slight abuse of notation, we regard the proposed codes as polar codes. Thus, a design method that considers the decoder and improves the overall performance is an important step for future polar code applications.

Except for the Binary Erasure Channel (BEC), existing polar code constructions rely either on analytic approximations/bounds such as the Bhattacharyya parameter [2], density evolution [15] and Gaussian approximation [16], [17] or heuristics [18]–[20], including Monte-Carlo-based simulations for specific decoding algorithms [21]–[23]. Furthermore, an efficient approximation method for the polar code construction problem is proposed in [24]. Additionally, several concatenation schemes of polar codes and other coding techniques have been proposed [25]–[29] to improve the finite length performance under different decoders. However, an explicit design tailored to SCL or BP decoding turns out to be cumbersome due to the many dependencies in the decoding graph and the high dimensionality of the optimization problem.

Polar code construction (or design), throughout this paper, refers to selecting an appropriate frozen bit position vector. In this work, we propose a new framework for polar code construction matched to a specific decoding algorithm embedded in the well-understood Genetic Algorithm (GenAlg) context. As a result, the optimization algorithm works on a specific error-rate simulation setup, i.e., it inherently takes into account the actual decoder and channel. This renders the GenAlg-based polar code optimization into a solid and powerful design method. Furthermore, to the best of our knowledge, the resulting polar codes in this work outperform any known design method for *plain* polar codes under SCL decoding *without* the aid of an additional CRC (i.e., CRC-aided SCL performance could be achieved *without* the aid of a CRC). Additionally, the BP decoder of the *proposed* code achieves (and slightly outperforms) the SCL decoding performance of the *conventional* code without any required decoder modifications. We noticed that the performance gains can be attributed to the significant reduction in the total number of low-weight codewords. Furthermore, the decoding complexity, latency and memory requirements can be reduced for a fixed target error-rate after carefully constructing the polar code using GenAlg. We make the source code public and also provide the best polar code designs from this work online.[1] Other optimization algorithms (e.g., differential evolution) can be used to solve the polar code design problem. However, due to the hard-decision nature of the problem (i.e., a bit-channel can be either frozen or non-frozen) GenAlg seems to be more suitable.

The paper is organized as follows. In Sec. II, we briefly review the fundamental concepts of polar codes. In Sec. III, the problem of polar code construction is presented along with the most relevant work. Sec. IV introduces the GenAlg and discusses its preliminary concepts. The GenAlg-based polar code construction is then presented in Sec. V. Results are presented in Sec. VI and Sec. VII for additive white Gaussian noise (AWGN) and Rayleigh fading channels, respectively. In Sec. VIII, we show that the improved code construction method can lead to significant reduction in decoding complexity, latency and memory requirements. Sec. IX depicts some conclusions.

[1] https://github.com/AhmedElkelesh/Genetic-Algorithm-based-Polar-Code-Construction

II. Polar Codes

Polar codes [2] are based on the concept of channel polarization, in which $N = 2^n$ identical copies of a channel W are combined and N synthesized bit-channels are generated. These synthesized bit-channels show a polarization behavior, in the sense that some bit-channels are purely noiseless and the rest are completely noisy. A recursive channel combination provides the polarization matrix

$$\mathbf{G}_N = \mathbf{B}_N \cdot \mathbf{F}^{\otimes n}, \qquad \mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

where $\mathbf{B}_N$ is a bit-reversal permutation matrix and $\mathbf{F}^{\otimes n}$ denotes the n-th Kronecker power of $\mathbf{F}$. Extensions for kernels other than the 2×2 kernel exist, but are not considered in this work. However, the results from this work can be applied straightforwardly. The polar codewords $\mathbf{x}$ are given by $\mathbf{x} = \mathbf{u} \cdot \mathbf{G}_N$, where $\mathbf{u}$ contains k information bits and $N - k$ frozen bits, w.l.o.g. we set the frozen positions to "0". The information set $\mathbb{A}$ contains the k most reliable positions of $\mathbf{u}$ in which the k information bits are transmitted and $\bar{\mathbb{A}}$ denotes the frozen positions (i.e., the complementary set to $\mathbb{A}$). The *conventional* generator matrix, denoted by $\mathbf{G}$, is constructed as the rows $\{\mathbf{r}_i\}$ of $\mathbf{G}_N$ with $i \in \mathbb{A}$. The task of the polar code construction, in its original form, is to find the information set $\mathbb{A}$ which maximizes the code performance (under SC decoding) for a specific channel condition. More details on the problem of polar code construction is provided in Sec. III.

In this work, a polar code with codeword length N and k information bits is denoted by $\mathscr{P}(N, k)$, i.e., the information set has the cardinality $|\mathbb{A}| = k$ and the code rate $R_c = k/N$.

Systematic polar encoding [30] can be applied which enhances the bit error rate (BER) performance with the same block error rate (BLER) performance, when compared to non-systematic polar codes. Throughout this work, we use non-systematic polar encoding. However, it is straightforward to use the GenAlg to construct systematic polar codes.

The basic polar decoding algorithms are:

- **SC decoding** [2]; the first proposed decoder for polar codes, where all information bits $\hat{u}_i$ are sequentially hard-decided based on the previously estimated bits, $\{\hat{u}_1, \ldots, \hat{u}_{i-1}\}$ and the channel information $\mathbf{y}$. Note that the bit indices $i \in \{1, \ldots, N\}$. Obviously, it suffers from unavoidable error propagation (i.e., decision error at decided bit $\hat{u}_i$ cannot be corrected later and will eventually affect all next bit decisions).
- **SCL decoding** [4]; denoted by SCL (L), utilizes a list of L most likely candidate paths during SC decoding; at every decision the decoder branches into two paths ($\hat{u}_i = 0$ and $\hat{u}_i = 1$) instead of the hard decision in the SC decoder. To limit the exponential growth of complexity, only the L most reliable paths are kept sorted in the list according to a specific path metric.

- **CRC-aided SCL decoding** [4]; denoted by SCL+CRC-r (L), where an additional high-rate CRC of r bits is concatenated to the polar code, to help in selecting the final codeword from the L surviving candidates, yielding significant performance gains in competing with the state-of-the-art error correcting codes.
- **BP decoding** [10]; denoted by BP $(N_{it,max})$ is an iterative message passing decoder based on the *encoding* graph of the polar code. Log-likelihood ratio (LLR) messages are iteratively passed along the encoding graph until reaching a maximum number of iterations $(N_{it,max})$ or meeting an early stopping condition. It can be inherently parallelized and allows soft-in/soft-out decoding. However, the error-rate performance of polar codes under BP decoding is typically not competitive with state-of-the-art SCL decoding. Throughout this work, we use the $\mathbf{G}$*-matrix-based* early stopping condition ("re-encoding"), where decoding terminates when $\hat{\mathbf{x}} = \hat{\mathbf{u}} \cdot \mathbf{G}_N$ [31].

III. Polar Code Construction

The polar code construction phase is about deciding the most reliable k bit positions that are set as the information bit positions, while the remaining $N-k$ bit positions are set as frozen bit positions. Thus, ranking the bit-channels according to their reliabilities is of major significance in the polar code construction phase. The information set $\mathbb{A}$ is the outcome from the polar code construction phase specifying the indices of the information bit positions. A corresponding logical $\mathbf{A}$-vector can be used such that $\mathbf{A} = [a_1, a_2, \ldots, a_N]$, where $a_i \in \{0,1\}$ and $1 \leq i \leq N$. Bit position i is frozen if $a_i = 0$, while bit position j is non-frozen (i.e., can be used for information transmission) if $a_j = 1$. For instance consider the $\mathscr{P}(8,4)$-code, the information set $\mathbb{A} = \{4,6,7,8\}$ can thus be represented by the logical vector $\mathbf{A} = [0\,0\,0\,1\,0\,1\,1\,1]$.

The code construction can be considered as an optimization problem, where the objective is to find the best set of k good positions in a set of indices $\{1,\ldots,N\}$, as shown in equation (1). The reliability measure is the main difference between various code construction algorithms, e.g., upper bound on the BLER, exact BLER or BER.

$$\begin{aligned} \mathbf{A}_{\text{opt}} &= \arg\min_{\mathbf{A}} \mathrm{BER}(\mathbf{A}) \text{ or } \mathrm{BLER}(\mathbf{A}) \\ &\text{subject to } \left(\sum_{i=1}^{N} a_i\right) = k, \\ &\qquad a_i \in \{0,1\}. \end{aligned} \tag{1}$$

where $\mathbf{A} = [a_1, a_2, \ldots, a_N]$, $R_c = k/N$ and $i = 1,2,\ldots,N$.

The BER and BLER in equation (1) are corresponding to a certain channel and a certain decoder. Thus, the information set $\mathbb{A}$ (or, equivalently, the $\mathbf{A}$-vector) is channel dependent, meaning that it depends on the respective channel parameter (e.g., design SNR for AWGN channel, or design ϵ for BEC). Thus, polar codes are non-universal. However, techniques are proposed in [32] to devise universal polar codes. In [10], Arıkan introduced *adaptive* polar codes, or channel-tailored polar codes, in which the polar code is designed using a channel-specific condition, i.e., setting the design SNR equal to the SNR of the transmission channel. It is worth mentioning that the minimum distance of polar codes d_{min} depends on the polar code construction (i.e., $\mathbb{A}$). Polar codes d_{min} is equal to the minimum weight of the rows $\{\mathbf{r}_i\}$ in the matrix $\mathbf{G}_N$ with indices in $\mathbb{A}$ (i.e., $i \in \mathbb{A}$) [33, Lemma 3].

Choosing the best k bit positions for information transmission is even more crucial for short length polar codes. This can be attributed to the fact that the bit-channels of short length polar codes are not fully polarized, and the portion of semi-polarized bit-channels (which would be normally unfrozen) leads to high error-rates. Although efficient polar code construction only exists for the BEC case [2], many algorithms were devised for the AWGN channel case. A survey on the effect of the design SNR and the effect of the specific polar code construction algorithm used on the error-rate performance of SC decoding is presented in [34].

In [2], Arıkan uses the symmetric capacity $I(W_i)$ or the Bhattacharyya parameter $Z(W_i)$ of the virtual channel W_i to assess the reliability of the bit-channels (i.e., denoted as "conventional construction" throughout this work). However, the Bhattacharyya parameters are preferred because of being connected with an explicit bound on the block error probability under SC decoding. A Monte-Carlo-based polar code construction was also proposed by Arıkan in [2].

In [15], a density evolution-based polar code construction algorithm was proposed. A Gaussian approximation of the density evolution for polar code construction was proposed in [16]. In [17], with the help of Gaussian approximation, the approximated BLER is used to assess the reliabilities of the bit-channels instead of the upper bound on the BLER (i.e., Bhattacharyya-based design), assuming SC decoding.

It is important to keep in mind that polar codes that are constructed based on the mutual information or the Bhattacharyya parameters of the bit-channels, as proposed by Arıkan, are tailored to hard-output SC decoders. Thus, they are not necessarily optimum when using other decoders such as the soft-output BP decoder [23], [33], [35] or the SCL decoder [23], [35], [36].

In [33], the authors observed that picking the frozen bit positions according to the RM rule enhances the error-rate performance significantly under maximum a posteriori (MAP) decoding due to the fact that the RM rule maximizes the minimum distance d_{min} of the code. An RM code can be viewed as a polar code with a different frozen/non-frozen bit selection strategy [33], where both codes are based on the same polarization matrix $\mathbf{G}_N$. However, the k information bit positions of RM codes are the positions corresponding to the k row indices with the maximum weights in the matrix $\mathbf{G}_N$. Consequently, the RM code construction phase is channel independent as it merely depends on the row weights of the matrix $\mathbf{G}_N$. A hybrid polar and RM code construction [25] results in significant error-rate improvement gains under SCL decoding without the CRC-aid, by improving the minimum distance of the resultant code. This underlines the benefits of improving the minimum distance of the code and also the need of an improved construction algorithm tailored to the decoder. Similarly, improved error-rate performance of multi-kernel polar codes under SCL decoding was achieved by optimizing the distance properties of polar codes as shown in [37]. A family of codes that interpolates between polar and RM codes was introduced in [35]. These codes pass smoothly from a polar to an RM code, for a fixed codelength and rate, by changing a design parameter α. These codes provide significant error-rate performance gains under BP and SCL decoding.

In [23], an LLR-based polar code construction is proposed, in which the LLRs of the non-frozen bits are tracked during BP decoding to identify weak information bit-channels, and then the information set $\mathbb{A}$ is modified by swapping these weak information bit-channels with strong frozen bit-channels. The resulting code shows an enhanced error-rate performance under both BP and SCL decoding due to the resultant reduction in the number of low-weight codewords.

Furthermore, some work has been done to construct polar codes which are tailored to a specific decoder, e.g., polar code construction assuming SCL decoding [19] and assuming BP decoding [21], [22], where a Monte-Carlo-based construction is proposed similar to [2].

As the output alphabet size of the resulting polar bit-channels grows exponentially with the codelength N, it is computationally of high complexity to precisely calculate $I(W_i)$ or $Z(W_i)$ per bit-channel. However, a quantization can be used to closely approximate them [24]. A recent discovery which reduces the complexity of the polar code construction assuming SC decoding is the partial order for synthesized channels [38], [39], that is independent of the underlying binary-input channel. According to [40], it suffices to compute the Bhattacharyya parameter (or any other reliability parameter) of a sub-linear number of bit-channels, if we take advantage of the partial order. Later an heuristic closed-form algorithm called polarization weight (PW) [18] was proposed to determine the reliability of bit-channels based on their indices and a carefully chosen parameter β [20], resulting in a significant complexity reduction in the polar code construction.

To the best of our knowledge, no analytical polar code construction rule exists thus far which would be *optimized* for BP or SCL decoding and, thus, the nature of the iterative or list decoding is not usually taken into account while designing the information set of a polar code [36]. Thus, the problem of taking the type of decoding into consideration while constructing the polar code is, thus far, an open problem. In this work, we propose a method which always converges to a "good enough" solution, i.e., leads to a better error-rate performance when compared to the state-of-the-art polar code construction techniques available nowadays.

IV. Genetic Algorithm

GenAlg was first introduced by Holland in 1975 [41] as an efficient tool that helps in achieving (good) solutions for high-dimensional optimization problems which are computationally intractable. Beside finding (good) local minima, a well-parametrized GenAlg is known for converging to these minima very quickly [42]. Due to that merit, GenAlg has attracted a lot of research in the Artificial Intelligence (AI) field leading to improved and adaptive variants of it.

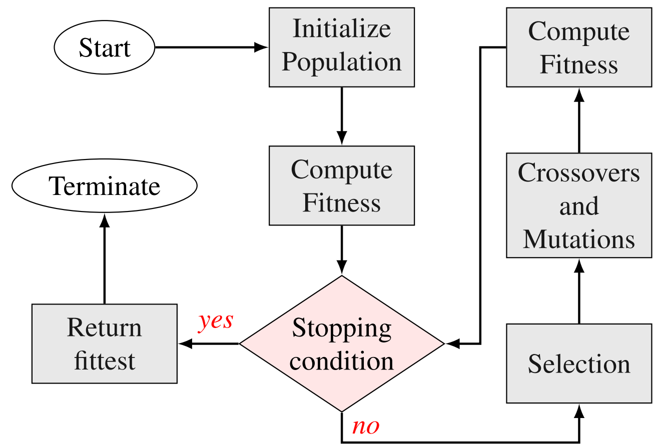


Fig. 1. An abstract view of the genetic algorithm (GenAlg).

Furthermore, researchers from other research fields (e.g., channel coding, signal processing) worked on adapting the GenAlg to some of their specific problems that lacked enough theoretical basis for solving. This lead to improvements over the existent theoretical methods of dealing with the same problems. One of the earliest applications of GenAlg in the field of channel coding was proposed in [43] where the GenAlg was used to find maximal distance codes. Besides, it was used in [44] for decoding linear block codes.

Later, it was proposed in [42] to enable discarding the systematic bits at the encoder side and help reconstructing them at the decoder side using the GenAlg and, thus, enabling rate adjustments without puncturing. Also, a GenAlg-based LDPC decoder was developed [45] where the SNR value is not required at the receiver side. Furthermore, a GenAlg-based decoder for convolutional codes was introduced in [46].

An abstract view of the GenAlg setup is depicted in Fig. 1. We briefly review the GenAlg, revisit the most significant definitions relevant to the scope of this work and refer the interested reader to [47] for more details. GenAlg is inspired by the natural evolution where *populations* of *offsprings*, resulting from *parents*, keep *evolving* and compete, while only the *fittest* offsprings are the ones that *survive*.

GenAlg tries to mimic the evolution of natural organisms where it typically starts with an initial population of candidate individuals and the fittest of them survive and give birth to new offsprings which represent the new population. The criterion of fitness is, therefore, critical for the GenAlg so that the offsprings keep evolving towards the fittest. If the size of population and the number of evolution stages are sufficiently large, the GenAlg converges to an ultimate (sub-) optimal solution. The most important GenAlg fundamentals are briefly introduced in the following subsections.

A. Population

The population, in the GenAlg context, is the collection of individuals that sample the search space at any arbitrary search instance. Each individual (also called “chromosome”) is, for instance, a binary vector of bits (hence called “genes”). These individuals represent the candidate solutions to the objective function of the optimization problem. Each of the individuals is tagged with its own fitness value according to some fitness function. One important parameter of the GenAlg is the (allowed) population size which impacts the optimization problem significantly. The larger the size, the better solution the optimization yields and, however, the slower the convergence speed.

B. Fitness Function

The fitness function, which is most commonly the objective function to be optimized, provides the baseline on which the individuals of a population are to be evaluated and, thus, allowed to survive. It provides a fitness value for each individual which serves as guideline for the GenAlg in the direction of the optimal solution. The precise choice of the fitness function, thus, plays an important role in the quality of the final solution and the speed of converging to it. One important property of the fitness function is its speed of computation as it is extensively executed in a typical optimization problem. A more accurate estimation (or computation) of the fitness function usually enhances the quality of the final solution.

C. Initialization

GenAlg typically starts with a randomly generated population of candidate individuals where each of them compete against each other and only the few fittest of them survive. The surviving individuals will then encounter evolutionary transformations (namely: mutations and crossovers) to generate offsprings which would represent the new population. However, a much faster and, possibly, better convergence is often achieved by an initial population of estimates that are good enough according to their fitness values. It is worth-mentioning that a more diverse initial population can enhance the quality of the acquired solution (i.e., widened search space).

D. Selection

Selection indicates how the GenAlg selects the parents of the next offsprings at each evolution stage, where fitter individuals are forwarded as parents for the upcoming offsprings and the weak ones perish. Many selection strategies exist [48], where they mainly differ in the criterion of parent selection and offspring proportion dedicated for each of the selected parents. The selection strategy applied in this work is called “Truncation selection” strategy where the population is *truncated* to the fittest T individuals and *selected* to be the parents of the next offsprings.

E. Crossover

Crossover is the most significant operation to which GenAlg’s evolution towards the (sub-) optimal solution is mostly attributed. Two parent vectors among the population are subjected to crossover, by selecting a random crossover point (or, generally, several points), such that the two parent vectors exchange their genes (i.e., bits) up to the crossover point. The simplest choice of the crossover point is a single midpoint crossover. The resultant vectors are to undergo competition with other offsprings for survival (i.e., *survival of the fittest*). Typically, crossover occurs at a user-defined rate called “crossover rate” p_c.

F. Mutation

Mutation is the other evolutionary transformation where a small random perturbance is caused to the individual vector offsprings. This perturbance is, in its most common and simplest form, a bit flip in a random position. Mutation, thus, provides more diversity and broadens the search space. Furthermore, mutation helps in reducing the occurrence of a famous phenomenon called *premature convergence*, where early convergence to a sub-optimal solution occurs due to the reduced population diversity [49]. For instance, if all individuals in a population converged to a bit 0 at position i while the optimal solution has bit 1 at this position, only mutation can find a way back towards *this* optimal solution. Similar to crossover, mutation occurs at a user-defined rate called “mutation rate” p_m.

V. GENETIC ALGORITHM-BASED POLAR CODE CONSTRUCTION

As mentioned earlier in equation (1), polar code construction can be viewed as an optimization problem searching for the optimum information set $\mathbb{A}$ that has the minimum (possible) cost function. This optimization problem can be solved using GenAlg. The BER has been selected as the cost function throughout this work for optimization conducted on AWGN channels, in order to be consistent with the results in [4]. Whereas BLER has been selected for optimization conducted on Rayleigh fading channels, in order to be consistent with the results in [50]. Moreover, the complexity can be considered as the cost function (e.g., minimum list size in SCL decoding or minimum number of BP iterations, to achieve a target error-rate). All presented results throughout this work are simulated on GPUs to accelerate our error-rate simulations [51]. Next, we discuss the polar code construction scheme based on GenAlg, Algorithm 1. The whole setup is depicted in Fig. 2.

A. Population

In this work, the population $\mathbb{P} = \{\mathbf{A}_i\}$, for $i = 1, \ldots, S$, is a collection of S candidate information sets represented by their binary $\mathbf{A}$-vectors, each with its own fitness value (e.g., BER or BLER), that represent the search space.

B. Initialization

As it turns out, GenAlg converges faster, and probably to a better solution, if its population $\mathbb{P}_{init}$ is initialized with a sufficiently good collection of estimated $\mathbf{A}$-vectors. For that purpose, the population is initially filled with a collection of $\mathbf{A}$-vectors, all based on the Bhattacharyya construction [2]

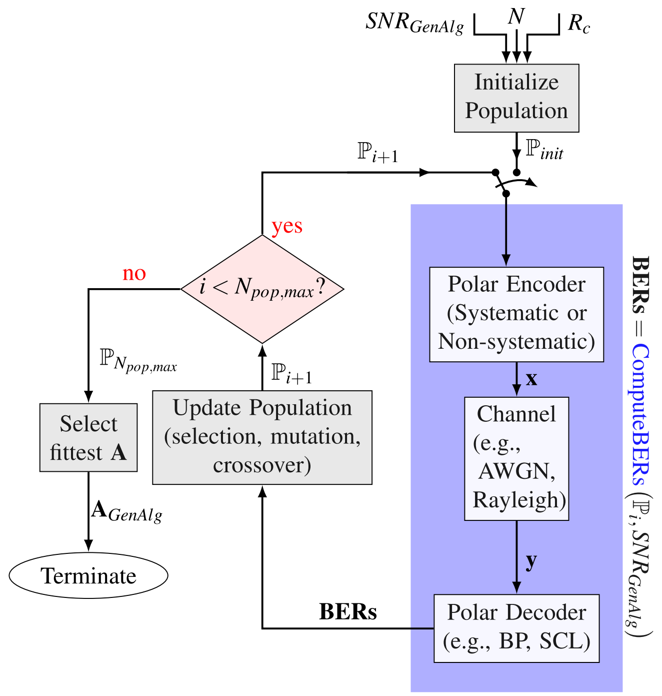


Fig. 2. An abstract view of the genetic algorithm (GenAlg)-based polar code construction.

Algorithm 1 Genetic Algorithm-Based Polar Codes

Input:
N, ▷ codelength
R_c, ▷ code rate
SNR_{GenAlg}, ▷ design SNR (E_b/N_0) of the GenAlg
$N_{pop,max}$, ▷ maximum number of populations
T, ▷ number of truncated parents

Output:
$\mathbf{A}_{GenAlg}$. ▷ optimum **A**-vector

$S \leftarrow 0.5 \cdot (T^2 + 3T)$
▷ Population size, see (2)
$\mathbb{P}_{init} \leftarrow$ `initializePopulation` $(N, R_c, SNR_{GenAlg}, S)$
for $i = 1, 2, \ldots, N_{pop,max}$ **do**
  $\mathbb{P} \leftarrow$ `updatePopulation`($\mathbb{P}$, **BERs**, T)
  BERs $\leftarrow$ `computeBERs`($\mathbb{P}$, SNR_{GenAlg})
end for
$\mathbf{A}_{GenAlg} \leftarrow$ `select fittest` **A**`-vector from` $\mathbb{P}$

obtained for BECs with various erasure probabilities (see Algorithm 2). Besides, we considered having the **A**-vectors constructed according to [25] among the initial population, in the SCL-tailored polar code construction phase, which improved the acquired solution $\mathbf{A}_{GenAlg}$ remarkably. Furthermore, one can possibly use the codes from [23], [35] in the initial population to ensure sufficient population diversity needed for quick convergence.

C. Fitness Function

The cost function chosen in this work is the error-rate (i.e., BER or BLER) at a user-defined design SNR

Algorithm 2 `initializePopulation`

Input:
N, ▷ codelength
R_c, ▷ code rate
SNR_{GenAlg}, ▷ design SNR of the GenAlg
S, ▷ population size

Output:
$\mathbb{P}_{init}$. ▷ initial population

desSNR $\leftarrow \{0, \cdots, 5\}$ dB
▷ design SNRs of $\mathbb{P}_{init}$ or equivalently
▷ various BEC erasure probabilities
`initialize an empty population` $\mathbb{P}_{init}$
for each SNR in **desSNR do**
  A $\leftarrow$`BhattacharyyaConstruction` (N, R_c, SNR)
  ▷ construction according to [2]
  `add` **A** `to` $\mathbb{P}_{init}$
end for
BERs $\leftarrow$`computeBERs`($\mathbb{P}_{init}$, SNR_{GenAlg})
$\mathbb{P}_{init} \leftarrow$ `select fittest` S **A**`-vectors in` $\mathbb{P}_{init}$

Algorithm 3 `computeBERs`

Input:
$\mathbb{P}_{input}$, ▷ input population of **A**-vectors
SNR_{GenAlg}, ▷ design SNR of the GenAlg

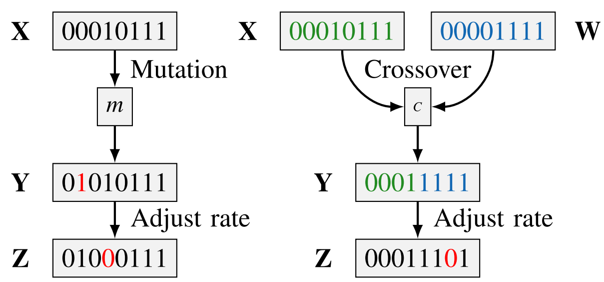
Output:
BERs. ▷ BER-vector corresponding to $\mathbb{P}_{input}$

`initialize an empty vector` **BERs**
for each **A** in $\mathbb{P}_{input}$ **do**
  BER $\leftarrow$ `polarDecode`(**A**, SNR_{GenAlg})
  ▷ simulate a specific **A**-vector under
  ▷ desired decoder (e.g., BP, SCL, ...)
  ▷ over a specific channel @ SNR_{GenAlg}
  `add BER to` **BERs**
end for

(SNR_{GenAlg}). The fitness function that decides the rank of each of the individual **A**-vectors is, thus, selected to be the inverse of the error-rate. In other words, the **A** leading to the minimum error-rate is announced to be the optimum **A** (see Algorithm 3). We noticed that an accurate error-rate simulation significantly improves the quality of the acquired solution $\mathbf{A}_{GenAlg}$. Alternatively, one might consider using mutual information, LLR-reliability $\sum_{i=1}^{N} |LLR_i|$ or any other reasonable metric as the fitness function. This is left as an open research point.

D. Mutation

Mutation guarantees more diversity and acts against premature convergence at a certain bit position. It is, straightforwardly, a bit flip of a random position in the **A**-vector representing a frozen-to-non-frozen (or a non-frozen-to-frozen) switch at that respective bit position. A mutation example is shown in Fig. 3a, where the offspring vector **Y** is the result



(a) Mutation or swapping (b) Crossover

Fig. 3. Examples of GenAlg's evolutionary transformations in the polar code construction context. Inputs ($\mathbf{X}$ and/or $\mathbf{W}$) and output ($\mathbf{Z}$) satisfy the constraints in equation (1).

Algorithm 4 `updatePopulation`

Input:
$\mathbb{P}_{input}$, ▷ input population of $\mathbf{A}$-vectors
BERs, ▷ BER-vector corresponding to $\mathbb{P}_{input}$
T, ▷ number of truncated parents
Output:
$\mathbb{P}_{out}$. ▷ new population after GenAlg ▷ transformations

initialize empty population $\mathbb{P}_{out}$
newParents $\leftarrow$ fittest T $\mathbf{A}$-vectors from $\mathbb{P}_{input}$
add **newParents** to $\mathbb{P}_{out}$
for each A in **newParents do**
  $\mathbf{m} \leftarrow$ mutation($\mathbf{A}$)
  add $\mathbf{m}$ to $\mathbb{P}_{out}$
end for
for each pair $\mathbf{A}_1, \mathbf{A}_2$ in **newParents do**
  $\mathbf{c} \leftarrow$ crossover($\mathbf{A}_1, \mathbf{A}_2$)
  add $\mathbf{c}$ to $\mathbb{P}_{out}$
end for

Algorithm 5 `Mutation`, See Fig. 3a

Input:
$\mathbf{A}_{input}$, ▷ input $\mathbf{A}$-vector
Output:
$\mathbf{A}_{out}$. ▷ output $\mathbf{A}$-vector after swapping a frozen ▷ and a non-frozen bit position ▷ we assume $p_m = 1$

$\mathbf{A}_{out} \leftarrow \mathbf{A}_{input}$ ▷ initialize
$idxs\_ones \leftarrow$ indices of 1's in $\mathbf{A}_{out}$
$idxs\_zeros \leftarrow$ indices of 0's in $\mathbf{A}_{out}$
$i \leftarrow$ random index from $idxs\_ones$
$j \leftarrow$ random index from $idxs\_zeros$
$\mathbf{A}_{out}[i] \leftarrow 0$ ▷ random bit flip 1 to 0 at position i
$\mathbf{A}_{out}[j] \leftarrow 1$ ▷ restore code rate R_c

Algorithm 6 `Crossover`, See Fig. 3b

Input:
$\mathbf{A}_{input_1}$, ▷ 1^{st} input $\mathbf{A}$-vector i.e., parent 1
$\mathbf{A}_{input_2}$, ▷ 2^{nd} input $\mathbf{A}$-vector i.e., parent 2
Output:
$\mathbf{A}_{out}$. ▷ output $\mathbf{A}$-vector ▷ we assume $p_c = 1$

$N \leftarrow$ length of $\mathbf{A}_{input_1}$
$R_c \leftarrow \frac{1}{N} \cdot \sum_{i=1}^{N} (\mathbf{A}_{input_1})$
$\mathbf{A}_{out} \leftarrow$ concat $\left([\mathbf{A}_{input_1}]_1^{N/2}, [\mathbf{A}_{input_2}]_{N/2+1}^{N} \right)$ ▷ concatenate two halves of ▷ $\mathbf{A}_{input_1}$ and $\mathbf{A}_{input_2}$
$R_{c_{out}} \leftarrow \frac{1}{N} \cdot \sum_{i=1}^{N} (\mathbf{A}_{out})$
while $R_{c_{out}} \neq R_c$ **do**
  bit flip in $\mathbf{A}_{out}$
  $R_{c_{out}} \leftarrow \frac{1}{N} \cdot \sum_{i=1}^{N} (\mathbf{A}_{out})$
end while

of bit flipping the 2^{nd} bit of the parent vector $\mathbf{X}$. However, the polar code construction problem has the constraint that the number of non-frozen bit positions (i.e., number of ones) is equal to k to maintain the code rate $R_c = k/N$. To restore the code rate R_c, one further mutation is applied to the resultant offspring vector $\mathbf{Y}$ yielding the vector $\mathbf{Z}$. Thus, the overall operation is a "swap" of a frozen and a non-frozen bit position. The pseudo algorithm of the *mutation* (i.e., swapping) is shown in Algorithm 5. It can be clearly seen that for this specific problem the mutation rate p_m was chosen to be 1 (i.e., one mutation always occurs for each parent per evolutionary step).

E. Crossover

The crossover applied throughout this work is a single midpoint crossover (see Fig. 3b), where the 1^{st} half of the first parent vector $\mathbf{X}$ is combined with the 2^{nd} half of the second parent vector $\mathbf{W}$ to generate the vector $\mathbf{Y}$. This often leads to a change in the number of ones in the resulting vector $\mathbf{Y}$ (i.e., remember from equation (1) that the total number of ones in the $\mathbf{A}$-vector should be equal to k and thus the code rate stays constant $R_c = k/N$). To restore the code rate R_c, sequential bit flipping operations are applied to the resultant vector until reaching the k ones in the binary $\mathbf{A}$-vector ($\mathbf{Z}$ in Fig. 3b). The pseudo algorithm of the *crossover* is shown in Algorithm 6. Similar to mutation, the crossover rate p_c was chosen to be 1 (i.e., one crossover always occurs for each pair of parents per evolutionary step).

F. Selection and Population Update

In this context, by the terms *selection* and *population update* we mean picking the fittest $\mathbf{A}$-vectors and then applying the evolutionary transformations (namely: mutations/swapping and crossovers) in order to generate the new population. We applied the following scheme in order to generate the new population:

- The fittest T $\mathbf{A}$-vectors are always pushed forward as members of the new population (i.e., self-offsprings). Although this could be slowing down the convergence, this, however, ensures convergence to the (local) optimum and guarantees a monotonic behaviour of the cost

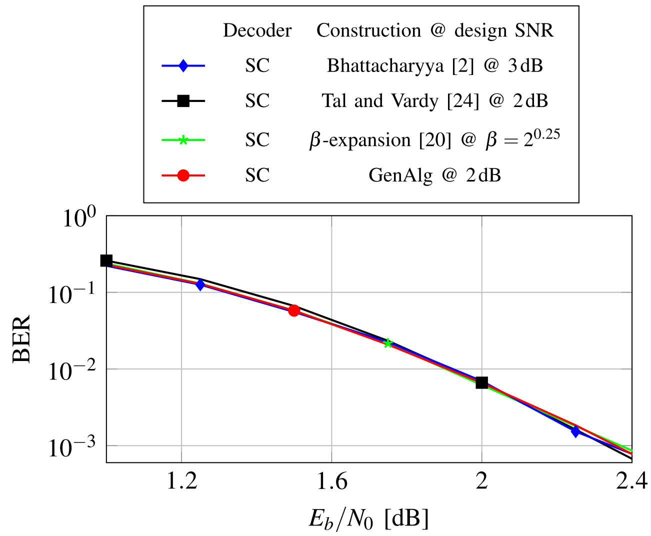
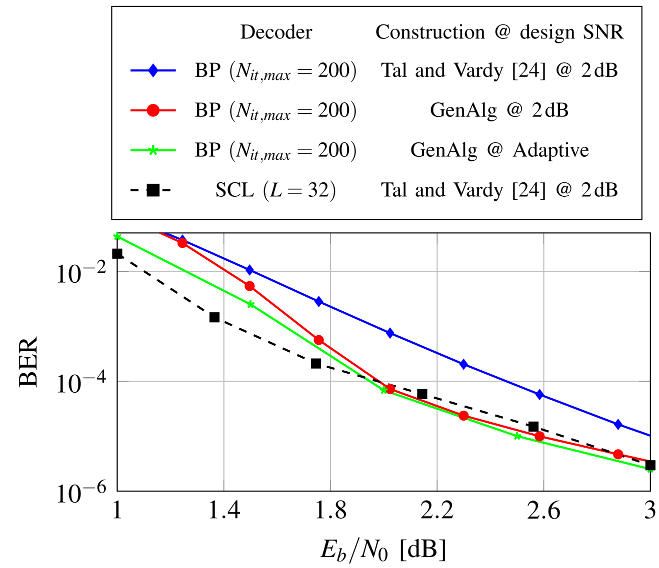
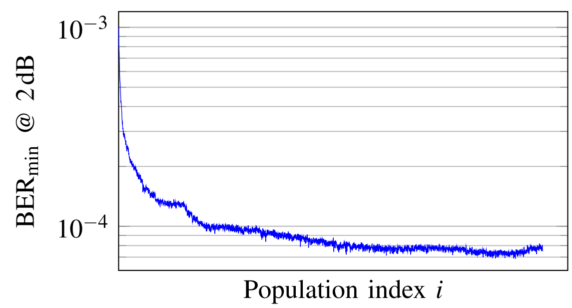


Fig. 4. BER performance of the GenAlg-based $\mathscr{P}(2048,1024)$-code under SC decoding over the AWGN channel and no CRC is used.



(a) BER performance



(b) Evolution of the BER at SNR_{GenAlg} $(E_b/N_0) = 2$ dB

Fig. 5. GenAlg-based construction of a $\mathscr{P}(2048,1024)$-code under BP $(N_{it,max} = 200)$ decoding over the AWGN channel and no CRC is used.

function through evolving populations which facilitates observing the candidate solutions.

- Crossovers are applied between each pair of the fittest T $\mathbf{A}$-vectors, resulting in new $\binom{T}{2}$ offsprings.
- Mutations (i.e., swapping operations) are applied on the fittest T $\mathbf{A}$-vectors, resulting in new T mutated offsprings.

Consequently, the size of the new population S is

$$S = \underbrace{T}_{T\ fittest} + \underbrace{\binom{T}{2}}_{crossover} + \underbrace{T}_{mutation} = \frac{T^2 + 3T}{2} \tag{2}$$

The pseudo algorithm of the *selection* and *population update* scheme followed in this work is shown in Algorithm 4. For all simulation results using the GenAlg as discussed next, we set $S = 20$ and $T = 5$.

VI. Simulation Results Over AWGN Channel

In this section, we show the results of designing polar codes using the GenAlg method over the AWGN channel. To be coherent with the results shown in [4], we use codes of length $N = 2048$ and code rate $R_c = 0.5$ in the AWGN channel simulations.

A. SC Decoder

All of the known polar code construction (i.e., frozen bit position selection) algorithms assume that the SC decoder is used. Thus, they all are tailored to hard-output SC decoding. As shown in Fig. 4, all construction algorithms yield codes having similar error-rate performance over the AWGN channel, including the GenAlg-based construction. A similar independent observation supporting our results was presented in [34].

B. BP Decoder

We use the GenAlg to design polar codes tailored to BP decoding over AWGN channel. Fig. 5a shows a BER comparison between a code constructed via the method proposed in [24] at a design SNR $(E_b/N_0) = 2$ dB and a code constructed using our proposed GenAlg at SNR_{GenAlg} $(E_b/N_0) = 2$ dB under BP $(N_{it,max} = 200)$ decoding. The GenAlg-based construction yields a 0.5 dB net coding gain at BER of 10^{-4} when compared to the construction proposed in [24]. It is worth mentioning that the only difference between the two codes is the selection of the frozen/non-frozen bit positions, i.e., both use exactly the same decoder. Fig. 5b shows the evolution (or enhancement) of the BER per population (i.e., we plot the minimum BER per population). Note that minor fluctuations in the BERs per population index can be explained by the Monte-Carlo output.

The d_{min} of the BP-tailored polar code using GenAlg was found to be 8, while the d_{min} of the polar code constructed via [24] is 16. This is due to the fact that the performance of a linear code under iterative decoding is dominated by the structure of the stopping sets in the Tanner graph of the code and not its d_{min} [52]. Thus, d_{min} is not the only parameter

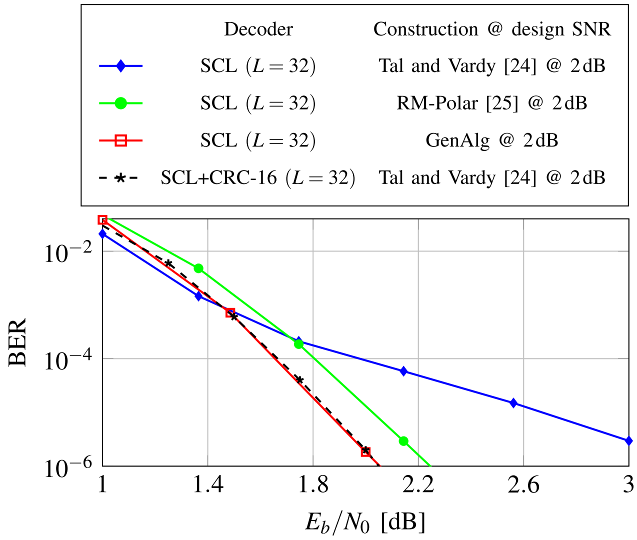


Fig. 6. BER performance of the GenAlg-based $\mathscr{P}(2048{,}1024)$-code under SCL decoding over the AWGN channel. The CRC-aided polar code (– ⋆ –): $N = 2048$, $k = 1024$, $r = 16$ and, thus, the polar code is a $\mathscr{P}(2048{,}1040)$-code.

to be maximized in order to design linear codes tailored to iterative decoding. A fact supporting this claim is that the RM code has a larger (maximum) d_{min} but worse error-rate performance under iterative decoding when compared to polar codes [10].

In the following, we assess the robustness of the GenAlg-based polar code design against inaccuracy in SNR estimation (i.e., having a mismatch between the design SNR and the operating SNR). We use the GenAlg to design an "adaptive" polar code, in which the **A**-vector is channel-tailored, or SNR (E_b/N_0)-tailored (i.e., the frozen set is optimized separately for each value of the SNR of the transmission channel). The BER performance of this GenAlg-based adaptive polar code (–⋆–) is very close to the polar code designed by the GenAlg at 2 dB (–●–) in the high SNR region as shown in Fig. 5a. This means that GenAlg at a specific SNR_{GenAlg} yields a polar code which might be good enough for a certain wider range of SNRs.

Note that the BER performance of the BP-tailored polar code (–●–) is very close to the performance of a polar code constructed via the method proposed in [24] under SCL $(L = 32)$ decoding (–■–), as also shown in Fig. 5a.

C. SCL Decoder

Next, the GenAlg is applied to construct polar codes tailored to SCL $(L = 32)$ over the AWGN channel. Fig. 6 shows a BER comparison between a code constructed via the method proposed in [24] at SNR $(E_b/N_0) =$ 2dB and a code constructed using our proposed GenAlg at SNR_{GenAlg} $(E_b/N_0) =$ 2dB under SCL decoding with list size $L = 32$. The GenAlg-based construction shows significant performance improvements, yielding a 1 dB net coding gain at BER of 10^{-6} when compared to the construction proposed in [24], where again the only difference between the two codes is the frozen/non-frozen bit positions.

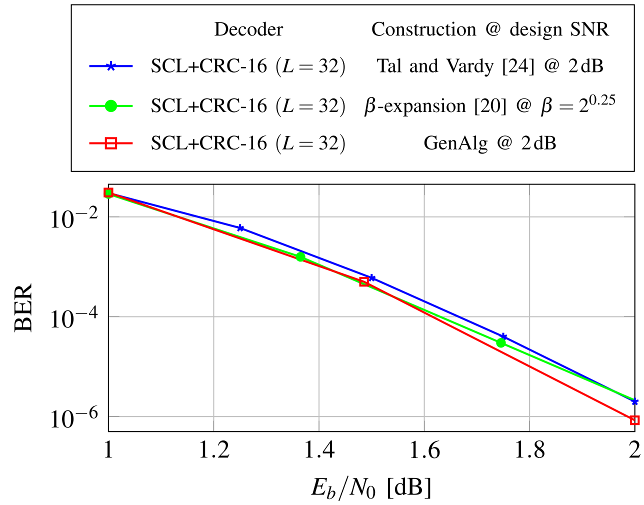


Fig. 7. BER performance of the GenAlg-based $\mathscr{P}(2048{,}1024)$-code under CRC-aided SCL decoding over the AWGN channel. The CRC-aided polar code: $N = 2048$, $k = 1024$, $r = 16$ and, thus, the polar code is a $\mathscr{P}(2048{,}1040)$-code.

The GenAlg-optimized polar code without CRC-aid under SCL decoding (–□–) performs equally well as the CRC-aided polar code under CRC-aided SCL decoding (– ⋆ –), with the same list size $L = 32$ and the same code rate $R_c = 0.5$, as shown in Fig. 6. The d_{min} of the SCL-tailored polar code using GenAlg is 16 which is exactly the same as the d_{min} of the polar code constructed via [24]. However, the RM-Polar code (–●–) has a $d_{min} = 32$. Thus, d_{min} is not the only parameter to be maximized in order to design polar codes tailored to SCL decoding. Note that maximizing d_{min} will lead to an RM code.

D. CRC-aided SCL Decoder

In this section, we use the GenAlg to design polar codes tailored to CRC-aided SCL decoding over AWGN channel. Fig. 7 shows a BER comparison between a code constructed via the method proposed in [24] at SNR $(E_b/N_0) = 2$ dB and a code constructed using our proposed GenAlg at SNR_{GenAlg} $(E_b/N_0) = 2$ dB under CRC-aided SCL decoding with list size $L = 32$. The GenAlg-based construction yields a slight improvement in terms of BER performance when compared to the construction proposed in [24]. We want to emphasize that all BER simulations are performed in the same decoding framework and, thus, the gains are *real* gains caused by the code design.

E. **A**-vector Analysis

The distribution of frozen bit-channels over the N synthesized virtual channels is shown in the "Frozen channel chart" in Fig. 8. Every square (or pixel) in Fig. 8 corresponds to a bit-channel (i.e., we show $N = 2048$ pixels per sub-figure). If a pixel is white, then the corresponding bit-channel is used for information transmission (i.e., included in the information set). If a pixel is colored, then the corresponding bit-channel is a frozen bit position (i.e., included in the frozen set).

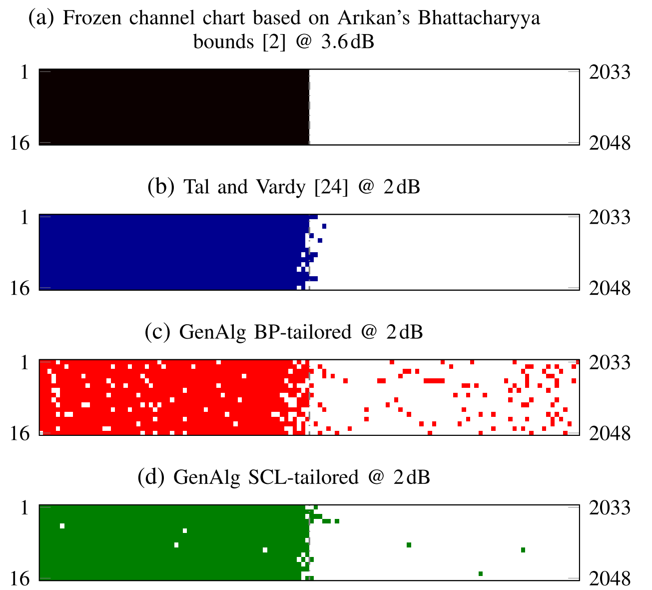


Fig. 8. Frozen channel chart (i.e., frozen bit position pattern) of a $\mathscr{P}(2048,1024)$-code over the AWGN channel with different polar code construction algorithms. The 2048 bit positions are plotted over a 16×128 matrix. Note that the bit-channels are sorted with decreasing Bhattacharyya parameter value. White: non-frozen; colored: frozen.

In Fig. 8a, we show the result of the Bhattacharyya-based construction at a specific channel parameter. For the sake of reproducibility, the frozen/non-frozen set constructed using the Bhattacharyya bounds [2] at design SNR 3.6 dB is equivalent to the frozen/non-frozen set constructed using the Bhattacharyya parameter for a BEC with $\epsilon = 0.32$. The $N-k$ bit-channels with the highest Bhattacharyya value are set to frozen (i.e., black pixel). The k bit-channels with the smallest Bhattacharyya value are set to non-frozen (i.e., white pixel). Then the bit-channels are sorted according to their Bhattacharyya value in descending order (i.e., the $N-k$ frozen bit-channels followed by the k non-frozen bit-channels). And we use this ordering to represent the other three constructions (Fig. 8b, 8c and 8d), which clearly highlights the differences between the output of different construction algorithms.

The conventional construction techniques (e.g., [2], [24]) assume that an SC decoder is used. Thus, these construction techniques yield very similar $\mathbf{A}$-vectors having a very similar distribution as shown in Fig. 8a and 8b. Our proposed GenAlg-based construction tailors the $\mathbf{A}$-vector (or the code) to a specific decoder (e.g., BP or SCL decoder). The BP decoder has an iterative (i.e., non-sequential) decoding nature when compared to SC decoding. Thus, the $\mathbf{A}$-vector tailored to BP decoding has a much different frozen bit-channel distribution as shown in Fig. 8c. Note that the $\mathbf{A}$-vector tailored to BP decoding contains a significant number of non-frozen bits in the first half part of the code, which is not the case in the conventional code construction algorithms assuming SC decoding. Similarly, the GenAlg-based construction tailored to SCL decoding should take the list decoding nature into consideration and, thus, a different frozen bit-channel distribution as shown in Fig. 8d. By comparing Fig. 8a with Fig. 8c and 8d, in a pixel-by-pixel manner, one can see the differences in the frozen/non-frozen sets due to the assumption of a different decoder other than the SC decoder (i.e., BP and SCL) while constructing the code.

TABLE I

ILLUSTRATION OF POLAR DESIGN AND DECODER ARCHITECTURE MISMATCH BY EVALUATING THE MINIMUM E_b/N_0 REQUIRED TO ACHIEVE A TARGET BER OF 10^{-4} FOR A $\mathscr{P}(2048,1024)$-CODE OVER AWGN CHANNEL

Construction @ design SNR	Decoder		
	SC	BP ($N_{it,max} = 200$)	SCL ($L = 32$)
Bhattacharyya [2] @ 3.6 dB	**2.7 dB**	2.45 dB	1.8 dB
Tal and Vardy [24] @ 2 dB	**2.65 dB**	2.45 dB	2 dB
GenAlg BP-tailored @ 2 dB	> 9 dB	**2 dB**	> 7 dB
GenAlg SCL-tailored @ 2 dB	> 6 dB	2.55 dB	**1.65 dB**

TABLE II

THE NUMBER OF LOW-WEIGHT CODEWORDS FOR A $\mathscr{P}(2048,1024)$-CODE OVER AWGN CHANNEL

Construction @ design SNR	d_{min}	A_8	A_{16}
Tal and Vardy [24] @ 2 dB	16	0	11648
GenAlg BP-tailored @ 2 dB	8	8	773
GenAlg SCL-tailored @ 2 dB	16	0	1

Furthermore, Table I investigates the effect of a mismatch between the polar code design and the polar decoder used. It is shown that this mismatch can lead to a rather high error-rate. Mismatch in this context means that a polar code which is designed tailored to decoder X is decoded with another different decoder Y. Table I shows that

- for SC decoding, the best construction method is [24]
- for BP decoding, the best construction method is our proposed GenAlg
- for SCL decoding, the best construction method is our proposed GenAlg.

Thus, the best case (i.e., achieving a target BER with the minimum SNR) is to have a polar code tailored to the used decoder. It should be emphasized that no additional CRC was used for the results in Table I.

F. Weight Enumerators A_d

The weight enumerator A_d is the number of codewords with weight d. There is no closed-form expression for the weight enumerators of polar codes [23]. However, an algorithm proposed in [53] can be used to find the number of minimum-weight codewords (i.e., $A_{d_{min}}$). Table II shows the number of low-weight codewords in polar codes constructed with different methods. Comparing the number of minimum-weight codewords provides one explanation for the improved error-rate performance of the GenAlg-based polar codes. It can be seen that the genetic algorithm significantly reduces the number of minimum-weight codewords and the total number of low-weight codewords, as shown in Table II.

Fig. 6 shows curves with the same slope for different polar codes with different d_{min}. This observation can be attributed to the fact that the SCL ($L = 32$) decoder is a sub-optimal decoder for RM-Polar codes [25] and CRC-aided polar codes [53] (i.e., a larger list size is needed to exploit

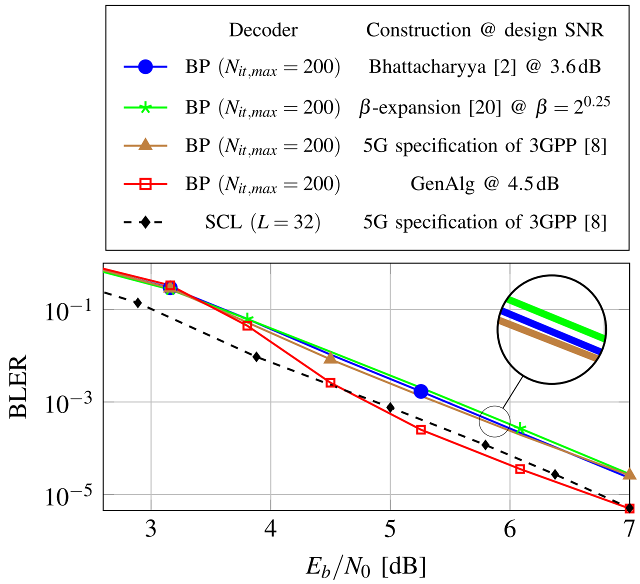


Fig. 9. BLER performance of the GenAlg-based $\mathscr{P}(1024,512)$-code under BP decoding over the Rayleigh fading channel and no CRC is used.

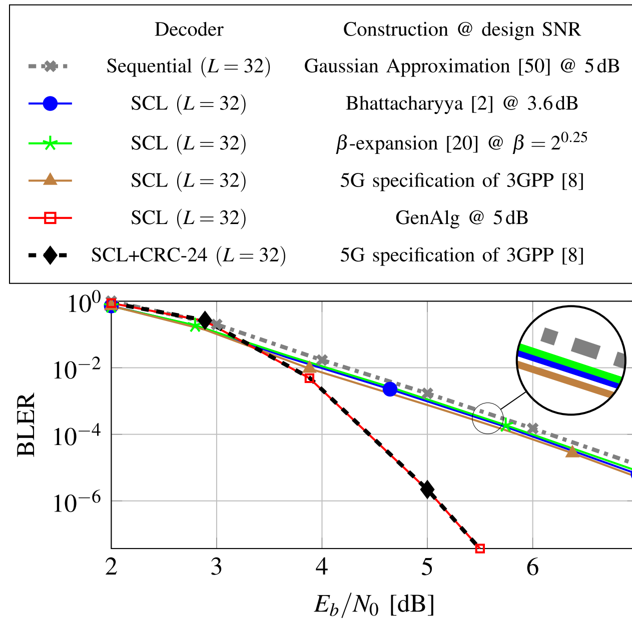


Fig. 10. BLER performance of the GenAlg-based $\mathscr{P}(1024,512)$-code under SCL decoding over the Rayleigh fading channel. The CRC-aided polar code (-◆-): $N = 1024$, $k = 512$, $r = 24$ and, thus, the polar code is a $\mathscr{P}(1024,536)$-code.

the benefits of the enhanced d_{min}). Furthermore, the GenAlg SCL-tailored polar code has $d_{min} = 16$ with only one codeword with weight 16 (i.e., $A_{16} = 1$). Thus, most of the decoding block errors are attributed to the weight-32 codewords, if we consider the case of transmitting the all-zero codeword.

VII. Results for the Rayleigh Fading Channel

The problem of designing or constructing polar codes for the Rayleigh fading channel was tackled in [50], [54]. To demonstrate the flexibility of the GenAlg-based design, we optimize polar codes for Rayleigh fading channels and outline some performance comparisons. To be coherent with the results shown in [50], we use codes of length $N = 1024$ and code rate $R_c = 0.5$ in the (ergodic) Rayleigh fading channel simulations. Following the notation in [55], the system model is given by

$$y = \alpha \cdot x + n$$

where y is the channel output, n is the Gaussian noise attributed to the channel such that $n \sim \mathscr{N}(0, \sigma_{ch}^2)$ and $\alpha > 0$ is the fading coefficient which follows a Rayleigh distribution with $E[\alpha^2] = 1$. Assuming perfect Channel State Information (CSI) (i.e., α is known to the receiver at each received bit position), the corresponding channel LLRs, namely L_{ch}, are computed as

$$L_{ch}(y) = \log \frac{P(y|\alpha, x = +1)}{P(y|\alpha, x = -1)} = \frac{2}{\sigma_{ch}^2} \cdot \alpha \cdot y.$$

A. BP Decoder

We now design polar codes tailored to BP decoding over the Rayleigh fading channel. Fig. 9 shows a BLER comparison between a code constructed via [20], a code constructed via Arıkan's Bhattacharyya bounds [2] at 3.6 dB, a code constructed based on the 5G specification of 3GPP [8] and a code constructed using our proposed GenAlg at SNR_{GenAlg} $(E_b/N_0) = 4.5$ dB under BP decoding with $N_{it,max} = 200$ iterations and the **G**-matrix-based early stopping condition. The GenAlg-based construction yields a 0.8 dB net coding gain at BLER of 10^{-4} when compared to the constructions proposed in [2], [8], [20]. The above mentioned four **A**-vectors yield different polar codes with $d_{min} = 16$. It is remarkable to observe that the BLER performance of the BP-tailored polar code (–□–) outperforms the performance of the proposed 5G polar code [8] under SCL $(L = 32)$ (-◆-) starting from 4.5 dB, as shown in Fig. 9.

B. SCL Decoder

In this section, polar codes are tailored to SCL $(L = 32)$ decoding over the Rayleigh fading channel using the GenAlg. Fig. 10 shows a BLER comparison between a code constructed via [20], a code constructed via Arıkan's Bhattacharyya bounds [2] at 3.6 dB, a code constructed based on the 5G specification of 3GPP [8] and a code constructed using our proposed GenAlg at SNR_{GenAlg} $(E_b/N_0) = 5$ dB under SCL $(L = 32)$ decoding. GenAlg-based construction yields a 2 dB net coding gain at BLER of 10^{-6} when compared to the constructions proposed in [2], [8], [20]. It is worth mentioning that our proposed SCL-tailored polar code (–□–) performs equally well, in terms of error-rate, as the best polar code with dynamic frozen symbols in [50, Fig. 4] for the same list size $L = 32$. Also, our proposed code without the aid of a CRC performs equally well, in terms of error-rate, as the proposed 5G polar code [8] under CRC-aided SCL decoding (-◆-). The GenAlg constructed polar code tailored to SCL decoding has a $d_{min} = 32$.

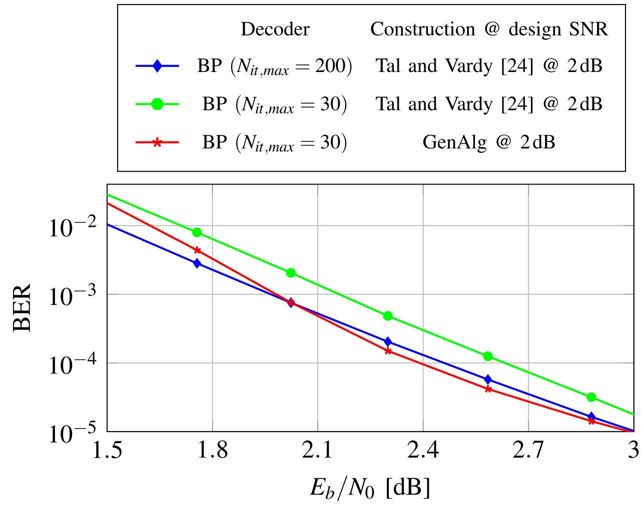


Fig. 11. BER performance of the GenAlg-based $\mathscr{P}$(2048,1024)-code under BP decoding with reduced $N_{it,max}$ over the AWGN channel and no CRC is used.

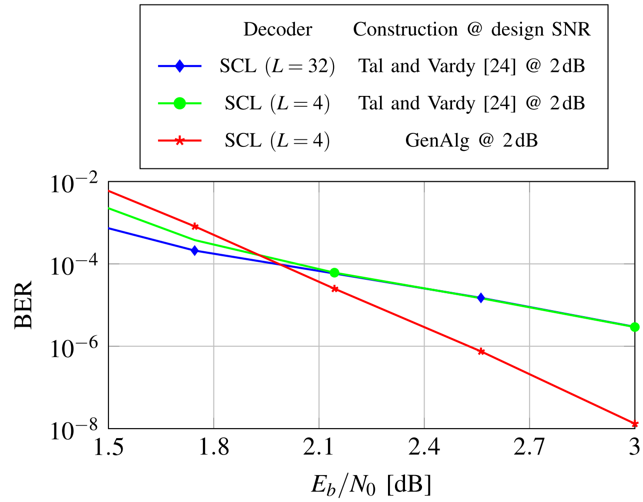


Fig. 12. BER performance of the GenAlg-based $\mathscr{P}$(2048,1024)-code under SCL decoding with reduced list size L over the AWGN channel and no CRC is used.

Note that the above shown gains come "for free", since optimizing the $\mathbf{A}$-vector is done offline and it does not affect the decoding complexity, latency or memory requirements. However, for a fixed target error-rate, the decoding complexity, latency and memory requirements can also be reduced using GenAlg as shown in the next section.

VIII. Decoding Complexity Reduction

Besides error-rate performance, also the decoding complexity, memory requirements and latency are of major significance when selecting the *best* channel coding scheme for a certain application. Further investigations are conducted to study the impact of the polar code construction using the GenAlg on the decoding complexity. This problem can be formulated as follows. We try to construct a polar code using the GenAlg such that, for a given fixed target error-rate level, the decoder used to decode the GenAlg-designed polar code is significantly less complex than the conventional polar code decoder. This complexity reduction is of potential importance when it comes to practical applications with constraints on the decoding complexity, latency and memory requirements.

A. BP Decoder

The GenAlg-based BP-tailored polar code only requires 30 BP iterations to achieve the same error-rate performance as the polar code constructed via [24] under 200 BP iterations, as shown in Fig. 11. Thus, an 85% reduction in the worst case decoding complexity and latency (remember the early stopping condition used) can be achieved with the same error-rate performance just by changing the set of frozen/non-frozen bit positions. Furthermore, at $E_b/N_0 = 2\,\text{dB}$ a 5% reduction in the average number of BP iterations is achieved with the same error-rate performance, reducing the average decoding complexity and latency by 5%.

B. SCL Decoder

The GenAlg-based SCL-tailored polar code only requires a list size of 4 to outperform, in terms of error-rate, the polar code constructed via [24] under SCL with list size 32, as shown in Fig. 12. Thus, an 87.5% reduction in the decoding complexity and memory requirements with an improved error-rate performance can be achieved just by tailoring the set of frozen/non-frozen bit positions to the decoder. Note that a larger list size tends to cause higher decoding latency [56], [57] and thus, the optimized polar code construction also reduces the decoding latency. It should be emphasized that no additional CRC was used for the results in Fig. 12.

Despite the *offline* construction complexity added by the GenAlg, the resulting polar codes can be decoded with state-of-the-art decoders with much lower decoding complexity, latency and memory requirements at a fixed target error-rate, when compared to polar codes constructed via other state-of-the-art polar code construction techniques (e.g., [24]). Thus, it is a trade-off between offline complexity (i.e., polar code construction complexity) and online complexity (i.e., polar decoding complexity).

According to [40], 80% of the channel (i.e., bit-channel reliability) computations can be saved for moderate block length polar codes (e.g., $N \approx 1000$). Thus, we believe that the GenAlg-based polar code construction complexity can be significantly reduced by fixing some good bit-channels as "for sure non-frozen" and some bad bit-channels as "for sure frozen", leading to obvious speed-ups in the convergence speed of the GenAlg. The needed prior knowledge can be easily derived from [20], [38], [40].

IX. Conclusion

As pointed out in [4], the best error-rate performance of finite length polar codes (i.e., under ML decoding without CRC concatenation) is not competitive with the state-of-the-art coding schemes. Thus in this work, we focus on polar code construction (i.e., changing the code itself which is defined by the $\mathbf{A}$-vector) in order to boost the error-rate performance under the state-of-the-art feasible practical decoders (e.g., BP, SCL). We propose a Genetic Algorithm-based polar code design where the channel type and the nature of the decoder are taken into consideration, yielding significant performance gains in terms of error-rate. This also shows the importance of a proper design algorithm and provides an estimate of the capabilities of well-suited polar codes. For a polar code of length 2048 and code rate 0.5 over the binary input AWGN channel under *plain* SCL decoding, approximately a 1 dB coding gain at BER of 10^{-6} is achieved when compared to the conventionally constructed polar codes. This enables achieving the same error-rate performance of polar codes under state-of-the-art CRC-aided SCL decoding with *plain* SCL decoding without the aid of a CRC. Furthermore, GenAlg is used to construct polar codes tailored to flooding BP decoding, finally closing the performance gap between conventional iterative BP decoding and conventional SCL decoding.

GenAlg was used to reduce the decoding complexity, latency and memory requirements of the BP and SCL decoders. Thus, after GenAlg-based $\mathbf{A}$-vector optimization, a few number of BP iterations or a small list size is needed, in order to achieve a fixed target error-rate.

In addition, for a polar code of length 1024 and code rate 0.5, GenAlg is used to optimize polar codes over Rayleigh fading channels, yielding a 2 dB performance gain under SCL decoding at BLER of 10^{-6} when compared to conventional polar design methods. The results suggest that the improved performance is due to the significant reduction in the total number of low-weight codewords. It is worth mentioning that the achieved gains come "for free", since optimizing the $\mathbf{A}$-vector is done once and offline. The source code and the best polar designs from this work can be found in [58].

Acknowledgment

The authors would like to thank Alexander Vardy and Ido Tal for providing their set of frozen/non-frozen bit positions as a baseline for comparison.

References

[1] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Genetic algorithm-based polar code construction for the AWGN channel," in *Proc. 12th Inter. ITG Conf. Syst. Commun. Coding (SCC)*, Feb. 2019, pp. 1–6.

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

[3] A. Balatsoukas-Stimming, P. Giard, and A. Burg, "Comparison of polar decoders with existing low-density parity-check and turbo decoders," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops*, Mar. 2017, pp. 1–6.

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

[5] V. Miloslavskaya and P. Trifonov, "Sequential decoding of polar codes," *IEEE Commun. Lett.*, vol. 18, no. 7, pp. 1127–1130, Jul. 2014.

[6] T. Wang, D. Qu, and T. Jiang, "Parity-check-concatenated polar codes," *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2342–2345, Dec. 2016.

[7] G. Liva, L. Gaudio, T. Ninacs, and T. Jerkovits. (2016). "Code design for short blocks: A survey." [Online]. Available: https://arxiv.org/abs/1610.00873.

[8] (2018). *Technical Specification Group Radio Access Network* [Online]. Available: http://www.3gpp.org/ftp/Specs/archive/38series/38.212/

[9] P. Giard *et al.*, "Hardware decoders for polar codes: An overview," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 149–152.

[10] E. Arıkan, "A performance comparison of polar codes and Reed-Muller codes," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 447–449, Jun. 2008.

[11] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Belief propagation list decoding of polar codes," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1536–1539, Aug. 2018.

[12] K. Niu, K. Chen, J. Lin, and Q. T. Zhang, "Polar codes: Primary concepts and practical decoding algorithms," *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 192–203, Jul. 2014.

[13] S. A. Hashemi, M. Mondelli, S. H. Hassani, C. Condo, R. L. Urbanke, and W. J. Gross, "Decoder partitioning: Towards practical list decoding of polar codes," *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 3749–3759, Sep. 2018.

[14] A. Elkelesh, S. Cammerer, M. Ebada, and S. ten Brink, "Mitigating clipping effects on error floors under Belief propagation decoding of polar codes," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2017, pp. 384–389.

[15] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 519–521, Jul. 2009.

[16] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3221–3227, Nov. 2012.

[17] D. Wu, Y. Li, and Y. Sun, "Construction and block error rate analysis of polar codes over AWGN channel based on Gaussian approximation," *IEEE Commun. Lett.*, vol. 18, no. 7, pp. 1099–1102, Jul. 2014.

[18] *Polar Code Design and Rate Matching*, document 3GPP, R1-167209, 2016.

[19] P. Yuan, T. Prinz, G. Böcherer, O. Iscan, R. Böhnke, and W. Xu, "Polar code construction for list decoding," in *Proc. 12th Int. ITG Conf. Syst. Commun. Coding (SCC)*, Feb. 2019, pp. 1–6.

[20] G. He *et al.*, "Beta-expansion: A theoretical framework for fast and recursive construction of polar codes," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.

[21] J. Liu and J. Sha, "Frozen bits selection for polar codes based on simulation and BP decoding," *IEICE Electron. Express*, vol. 14, no. 6, Mar. 2017, Art. no. 20170026.

[22] S. Sun and Z. Zhang, "Designing practical polar codes using simulation-based bit selection," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 7, no. 4, pp. 594–603, Dec. 2017.

[23] M. Qin, J. Guo, A. Bhatia, A. G. I. Fàbregas, and P. Siegel, "Polar code constructions based on LLR evolution," *IEEE Commun. Lett.*, vol. 21, no. 6, pp. 1221–1224, Jun. 2017.

[24] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.

[25] B. Li, H. Shen, and D. Tse. (2014). "A RM-polar codes." [Online]. Available: https://arxiv.org/abs/1407.5483

[26] P. Trifonov and V. Miloslavskaya, "Polar subcodes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 254–266, Feb. 2016.

[27] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Flexible length polar codes through graph based augmentation," in *Proc. 11th Int. ITG Conf. Syst. Commun. Coding (SCC)*, Feb. 2017, pp. 1–6.

[28] J. Guo, M. Qin, A. G. I. Fàbregas, and P. H. Siegel, "Enhanced Belief propagation decoding of polar codes through concatenation," in *Proc. IEEE Inter. Symp. Inf. Theory (ISIT)*, Jun. 2014, pp. 2987–2991.

[29] A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Improving Belief propagation decoding of polar codes using scattered EXIT charts," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Sep. 2016, pp. 91–95.

[30] E. Arıkan, "Systematic polar coding," *IEEE Commun. Lett.*, vol. 15, no. 8, pp. 860–862, Aug. 2011.

[31] B. Yuan and K. K. Parhi, "Early stopping criteria for energy-efficient low-latency Belief-propagation polar code decoders," *IEEE Trans. Signal Process.*, vol. 62, no. 24, pp. 6496–6506, Dec. 2014.

[32] E. Sasoğlu and L. Wang, "Universal polarization," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 2937–2946, Jun. 2016.

[33] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2009, pp. 1488–1492.

[34] H. Vangala, E. Viterbo, and Y. Hong. (2015). "A comparative study of polar code constructions for the AWGN channel." [Online]. Available: https://arxiv.org/abs/1501.02473

[35] M. Mondelli, S. H. Hassani, and R. L. Urbanke, "From polar to Reed-Muller codes: A technique to improve the finite-length performance," *IEEE Trans. Commun.*, vol. 62, no. 9, pp. 3084–3091, Sep. 2014.

[36] V. Bioglio, C. Condo, and I. Land. (2018). "Design of polar codes in 5G new radio." [Online]. Available: https://arxiv.org/abs/1804.04389
[37] V. Bioglio, F. Gabry, I. Land, and J.-C. Belfiore, "Minimum-distance based construction of multi-kernel polar codes," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.
[38] C. Schürch, "A partial order for the synthesized channels of a polar code," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 220–224.
[39] M. Bardet, V. Dragoi, A. Otmani, and J.-P. Tillich, "Algebraic properties of polar codes from a new polynomial formalism," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 230–234.
[40] M. Mondelli, S. H. Hassani, and R. Urbanke, "Construction of polar codes with sublinear complexity," *IEEE Trans. Inf. Theory*, to be published.
[41] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.
[42] L. Hebbes, R. R. Malyan, and A. P. Lenaghan, "Genetic algorithms for turbo codes," in *Proc. Int. Conf. Comput. Tool*, vol. 1, Nov. 2005, pp. 478–481.
[43] K. Dontas and K. D. Jong, "Discovery of maximal distance codes using genetic algorithms," in *Proc. 2nd Inter. IEEE Conf. Tools Artif. Intell.*, Nov. 1990, pp. 805–811.
[44] H. Maini, K. Mehrotra, C. Mohan, and S. Ranka, "Genetic algorithms for soft-decision decoding of linear block codes," *Evol. Comput.*, vol. 2, no. 2, pp. 145–164, Jun. 1994.
[45] A. Scandurra, A. D. Pra, L. Arnone, L. I. Passoni, and J. C. Moreira, "A genetic-algorithm based decoder for low density parity check codes," *Latin Amer. Appl. Res.*, vol. 36, no. 3, pp. 169–172, Jul. 2006.
[46] H. Berbia, M. Belkasmi, F. Elbouanani, and F. Ayoub, "On the decoding of convolutional codes using genetic algorithms," in *Proc. Int. Conf. Comput. Commun. Eng.*, May 2008, pp. 667–671.
[47] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994.
[48] T. Blickle and L. Thiele, "A comparison of selection schemes used in evolutionary algorithms," *Evol. Comput.*, vol. 4, no. 4, pp. 361–394, Dec. 1996.
[49] L. Juan, C. Zixing, and L. Jianqin, "Premature convergence in genetic algorithm: Analysis and prevention based on chaos operator," in *Proc. 3rd World Congr. Intell. Control Automat.*, vol. 1, Jun. 2000, pp. 495–499.
[50] P. Trifonov, "Design of polar codes for Rayleigh fading channel," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2015, pp. 331–335.
[51] S. Cammerer, B. Leible, M. Stahl, J. Hoydis, and S. ten Brink, "Combining Belief propagation and successive cancellation list decoding of polar codes on a GPU platform," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 3664–3668.
[52] M. Schwartz and A. Vardy, "On the stopping distance and the stopping redundancy of codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 922–932, Mar. 2006.
[53] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2044–2047, Dec. 2012.
[54] A. Bravo-Santos, "Polar codes for the Rayleigh fading channel," *IEEE Commun. Lett.*, vol. 17, no. 12, pp. 2352–2355, Dec. 2013.
[55] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.
[56] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware architecture for list successive cancellation decoding of polar codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 8, pp. 609–613, Aug. 2014.
[57] 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, Feb. 2016.
[58] A. Elkelesh. *Genetic Algorithm-based Polar Code Construction*. Accessed: Dec. 2018. [Online]. Available: https://github.com/AhmedElkelesh/Genetic-Algorithm-based-Polar-Code-Construction

Ahmed Elkelesh received the B.Sc. degree (Hons.) in information engineering and technology from the German University in Cairo in 2013 and the M.Sc. degree (Hons.) in communications engineering and media technology from the University of Stuttgart in 2016, where he is currently pursuing the Ph.D. degree. During his years of study in Germany, he was a Research Assistant with Fraunhofer IPA Stuttgart and an intern at Sony Stuttgart Technology Center. Since 2016, he has been a Member of Research Staff with the Institute of Telecommunications, University of Stuttgart. His main research topics are channel coding, with particular emphasis on polar codes and LDPC codes, information theory, modulation, machine learning, and SDR. He was a recipient of the Anton-und-Klara-Röser Prize in 2017, for his master's thesis.

Moustafa Ebada received the B.Sc. degree (Hons.) from the Communications Department, German University in Cairo, Egypt, in 2013, and the M.Sc. degree in electrical engineering and information technology from the University of Stuttgart, Germany, in 2016, where he is currently pursuing the Ph.D. degree. During his master studies, he was a Research Assistant with multiple institutes of the University of Stuttgart, in the fields radio frequency technology, signal processing, and telecommunications. Since 2016, he has been a Member of Research Staff with the Institute of Telecommunications, University of Stuttgart. His main research topics are channel coding, particularly polar code construction and decoding, as well as designing short LDPC codes for high-speed applications. His research interests include machine learning, particularly designing error correction codes and utilization of the state-of-the-art coding schemes in the field of multiple access channel.

Sebastian Cammerer (S'16) received the B.Sc. and M.Sc. degrees (Hons.) in electrical engineering and information technology from the University of Stuttgart, Germany, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree. During his years of study, he was a Research Assistant at multiple institutes of the University of Stuttgart. Since 2015, he has been a Member of Research Staff with the Institute of Telecommunications, University of Stuttgart. His main research topics are channel coding and machine learning for communications. His further research interests are in the areas of modulation, parallelized computing for signal processing, and information theory. He was a recipient of the VDE-Preis 2016 for his master's thesis, the Rohde & Schwarz Best Bachelor Award in 2015, the Anton-und Klara Röser Prize in 2016, and the Best Publication Award of the University of Stuttgart in 2019.

Stephan ten Brink (M'97–SM'11) was with Bell Laboratories, Holmdel, NJ, USA, from 1995 to 1997 and from 2000 to 2003, conducting research on multiple antenna systems. He was with Realtek Semiconductor Corporation, Irvine, CA, USA, as the Director of the Wireless ASIC Department, from 2003 to 2010, developing WLAN and UWB single chip MAC/PHY CMOS solutions. In 2010, he returned to Bell Laboratories, Stuttgart, Germany, as the Department Head of the Wireless Physical Layer Research Department. He has been a Faculty Member with the University of Stuttgart, Germany, since 2013, where he is currently the Head of the Institute of Telecommunications. He was a recipient and co-recipient of several awards, including the IEEE Stephen O. Rice Paper Prize and the IEEE Communications Society Leonard G. Abraham Prize for contributions to channel coding and signal detection for multiple-antenna systems. He is best known for his work on iterative decoding (EXIT charts) and MIMO communications (soft sphere detection and massive MIMO).