applied sciences

MDPI

Article

Sequential Polar Decoding with Cost Metric Threshold

Ilya Timokhin * and Fedor Ivanov

Higher School of Economics, National Research University, 101000 Moscow, Russia; fivanov@hse.ru
* Correspondence: is.timokhin@hse.ru

Abstract: Polar codes have established themselves as a cornerstone in modern error correction coding due to their capacity-achieving properties and practical implementation advantages. However, decoding polar codes remains a computationally intensive task. In this paper, we introduce a novel approach to improve the decoding efficiency of polar codes by integrating the threshold-based SC-Creeper decoding algorithm, originally designed for convolutional codes. Our proposed decoder with an additional cost function seamlessly merges two established decoding paradigms, namely the stack and Fano approaches. The core idea is to leverage the strengths of both decoding techniques to strike a balance between computational efficiency and performance, with an additional method of controlling movement along a code tree. Simulations demonstrate the superiority of the proposed improved SC-Creeper decoder with tuned parameters. The improved SC-Creeper decoder achieves the performance of the CA-SCL-8 decoder in terms of high code rates and overcomes it in terms of the $N = 1024$ code length, while simultaneously surpassing the efficiency of the traditional Fano decoding algorithm.

Keywords: polar codes; successive cancellation decoding; Fano decoding; stack decoding; Creeper; threshold

1. Introduction

Polar codes [1] achieve near-Shannon-limit performance while maintaining low encoding and decoding complexity, making them ideal for resource-constrained devices. However, their practical utility for short to moderate code lengths has been hampered [2] by the comparative inefficiency of the Successive Cancellation (SC) decoder. The Successive Cancellation List (SCL) decoder, introduced by Tal and Vardy [3], follows a similar path-wise traversal strategy as SC but maintains a list of up to L candidate paths for further exploration. The Successive Cancellation Stack (SCS) decoder [4] offers improved error correction ratios and throughput; however, this achievement comes with the trade-off of heightened space complexity. Another noteworthy development is the SC-Fano decoding algorithm [5], which integrates sequential decoding concepts into the polar decoding traversal procedure. Importantly, some of the strategies employed in these decoding algorithms have their origins in the field of convolutional coding [6], showcasing the interplay between diverse coding techniques.

In this study, we investigate an alternative approach to polar code decoding—the Creeper algorithm [7], originally explored for convolutional codes. The SC-Creeper decoder is engineered for low decoding complexity, making it amenable to resource-constrained environments. The SC-Creeper approach [8,9] represents a novel decoding algorithm, offering an enticing blend of enhanced performance and low computational complexity, rendering it a compelling candidate for practical implementation. This work explores the utilization of the SC-Creeper algorithm for polar decoding and introduces a modified version, leveraging additional threshold values for special nodes in the coding tree, to significantly improve its error correction capabilities.

check for updates

Citation: Timokhin, I.; Ivanov, F. Sequential Polar Decoding with Cost Metric Threshold. *Appl. Sci.* **2024**, *14*, 1847. https://doi.org/10.3390/app14051847

Academic Editor: Ugo Vaccaro

Received: 25 January 2024
Revised: 21 February 2024
Accepted: 21 February 2024
Published: 23 February 2024

2. Preliminaries

2.1. Polar Codes

Polar codes harness channel polarization, wherein initially identical independent channels evolve into two distinct sub-channel types: highly reliable and highly unreliable. Through iterative transformations, these sub-channels converge into two polarized extremes: noisy and nearly noise-free. For finite code lengths N, polarization is not complete, resulting in sub-channels of varying reliability between the extremes. The challenge is to select a subset $\mathcal{I}$ of K sub-channels from $0, 1, \ldots, N-1$ to encode K information bits, while the rest form the frozen sub-channels, denoted as $\mathcal{F}$. A binary (N, K) polar code is defined by the information index set $\mathcal{I}$ and its complement $\mathcal{F}$ with $|\mathcal{I}| = K$, $|\mathcal{F}| = N - K$, and $N = 2^n$, where $n \in \mathbb{N}$. For some message $\mathbf{d} = (d_0, \ldots, d_{N-1})$ and for some codeword $\mathbf{x} = (x_0, \ldots, x_{N-1})$, the encoding process is the basic equation $\mathbf{x}_0^{N-1} = \mathbf{d}_0^{N-1}\mathbf{G}_N$, where $\mathbf{d}, \mathbf{x} \in \{0,1\}^N$, and $\mathbf{G}_N$ is the generator matrix of N-th order, formed as $\mathbf{G}_N = \mathbf{F}^{\otimes n}$, with $\mathbf{F}$ being Arikan's standard polarizing kernel $\mathbf{F} \triangleq \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$ and $\otimes \boldsymbol{n}$ is the $\boldsymbol{n}$-th Kronecker power of the matrix $\mathbf{F}$.

2.2. Successive Cancellation Techniques

Successive Cancellation (SC) treats the decoding process as a tree traversal, with each node representing a bit decision.

It decodes the i-th bit by maximizing the likelihood function $W_n^{(i)}(\mathbf{y}_1^N, \hat{\mathbf{u}}_1^{i-1} | u_i)$, with $\mathbf{y}_1^N = (y_1, y_2, \ldots, y_N)$ representing the received log-likelihoods vector (LLR) from the channel.

$$\hat{u}_i = \begin{cases} \underset{u_i \in \{0,1\}}{} W_n^{(i)}(\mathbf{y}_1^N, \hat{\mathbf{u}}_1^{i-1} | u_i), \text{ if } i \in \mathcal{I}, \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

The Successive Cancellation List (SCL) decoding method takes a different approach compared to SC. Instead of directly estimating the message vector, SCL creates a list of potential candidates and determines [10] the correct one in the final step. To identify the path closest to the source word in terms of probability, additional Cyclic Redundancy Check (CRC) bits are used [11]. These CRC bits are appended to the end of the source word and retained during decoding. This method is known as CRC-aided SCL (CA-SCL) and significantly enhances the performance and path selection accuracy.

The Successive Cancellation Stack (SCS) algorithm [12] is an optimization of the SCL decoder, designed to reduce the decoding complexity (memory and computational) with basic list size and stack size (D) parameters.

Operating on the principle of successive cancellation, the SCS decoder recursively processes the received data to decode the original message. It employs a stack-based approach to efficiently manage the decoding process, maintaining a stack of partially decoded bits to optimize the computation. By iteratively traversing this stack and cancelling out unreliable bits, this decoder achieves a low-complexity decoding process.

2.3. Creeper Decoding Concepts

The Convolutional Creeper approach merges aspects of stack-based decoding with the Fano algorithm. Traditionally applied to convolutional codes, the Fano algorithm employs a sequential decoding method within a code tree structure. During each decoding step, it navigates the tree, opting for either the parent predecessor node or a designated child node. This hybrid approach, integrating stack-based techniques and the Fano algorithm's sequential strategy, offers a nuanced decoding process for efficient error correction and data retrieval in communication systems.

The key components of Fano decoding include two constraints: the step size Δ and the dynamic threshold T. The search process continues along a code path as long as the Fano metric [13] increases.

Let us assume that for each i-th virtual channel, there is a channel error probability value p_i. SC-Fano employs a path metric called the Fano metric (PMF). It starts from $PMF_{-1} = 0$ and calculates iteratively as follows:

$$PMF_i = PMF_{i-1} + \log_2 \frac{\Pr(u_i|\mathbf{y}_0^{N-1}, \hat{\mathbf{u}}_0^{i-1})}{(1-p_i)}, \quad (2)$$

It measures the discrepancy between the likelihood of the current received symbol being associated with each possible codeword and the likelihood of obtaining the correct codeword. A smaller Fano metric indicates [14] a better match between the received symbol and the corresponding codeword. This metric is also applicable for code tree path pruning, which provides a more flexible decoding process.

It is worth noting that the choice of Δ significantly impacts SC-Fano's ability to move backward through the tree. A large Δ may limit backward movement, resembling basic SC decoding, while a very small Δ may improve the performance but hinder backward traversal.

Assuming that the current node is $v_i^{(t)}$, where t represents the level, SC-Creeper stores only the current node (address and value), its children and its ancestors only.

The stack $\mathcal{N}$ in SC-Creeper uses two mappings, $V : \mathcal{N} \to \nu$ and $Pos : \mathcal{N} \to p_\nu$, to obtain the header node's element and the position of node ν, respectively. When the PMF metric for a node falls below the threshold T, this node is labeled as "excluded". Conversely, if the PMF metric surpasses T, it is referred to as a "valid node". All of the actual valid nodes are stored in the $\mathcal{T}$ stack. In cases where both children of a node are valid, and the PMF metric of one child exceeds any other metric calculated thus far, this particular node is designated as a T-node. To manage these T-nodes, SC-Creeper employs a dedicated second stack.

3. Our Contribution

3.1. Cost Function

In the original version, the decoder moves along the branches of the code tree under the control of the dynamic threshold T. In the modified algorithm, similar control is carried out by the cost function Γ, which, in the general case, may not produce T-nodes. In this case, however, the equality always holds. The current movement of the decoder is controlled by the current cost $\gamma_i = \Gamma(v_i)$, where v_i is the number of the last (current) vertex. Thus, the current threshold may not change at every node of the branch being viewed, but only at some of them.

Let us assume that we have found two T-nodes such that each is an ancestor for the current vertex and their PMF metrics have not yet been updated. In this case, we will call such metrics the first and second thresholds.

Let us introduce the concept of a new and an old node. A node is considered new if it has never been visited before during the decoding process or was visited but did not have continuation (i.e., it was an open node). Otherwise, the node is considered old. During operation, the decoder identifies the nodes of the branch being viewed in accordance with the gradation—new or old. For any node, the inequality $\gamma_{i+n} < \gamma_n$ is always true: the values of the current thresholds for different visits to the node (the difference is i). It follows from this that, to identify a node, it is enough to know whether it was visited during the penultimate analysis of the current branch, and, if so, with which threshold.

Having monitored the operation of the Creeper decoder over time, we can draw the following conclusions. The decoder always either continues the branch with a metric not less than the metrics of previously viewed branches (if it is stored in the decoder memory), or restores the branch with the maximum metric (if it was previously viewed and trimmed). Until the moment at which any of the edges of the code tree begins to be viewed again, the operation of the decoder is similar to that of the SCS decoder, which affects the performance of the entire decoding.

The main reason is a serious consequence of the algorithm that can occur when moving through the same branches of the code tree repeatedly—when re-analyzing the code subtree rooted at the next T node, the decoder will repeat exactly the same operations as in the first pass. The developed cost metric allows us to avoid repeated passes and cycles, which slightly reduce the performance of the algorithm.

In Figure 1, it is shown that after defining two thresholds, the T-nodes will be updated to find nodes with a threshold value that becomes "old". In this case, a situation arises wherein the costs of two vertices a and b will coincide ($\Gamma(a) = \Gamma(b)$), since they are between two threshold values. After the PMF update process in the old node, however, the cost equality may not hold. The current thresholds for decoder control were indicated in red and T-Nodes in blue. Arrows indicate the transition from one vertex to another in the tree. Thus, the transition from the old threshold value corresponding to vertex a to the new threshold value at vertex b through the T-Node is possible due to the calculation of the cost metric at each movement step.

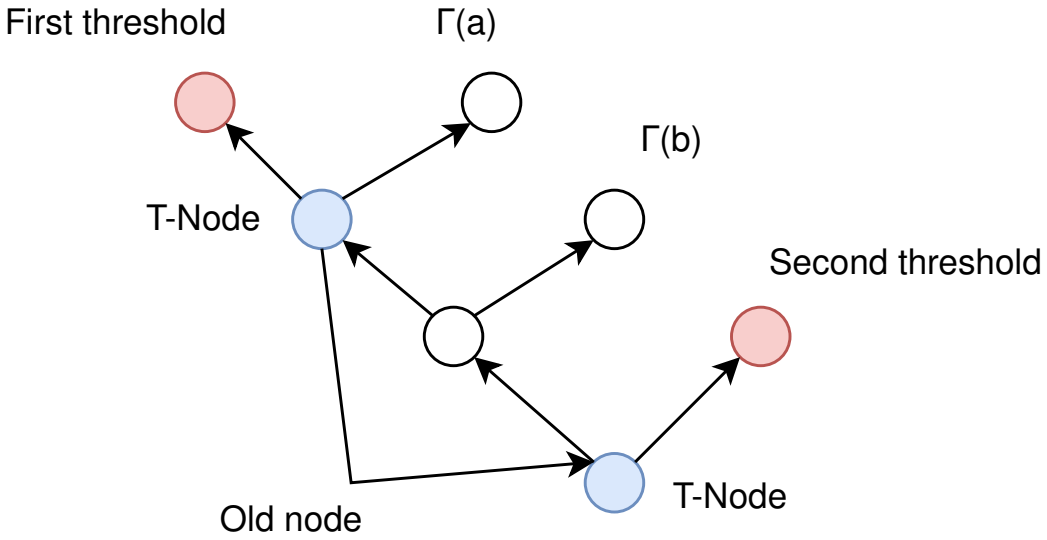

Figure 1. SC-Creeper scheme with cost metric Γ.

The cost function allows [15,16] the decoder to make informed decisions about which path to take in the decoding tree and to discard decoding paths that are deemed less promising. Paths with higher costs can be pruned (or bypassed, as we did in this study) to focus the computational resources on more likely paths, thereby improving the efficiency of the decoding process.

3.2. SC-Creeper and Additional Threshold

The recurrent procedure for the identification of nodes when moving forward is as follows. Let it be known for some node whether it is new or old. If a node is new, then all its descendants are, by definition, new nodes. If a node is old, then its descendants retain this gradation as long as their metrics are greater than or equal to the current μ'_i value (here, μ'_i is the current threshold with which node i was visited the penultimate time). The first child of a node whose metric has a value less than the cost of the given one μ'_i is new. Then, the recurrent procedure is repeated. As a criterion for changing the value at a vertex when moving forward, we can use one of the following conditions: either the parent vertex is a new node or a neighboring vertex in the tree, or the vertex itself will become a new node and update the cost by $+\infty$. Suppose that, for some node i, in addition to the metric μ'_i, the metric γ_i from the Γ-function is also known. Then, for the sibling's metric of the cost function, the following equation is correct:

$$\gamma_i' = \begin{cases} \infty, & i\text{-th node is new,} \\ \mu_i', & \text{otherwise.} \end{cases} \tag{3}$$

As a controlled functional Γ, it is necessary to consider models that satisfy the following relations:

- $\Gamma(l) = \Gamma(l') = T(l'), l' \leq l$ (maintaining the direction of movement of the decoder);
- $\Gamma(l) = -\infty, l < 0$ (positive certainty);
- $\Gamma(l) = \Gamma(l-1)$ if and only if the cost function is not updated (hence, $T(l) = T(l-1)$).

A detailed description of SC-Creeper with cost function is contained in the Algorithm 1. This algorithm is based on the basic SC-Creeper, but steps are specified that include manipulation of metrics and stack recalculation.

Algorithm 1: SC-Creeper with cost function

Require: y_0^{N-1}—received output LLRs, Δ—Creeper step parameter, $\Gamma(.)$—cost function definition, $T(.)$—threshold function definition;
Ensure: estimated codeword with the path ν_i.
ν_0—the root of tree, $\mu_0 = T(\nu_0), \Gamma(\nu_0) = \gamma_0$;
$\mathcal{N} = \mathcal{T} = \varnothing, i = 0, t = -\infty, t' = \infty$;
while $i < N$ or $\mu_i > t + \Delta$ **do**
  $i = i + 1$;
  Calculate PMF values μ_i, μ_i' and cost metrics γ_i, γ_i';
  Current node is ν_i: $t' = \gamma_i$;
  if $\gamma_i < t + \Delta$ **then**
    Update $\mathcal{T}$, $\mathcal{N}$ stacks with the basic Creeper algorithm;
    $push(\mathcal{T}, \mu_i')$, $push(\mathcal{N}, \gamma_i')$;
  else
    $push(\mathcal{T}, \mu_i)$, $push(\mathcal{N}, \gamma_i)$;
    if $-t' = \infty$ **then**
      $t = T(i)$;
    else
      $t = \Gamma(i)$;
    end if
  else
    Go to alternative node, i-th level: $\nu_{alt} = Pos(\mathcal{N})$;
    $t' = \gamma_i'$;
    if $T(i) < \gamma_{i-1} + \Delta$ **then**
      $t = \gamma_{i-1}$;
    end if
    Update $\mathcal{T}$, $\mathcal{N}$ stacks with the basic Creeper algorithm;
  end if
end while

When viewing the branches of the code tree once, the operation of the decoder completely coincides with decoding according to the original algorithm. With repeated passes of the same branches, the control thresholds are lowered (there is some similarity here with the adaptive Fano algorithm [17]). This eliminates the side effect of the original version.

At each iteration, updated Creeper decoding performs a local search within a neighborhood of the current decoding solution. This search explores nearby candidate solutions, evaluating their likelihood based on the received symbols and the properties of the error-correcting code. This version of the Creeper algorithm also provides a better "lookup" ability to return to vertices that the original Creeper would have already eliminated from its stack. It allows the decoder to backtrack to previous decision points and explore different branches of the decoding tree, increasing the likelihood of finding the correct solution.

Additional procedures for the refining of movement by nodes significantly refine the choice of the final path in the Creeper algorithm. Since backward tree movement in the case of polar codes is not as reliable a procedure as forward tree movement, an additional metric is introduced to control backward movement.

Thus, in addition to the more accurate decoding of nodes and a more complete traversal through the tree, it is possible to obtain optimization in two parameters at once.

- Number of operations. If, in the original version of decoding, the number of passes through the different nodes of the tree (as well as the T-node architecture as a whole) depended only on the step width Δ, then, here, it becomes possible to avoid unnecessary passes and cycles, reducing the number of operations (since the cost function will signal that the cyclic node does not need to be entered again).
- Performance. Since searching by the Fano metric requires a certain level of precision and an understanding of the "closeness" in decoding, the cost function can also point to those paths that contain the smallest value of the Γ metric. Recall that information about the old node is stored until the metrics for each possible path are calculated, i.e., at each stage, there is either a "bypass" (if the metric is too high) or a bypass. This allows us to construct the most probable codeword in the most accurate way.

Importantly, while the algorithm's sophistication and effectiveness are heightened, its memory consumption remains consistent. This is attributable to the utilization of the same two stacks employed in the original Creeper algorithm, ensuring the efficient use of the computational resources without necessitating additional memory overhead.

However, the decoding efficiency is strongly tied to how the threshold and cost functions are defined. Thus, for example, with an exponential threshold and linear cost functions, it is impossible to build an effective decoding model, since most nodes will not be considered during such a traversal (e.g., they will not satisfy the threshold condition).

3.3. Key Difference from Basic SC-Creeper Decoder

When considering SC-Creeper with a cost metric, one may consider the fundamental differences between the proposed approach and the basic SC-Creeper algorithm. In this subsection, we list the key differences in the two approaches.

1. Penalty for the wrong node: The basic SC-Creeper has the ability to move in the opposite direction, but backward movement does not always help to find the correct T-node or a valid node. In this case, an algorithm with an additional metric may decide not to use backward movement, but, for example, to continue moving in the direction of the largest LLRs.
2. Memory optimization: Although an additional metric must be calculated periodically, this action prevents the decoder from following a potentially incorrect path. Thus, instead of storing the stack of the entire path and recalculating it at a later stage, it is necessary to store only two values of the $\Gamma(\dot{})$ metric in order to control and optimize the variation of the stack.
3. Purpose of threshold values: For SC-Creeper, values such as Δ and T are the algorithm step and the degree of quantization when moving to a new tree node. However, in the modified version, the step plays a less significant role (that is, for different values of Δ, the performance variation is much lower), however, instead of the constant variable T, the function $T(\dot{})$ is used, which serves as an upper threshold value for the searching of T-nodes. A more flexible search allows us to achieve better performance results.

4. Simulation Results

Let us conduct a comprehensive performance evaluation of the proposed SC-Creeper decoder, comparing it with four other decoding methods: SC, SCL ($L = 8$), SCS ($L = 8$, $D = 1024$) and SC-Fano. The evaluation focuses on two critical aspects: performance and normalized complexity. To accurately simulate the channels, the authors use Additive

White Gaussian Noise (AWGN [18,19]) with Binary Phase Shift Keying (BPSK). Performance is evaluated with the Frame Error Rate (FER) metric.

For CRC checking, we utilize a polynomial with an additional 16 bits, defined as $g(x) = x^{16} + x^{15} + x^2 + 1$. A CRC code with 16 bits provides a higher level of error detection compared to shorter CRC codes. This is important in polar decoding because it helps to identify and correct errors more effectively. Moreover, it is a common choice in many communication standards and protocols. In this work, we study the 5G NR strategy [20] to perform polar codes and to construct a reliable set.

In the case of SC-Creeper, the parameter $\Delta = 1$ is chosen, so movement along the tree (in both directions) occurs in steps of approximately one vertex. For the additional cost function, the iterative procedure presented below is chosen:

$$\Gamma(i) = \begin{cases} -\infty, \ i \leq 0, \\ \Gamma(i-1) + 2, \ \text{if node is new}, \\ i, \ \text{otherwise}. \end{cases} \tag{4}$$

Moreover, for the first realization, we assume that $T \equiv \Gamma$, so the threshold function has the same properties and limitations as a cost function.

Next, in Figures 2–4, we provide the simulation result for codes with length $N = \{256, 512, 1024\}$, a high rate $R = \frac{3}{4}$, a moderate rate $R = \frac{1}{2}$ and a low rate $R = \frac{1}{4}$. The number of blocks to test the performance of the decoders is chosen to be 10^6, and the number of maximum failed decoding attempts at one SNR point is equal to 30.

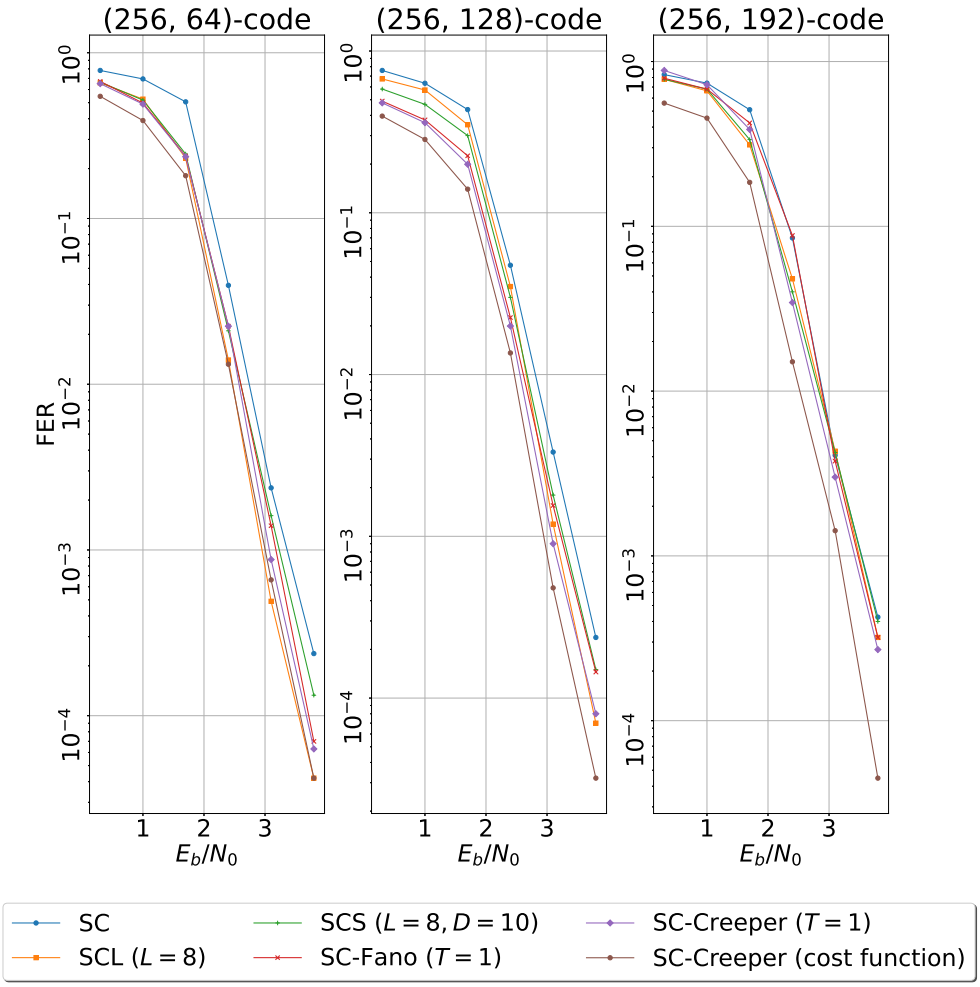

Figure 2. Comparison of FER values, $N = 256$.

SC-Creeper's performance is comparable to that of SCL with list size $L = 8$. However, using the additional cost metric improves the performance, so the modified SC-Creeper has a stronger correction ability. Thus, the option of using an algorithm without a list with two stacks appears to be more optimal in terms of performance (and also more optimal than the basic SCS decoder, which has only one stack). One can also notice that at low and moderate $R \leq \frac{1}{2}$, the modified Creeper algorithm, as well as its main version, perform worse than, for example, the SCL-8 decoder. This is due to the fact that a frozen set is growing, for which there is no clear definition of the threshold values and cost functions. It is possible that, with additional regularization at low rates, it will be possible to obtain higher performance, taking into account the modification of the frozen set $\mathcal{F}$.

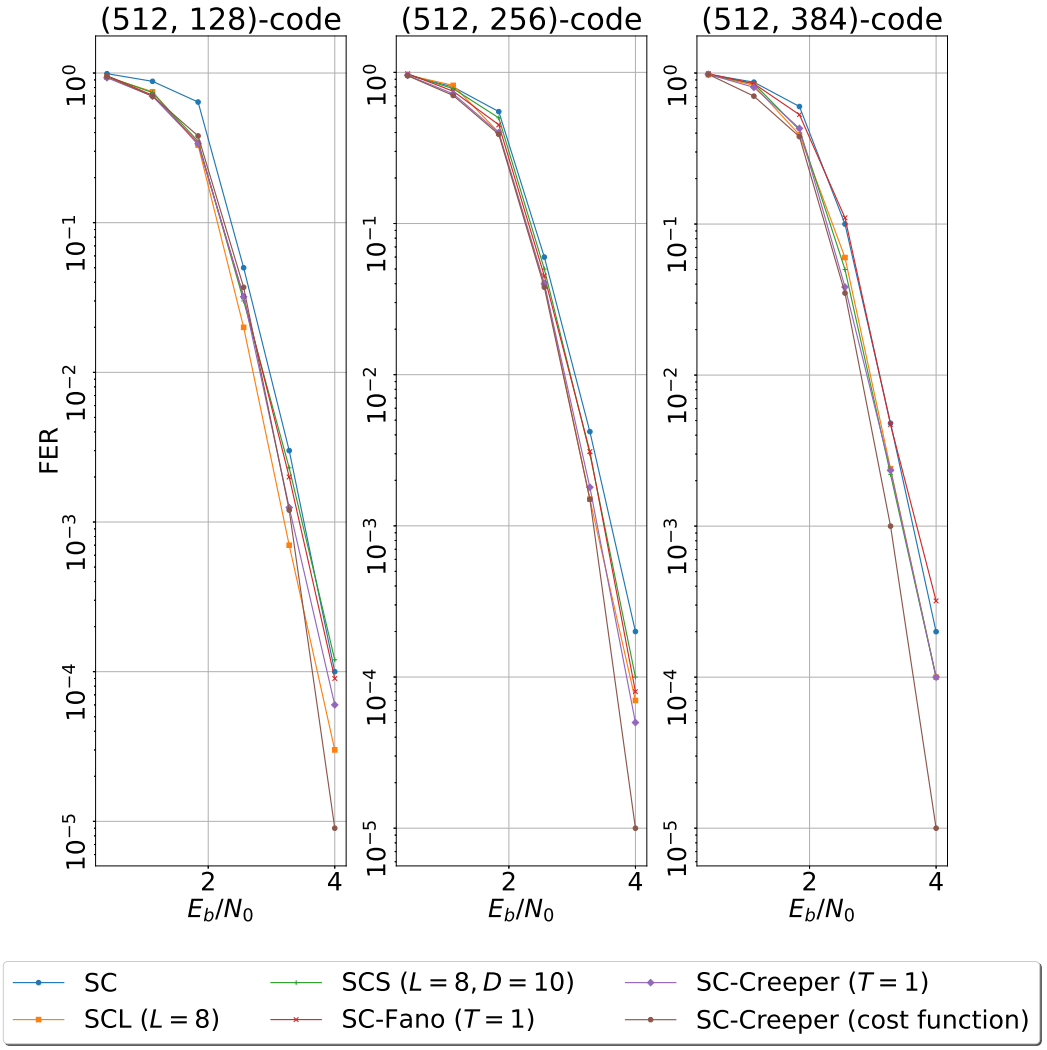

Figure 3. Comparison of FER values, $N = 512$.

Comparing the performance of various polar decoding algorithms in relation to the Frame Error Rate (FER) at different E_b/N_0 values provides valuable insights into their effectiveness across different scenarios. Let us analyze the characteristics and performance of each algorithm.

The basic SC algorithm serves as the foundation, offering decent performance but struggling with high code rates and long block lengths due to its sequential nature. SCL algorithms, such as SCL-8, maintain lists of candidate solutions, which is why they are effective across a range of code rates and block lengths, providing a good trade-off between complexity and performance.

SCS aims to simplify the decoding process while maintaining reasonable performance. It may sacrifice some decoding accuracy for reduced complexity, making it suitable for low-complexity applications.

SC-Fano introduces Fano metric-based decisions, improving upon SC but still falling short of SCL in some cases. SC-Creeper employs iterative refinement, particularly beneficial for high code rates, with potential complexity reductions compared to SCL.

The improved SC-Creeper builds upon the basic SC-Creeper algorithm, achieving the best performance at high code rates. While it may exhibit poorer performance at low code lengths and rates compared to SCL-8, it remains competitive and may surpass SCL-8 in certain cases. Despite its enhanced performance, the complexity of the improved SC-Creeper is comparable to that of its basic version, with potential reductions in operations for specific scenarios, leading to an overall complexity reduction.

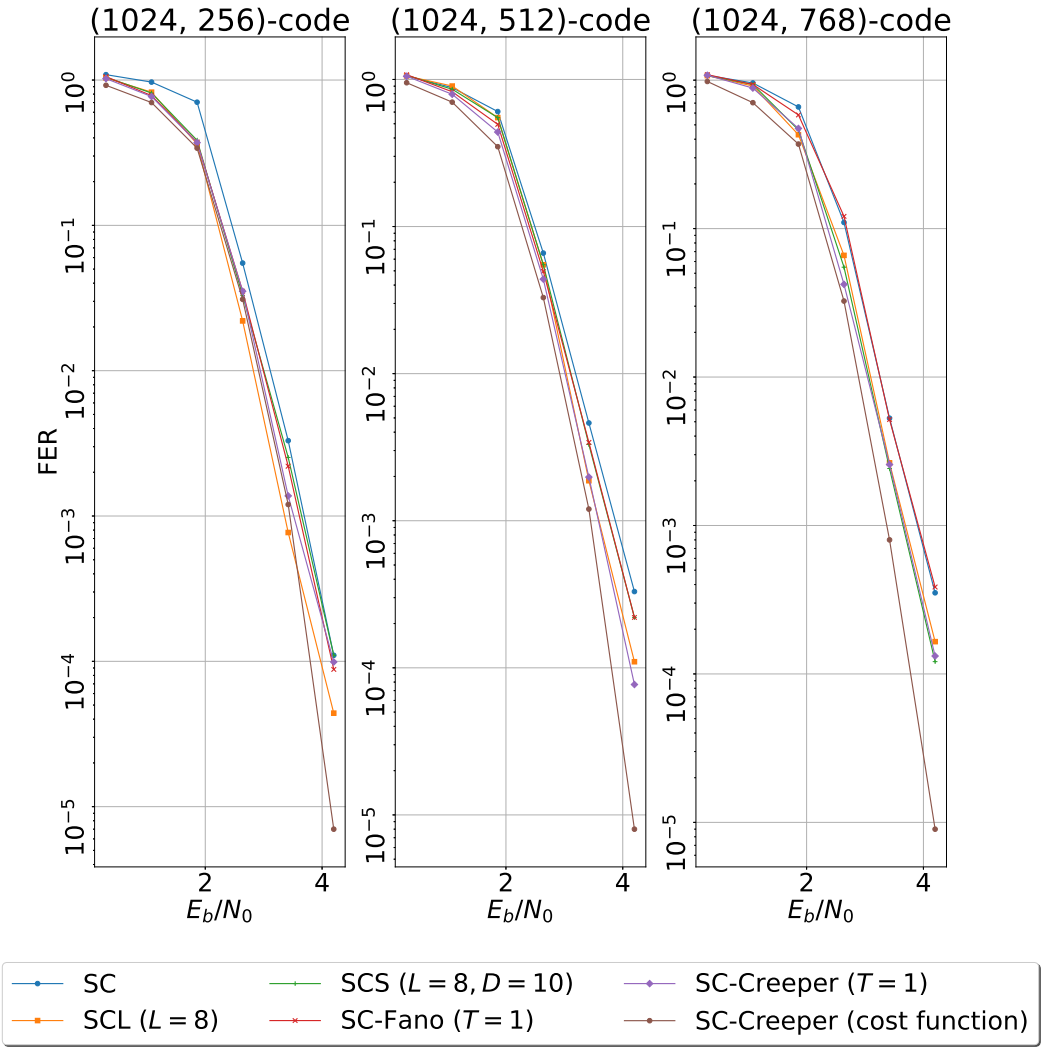

Figure 4. Comparison of FER values, $N = 1024$.

In summary, the performance of polar decoding algorithms varies based on factors such as the code rate, block length and complexity constraints. While SCL-8 remains a strong performer across a wide range of scenarios, algorithms like SC-Creeper and its improved version excel in high-code-rate regimes, offering superior performance with comparable complexity. However, for low code lengths and rates, SCL-8 and SCL-8-like algorithms may still maintain a competitive edge, with SC-Creeper variants exhibiting comparable performance.

Figures 5–7 show the normalized complexity plots for all considered decoders, including the new SC-Creeper. Complexity refers to the relative number of operations required to successfully complete a decoding algorithm.

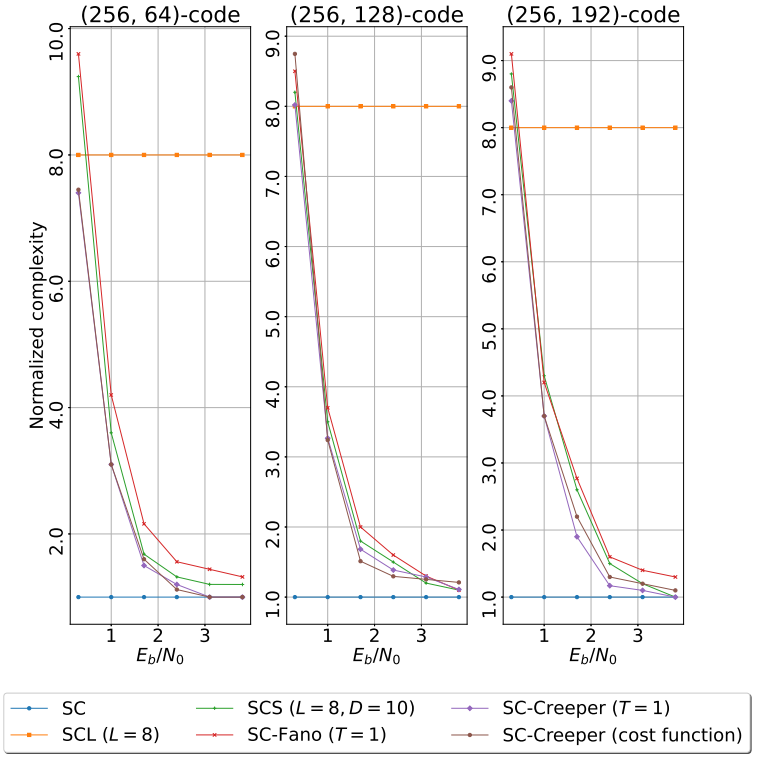

Figure 5. Comparison of normalized complexity.

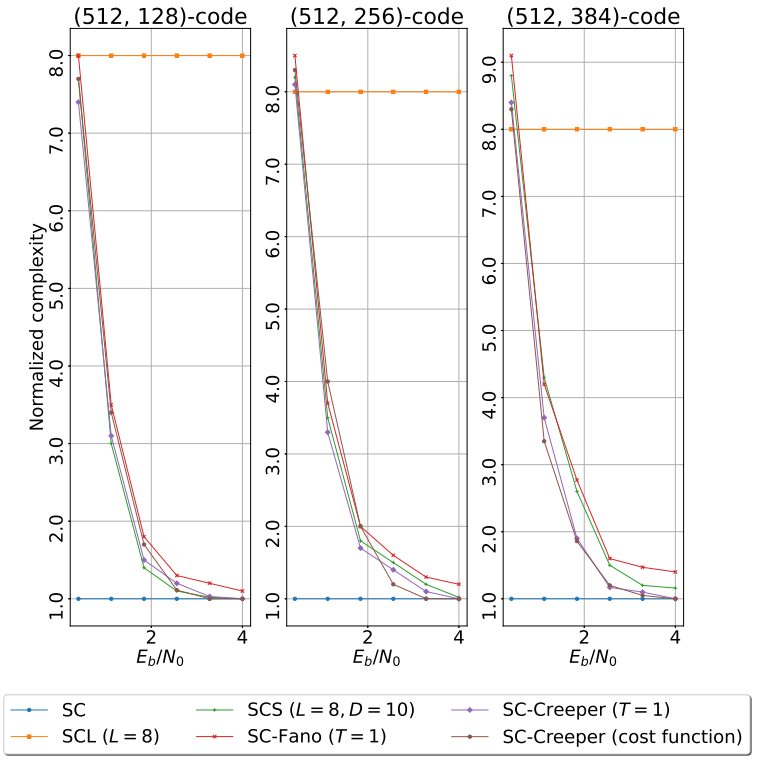

Figure 6. Comparison of normalized complexity.

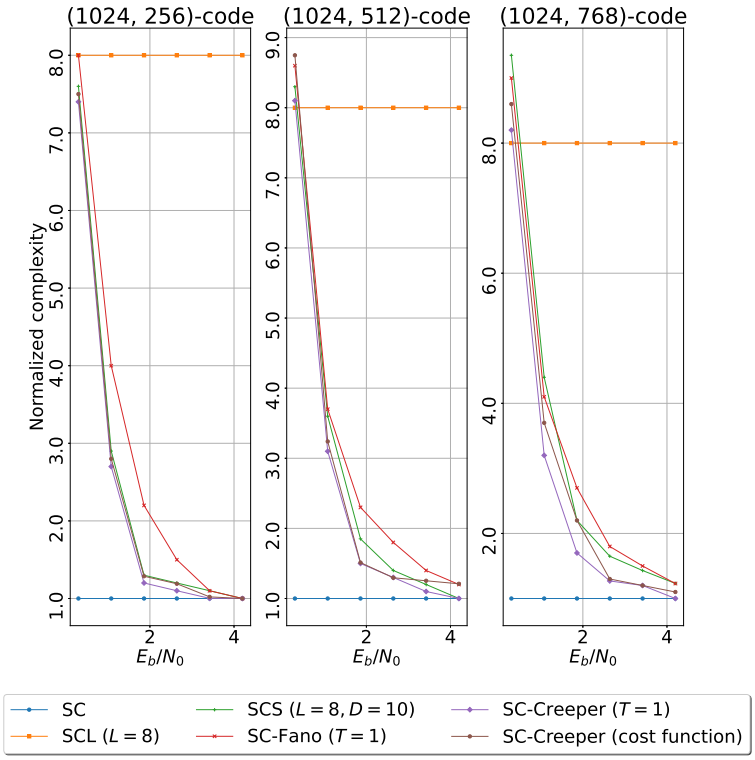

Figure 7. Comparison of normalized complexity.

It can be noted that all methods are "between" SC (in other words, SCL-1) and SCL-8 in complexity. Thus, as E_b/N_0 increases, we can say that the complexity of the SC-Creeper approaches is comparable to that of an SC decoder using $O(N \log N)$ operations.

However, the modified SC-Creeper uses slightly more operations due to the application of additional conditions and different stopping criteria compared to the original SC-Creeper algorithm. Nonetheless, for $\Delta = 0$, both algorithms have similar complexity and are identical.

5. Conclusions

In conclusion, our research has yielded significant advancements in the realm of polar code decoding, culminating in the development of the improved SC-Creeper algorithm. Through extensive experimentation and rigorous analysis, we have demonstrated the efficacy of our approach in achieving notable performance improvements over conventional decoding techniques. The enhanced SC-Creeper decoder not only rivals but surpasses the capabilities of established methods such as CA-SCL-8 and traditional Fano decoding algorithms. For large code lengths, the improvement in performance is obvious for any length of the original message; the performance can not only overcome the value of SCL-8, but also approach SCL-16. Thus, with the help of additional control functions (threshold, cost), we can hypothetically increase the list of potential candidates for the correct codeword from 8 to 16.

The new SC-Creeper uses a classic tree-moving scheme, as in the sequential Fano decoding strategy. However, this scheme has been improved with the ability to traverse only those nodes in the tree that lead to the most likely codeword. The cost function also allows us to ignore repeated passes of nodes, which was unacceptable in the original SC-Creeper.

Our research is characterized by the application of innovative methods, including iterative refinement techniques and adaptive cost metric calculations, which contribute to the enhanced performance of the decoder. Through comprehensive experimental validation, we have meticulously assessed the decoder's effectiveness across a range of communication scenarios.

It can also be noted that decoding based on the structure of the tree and its modifications is a promising direction nowadays [21,22]. The development of more sophisticated decoding strategies, coupled with enhanced computational efficiency and scalability, will not only benefit high-throughput communication systems but also find application in a wide array of fields, ranging from wireless communication to data storage and beyond.

6. Patents

The study was implemented in the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) in 2024.

Author Contributions: Conceptualization, F.I.; methodology, F.I.; software, I.T.; formal analysis, I.T.; investigation, F.I.; writing—original draft, I.T.; writing—review and editing, F.I. and I.T.; visualization, I.T.; project administration, F.I. All authors have read and agreed to the published version of the manuscript.

Funding: The study was implemented in the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) in 2024.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All project source code and decoding platform are developed by HSE University. Fragments responsible for Creeper decoding from this work can be provided upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Arikan, E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inf. Theory* **2009**, *55*, 3051–3073. [CrossRef]
2. Zhou, L.; Zhang, M.; Chan, S.; Kim, S. Review and Evaluation of Belief Propagation Decoders for Polar Codes. *Symmetry* **2022**, *14*, 2633. [CrossRef]
3. Tal, I.; Vardy, A. List decoding of polar codes. In Proceedings of the 2011 IEEE International Symposium on Information Theory Proceedings (ISIT), Petersburg, Russia, 31 July–5 August 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1–5.
4. Aurora, H.; Condo, C.; Gross, W.J. Low-complexity software stack decoding of polar codes. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–5.
5. Jeong, M.O.; Hong, S.N. SC-Fano decoding of polar codes. *IEEE Access* **2019**, *7*, 81682–81690. [CrossRef]
6. Johannesson, R.; Zigangirov, K.S. *Fundamentals of Convolutional Coding*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
7. Nyström, J.; Johannesson, R.; Zigangirov, K. Creeper—An algorithm for sequential decoding. In Proceedings of the Sixth Joint Swedish-Russian International Workshop on Information Theory, Molle, Sweden, 22–27 August 1993.
8. Ivanov, F.; Chetverikov, I.; Kreshchuk, A.; Timokhin, I. Successive Cancellation Creeper Decoding of Polar Codes. In Proceedings of the 2022 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), Yekaterinburg, Russia, 11–13 November 2022; pp. 60–64. [CrossRef]
9. Timokhin, I.; Ivanov, F. On the improvements of successive cancellation Creeper decoding for polar codes. *Digit. Signal Process.* **2023**, *137*, 104008. [CrossRef]
10. Zhang, D.; Wu, B.; Niu, K. Path Metric Inherited SCL Decoding of Multilevel Polar-Coded Systems. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Nanjing, China, 29 March 2021; pp. 1–6.
11. Niu, K.; Chen, K. CRC-aided decoding of polar codes. *IEEE Commun. Lett.* **2012**, *16*, 1668–1671. [CrossRef]
12. Niu, K.; Chen, K. Stack decoding of polar codes. *Electron. Lett.* **2012**, *48*, 695–697. [CrossRef]
13. Jiang, Y.; Zhang, L. A Fano Decoding for Polar Codes Based on Node Acceleration. In Proceedings of the 2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall), Hong Kong, China, 10–13 October 2023; pp. 1–5. [CrossRef]
14. Han, Y.; Chen, P.N.; Fossorier, M. A generalization of the fano metric and its effect on sequential decoding using a stack. In Proceedings of the IEEE International Symposium on Information Theory, Espoo, Finland, 26 June–1 July 2002; p. 285. [CrossRef]
15. Liberatori, M.C.; Arnone, L.J.; Castiñeira Moreira, J.; Farrell, P.G. Soft Distance Metric Decoding of Polar Codes. In Proceedings of the Cryptography and Coding, Oxford, UK, 15–17 December 2015; Groth, J., Ed.; Springer: Cham, Switzerland, 2015; pp. 173–183.
16. Chen, H.; Xu, R.; Bai, B. Path Metric Range Based Iterative Construction Method for Polar Codes. In Proceedings of the 2023 International Conference on Wireless Communications and Signal Processing (WCSP), Hangzhou, China, 2–4 November 2023; pp. 305–310.

17. Feng, Z.; Liu, L. On the adaptive Fano-SC-Flip Decoding of Polar Codes. In Proceedings of the 2022 IEEE/CIC International Conference on Communications in China (ICCC), Foshan, China, 11–13 August 2022; pp. 226–231. [CrossRef]
18. Vangala, H.; Viterbo, E.; Hong, Y. A comparative study of polar code constructions for the AWGN channel. *arXiv* **2015**, arXiv:1501.02473.
19. Dai, J.; Niu, K.; Si, Z.; Dong, C.; Lin, J. Does Gaussian Approximation Work Well for The Long-Length Polar Code Construction? *IEEE Access* **2017**, *5*, 7950–7963. [CrossRef]
20. Ercan, F.; Condo, C.; Hashemi, S.A.; Gross, W.J. On error-correction performance and implementation of polar code list decoders for 5G. In Proceedings of the 2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 3–6 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 443–449.
21. Lu, Y.; Zhao, M.M.; Lei, M.; Zhao, M.J. Fast Successive-Cancellation Decoding of Polar Codes with Sequence Nodes. *IEEE Trans. Green Commun. Netw.* **2024**, *8*, 118–133. [CrossRef]
22. Yao, X.; Ma, X. A Balanced Tree Approach to Construction of Length-Flexible Polar Codes. *IEEE Trans. Commun.* **2024**, *72*, 665–674. [CrossRef]