A General and Flexible Channel Decoding Approach Based on Probabilistic Programming Language

Mohammad Sadegh Rostami[1], Ali Shahzadi[*1], and Morteza Dorrigiv[1]

Abstract-- **Channel coding is a vital component within digital telecommunications, helping to deal with unwanted factors like noise and enabling the establishment of a more robust communication link. Among the most renowned coding schemes are linear block codes, for which a variety of decoding methods have been proposed in recent years. This paper demonstrates how a linear block coding problem can be expressed as a Probabilistic Graphical Model (PGM). We then explain how Probabilistic Programming Languages (PPLs), which are tools for solving such PGMs, can be used to decode this type of coding. Employing the Figaro programming language, as a PPL, we have simulated the decoding of several famous linear block codes and found that the results of our proposed method closely match those of existing techniques. Our approach offers several advantages, such as the flexibility to utilize diverse inference methods, the ability to choose between hard and soft decoding dynamically, and the implementation of a wide range of coding techniques. PPLs also enable the adjustment of decoding algorithm parameters and the estimation of channel conditions, ultimately enhancing the receiver's adaptability to varying channel conditions. Finally, we discuss the advantages and disadvantages of our proposed method.**

Index Terms-- **Channel Decoding, Linear Block Code (LBC), Probability, Probabilistic Graphical Model (PGM), Probabilistic Programming Language (PPL), Bit-Error Rate (BER). Nomenclature**

I. Introduction

Information theory and coding constitute a crucial field within telecommunications, primarily focused on reducing data transmission errors [1]. Telecommunications systems employ different resources such as frequency, time, and code, each of which has inherent limitations necessitating optimization strategies for effective resource utilization. By employing data coding and decoding techniques, telecommunication systems strive to optimize some resources utilization. Ensuring error-free transmission and reception of data is of paramount importance in these systems. However, factors such as noise, limited transmitter power, and environmental conditions introduce errors, prompting numerous studies aimed at minimizing data errors [2]. Channel coding is one of the methods used to reduce these errors and to deal with them. Over the years, several methods have been introduced for encoding and decoding data [1,3][1].

Decoding methods in telecommunications exhibit a wide range of diversity and can be implemented using FPGA, ASIC circuits, software programs, and other approaches [4,5]. Numerous techniques have been employed for data decoding, including Belief Propagation (BP), Junction Tree (JT), Successive Cancellation (SC), Trellis, and more [1,3,6–8]. However, existing decoding methods suffer from certain limitations, such as:

- Utilization of only one or two decoding methods [9],
- Sole reliance on either hard or soft decoding algorithms [7,10],
- Implementation of only a limited subset of available decoders [9,11,12],
- Limited flexibility in modifying decoding parameters [4],
- Default probabilistic parameter settings, among others.

In this paper, we propose a novel approach to address these limitations in linear block codes. We introduce a general and flexible decoding method based on Probabilistic Programming Languages (PPL).

Some decoders can be represented as Probabilistic Graphical Models (PGMs) [4,13]. These decoders consist of interconnected nodes and edges, forming a PGM that captures the decoding problem. Since communication channels introduce random errors, decoding can be formulated as a PGM inference problem. Probabilistic Programming Languages (PPLs) have emerged as a field of study for accurate inferences in graphical networks, supporting both Bayesian and Markov graphs, referred to as directional and non-directional graphs, respectively. Figaro is one such PPL, implemented as a functional, object-oriented library in Scala [14,15].

Linear block codes, such as Hamming, LDPC, and polar codes, can be represented as probabilistic graphs [4,6,16,17]. Decoding methods for these codes often leverage graph and probability theory, making PGM-based methods suitable for decoding. Despite the various capabilities of PPLs, our studies indicate that they have not been extensively utilized in communication channel decoding. The novelty of our paper lies

[1] The term coding/decoding in this paper refers to "channel coding/decoding", which is described in Section *II*.

1-Faculty of Electrical and Computer Engineering, Semnan University, Semnan, Iran.

Corresponding author: shahzadi@semnan.ac.ir

in the application of PPLs for decoding, aiming to open new horizons in channel decoding based on PPLs. Furthermore, we demonstrate the versatility of PPLs in addressing various aspects of channel coding.

Our goal in this paper is to demonstrate how a linear block coding problem can be executed using a probabilistic programming language. Since this paper is interdisciplinary research and some of the esteemed readers may not have a telecommunications background, we have provided some coding theory details. These details help us better understand how linear block decoding is performed with the help of PPL. According to the simulations we have conducted, using PPL for linear block decoding does not improve the error rate. However, it brings along a wide range of other applications for us.

The remainder of this paper is structured as follows: Section *II* provides an overview of communication channels, their diagrams, and channel coding/decoding. Section *III* introduces linear block codes, probabilistic programming languages, and briefly highlights their strengths. We specifically present Figaro briefly as an example of a PPL. The objective of Sections *II* and *III* is to illustrate how a linear block coding problem can be transformed into a probabilistic graphical problem. In Section *IV*, we present our proposed method for utilizing PPLs in data decoding. Section *V* presents the results obtained from implementing the proposed method on Hamming codes across different channels. Section *VI* discusses the advantages and disadvantages of the proposed method. Finally, Section *VII* concludes the paper by providing closing remarks and suggestions.

II. BACKGROUNDS: COMMUNICATION CHANNEL, CODING AND DECODING

Over the past three decades, the advancement of digital processors has revolutionized the field of telecommunications, transitioning it from analog to digital systems [18]. This shift has been driven by the utilization of digital signal processing capabilities, resulting in the majority of communications being conducted in the digital domain. Fig. 1, provides an overview of the block diagram of a digital telecommunications system [1]. In this block diagram, the source data, typically in analog form, is initially converted to digital format. It then proceeds through several blocks in the transmitter, including the source encoder, channel encoder, and modulator, before being transmitted. At the receiver, the demodulator receives the transmitted data, demodulates it, and passes it to the channel decoder. Subsequently, it undergoes source decoding, and finally, the received data is converted back to analog using a digital-to-analog converter (DAC). If the source data is inherently digital, the analog-to-digital converter (ADC) and digital-to-analog converter (DAC) blocks can be omitted from the diagram.

In the following subsections, we will provide a brief explanation of some of the blocks depicted in Fig. 1, and elaborate on how noise introduces a probabilistic problem at the receiver.

A. Modulator and Demodulator

Modulation and demodulation techniques are utilized to effectively utilize frequency space, reduce antenna length in wireless systems, and increase the number of users in a given medium [18]. Digital telecommunication systems employ various modulation methods, primarily categorized based on frequency, amplitude, and phase [18]. Some applicable modulation schemes include M-ary Quadrature Amplitude Modulation (M-QAM), M-ary Phase Shift Keying (MPSK), Quadrature Phase Shift Keying (QPSK), and Code Division Multiple Access (CDMA) [2,19].

A key parameter used in modulation analysis is the bit error probability at the receiver, which is typically a function of the main signal power to noise signal power ratio (SNR). A higher SNR indicates a higher quality signal. Equations (1), (2), and (3) calculate the probability of bit errors in the receiver for QPSK, MPSK, and M-QAM modulations in an Additive White Gaussian Noise (AWGN) channel. These equations demonstrate how the received signal is transformed into a probabilistic signal during its passage through the channel in the presence of noise [20].

$$P_b = Q(\sqrt{2\gamma_b}) \tag{1}$$

$$P_b \approx \frac{2}{Log_2^M} Q(\sqrt{2\gamma_b\, Log_2^M}\, Sin\frac{\pi}{M}) \tag{2}$$

$$P_b \approx \frac{4}{Log_2^M} Q(\sqrt{\frac{3\overline{\gamma_b}\, Log_2^M}{M-1}}) \tag{3}$$

In these equations, $Q(0)$ represents the evaluation of the Q-function as defined in (4), and γ_b denotes the ratio of signal energy to noise energy at the receiver. M indicates the number of transmitted symbols. By determining the value of M based on the modulator's structure and calculating γ_b based on the channel conditions, the error probability at the receiver can be computed. The determination of whether a sent bit corresponds to 0 or 1 is not a deterministic value but is obtained with a probability within the range of [0, 1].

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2}\, dt = \frac{1}{2} erfc(\frac{x}{\sqrt{2}}) \tag{4}$$

Once again, we emphasize that the objective of this paper is to demonstrate how a linear block decoding problem can be executed using probabilistic programming languages. Therefore, while equations (1) to (4) have not been directly utilized in the subsequent sections of this paper, their presentation aids in better understanding how linear block decoding is transformed into a probabilistic problem. Understanding this concept helps us articulate one of the main ideas behind utilizing PPL for linear block decoding more effectively.

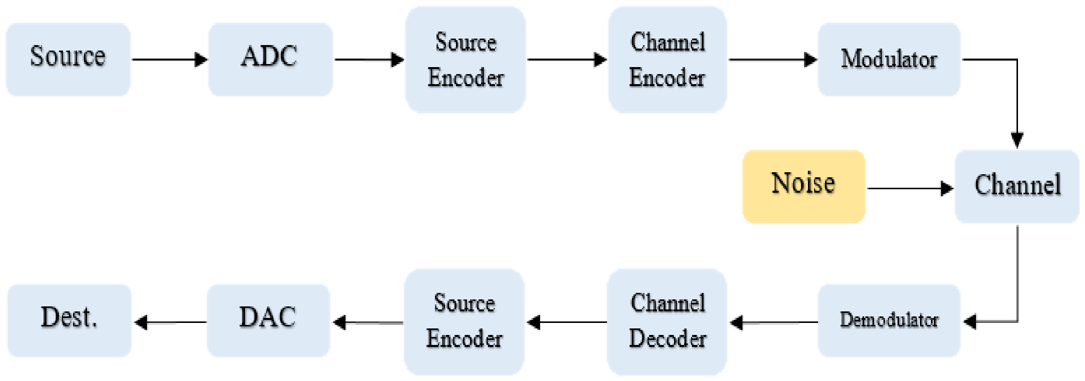

Fig. 1. Digital Communication Block Diagram

B. Channel and Noise

The channel model is determined by considering the characteristics of the channel itself, the type of noise present, and the models employed to describe error probabilities. In wireless channels, well-known models such as Rayleigh, Rician, and Nakagami are utilized. On the other hand, the Binary Symmetric Channel (BSC) and Binary Erasure Channel (BEC) find common use in twisted-pair wire transmission environments. The Binary AWGN (BAWGN) model applies to both wireless and wired channels. To provide further clarification, (5) presents the Probability Density Function (PDF) of SNR in the Rayleigh channel. In this equation, γ_0 represents the SNR of the AWGN channel, σ^2 denotes the noise variance, and *x* represents a random variable [20].

$$f(x) = \frac{1}{2\gamma_0\sigma^2}\exp\left(\frac{-x}{2\gamma_0\sigma^2}\right); x \geq 0 \tag{5}$$

According to this equation, each received bit at the receiver is subject to a probabilistic error, and the magnitude of this error depends on the SNR.

C. Channel Coding

Coding theory is commonly applied in telecommunications, encompassing four main areas: source coding, channel coding, data encryption, and line coding [1,3,21,22]. Channel coding plays a crucial role in addressing the aforementioned errors. Various coding/decoding methods exist, each designed to detect or correct specific channel errors. For instance, Reed-Solomon codes are employed for burst error detection and correction in satellite communications [23] while LDPC (Low Density Parity Check) codes are utilized in 10GBase-T Ethernet [16,17].

The Bit Error Rate (BER) is a significant criterion for evaluating coding performance. A lower error rate is preferable for a given SNR. In practice, the BER typically ranges from 10^{-1} to 10^{-7} [24]. However, this value can vary according to different applications and conditions. Coding complexity is another criterion used for evaluation.

It is important to note that various decoding approaches may be available for a particular coding method. Therefore, selecting a method with lower delay, complexity, and energy consumption is desirable, although achieving all of these criteria simultaneously may not be possible.

III. Linear Block Code and Probabilistic Programming Language

In this section, we provide a brief overview of linear block codes, specifically focusing on the Hamming code as an illustrative example. We then proceed to provide a concise explanation of PPLs, with a particular emphasis on Figaro as a PPL language. The objective of this section is to demonstrate how a linear block code can be represented as a PGM and how Figaro can operate as a PPL.

A. Linear Block Coding[2]

Linear block codes belong to a category of codes used in information theory and coding. The term "block" signifies that the encoder processes multiple input bits together, and the decoder decodes them as a group. The term "linear" indicates that the sum of any two valid codewords will also be a valid codeword. These codes are created using a generator matrix ($\boldsymbol{G}$). If the number of input bits in the encoder block is k and the number of output bits is N, the dimension of the $\boldsymbol{G}$ matrix is $k \times N$ $(where\ k < N)$. The relationship between the input bits ($\boldsymbol{d}$) and the output codeword ($\boldsymbol{c}$) is represented by (6).

$$\boldsymbol{d}_{1\times k} \cdot \boldsymbol{G}_{k\times n} = \boldsymbol{c}_{1\times n} \tag{6}$$

Some well-known linear block codes include Hamming, Reed-Muller, BCH, LDPC, and Reed-Solomon, among others. Additionally, Convolutional codes are not inherently block codes, but they can be treated as block codes when they operate on fixed-length blocks of input data.

Hamming codes are particularly popular and are represented as $[2^r - 1. 2^r - r. 3]$. The expression $2^r - 1$ indicates the number of bits in the decoder output, while $2^r - r$ denotes the number of input bits. The minimum distance of 3 indicates the code's ability to detect two bits of error and correct one bit of error. For example, when $r = 3$, $N = 7$, and $k = 4$. Fig. 2, illustrates an example of the $\boldsymbol{G}$ matrix for $r = 3$, while Fig. 3, depicts the graphical representation of (6) for this example. Different linear block codes employ various methods to generate the $\boldsymbol{G}$ matrix.

[2] Most of the content in this section, and its two sub-sections, are adapted from [1,3] and are interpreted by the authors.

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Fig. 2. ***G*** matrix for a specific given Hamming (7,4)

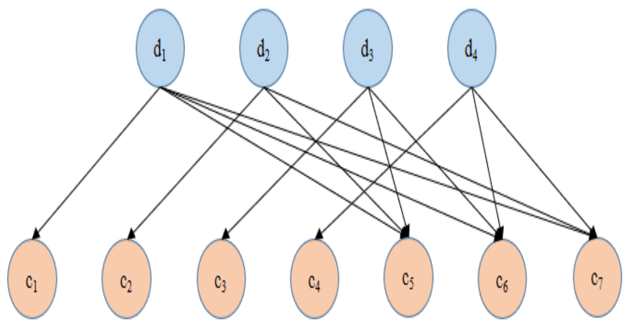

Fig. 3. Relationship between input data of an encoder and corresponding output using the ***G*** matrix in Fig. 2

B. *Probabilistic Programming Language*

In many real-world problems, uncertainty plays a significant role, resulting in the presence of random values and probabilistic parameters. Probabilistic Reasoning Systems (PRSs) offer effective approaches to tackle such problems. When there is a general understanding of the problem, the components of the problem can be reasoned about using logic.

However, as the complexity of the problem increases and additional contextual or auxiliary data becomes available, direct inference using traditional PRS methods becomes challenging or even infeasible. Fig. 4, depicts the model of a probabilistic reasoning system, which comprises the following components:

i. Probabilistic Model: This component represents the general knowledge about a particular situation. It incorporates the probabilistic relationships and dependencies among variables.

ii. Evidence: refers to the information and observations available about the situation. It provides input to the reasoning system and influences the answers to queries.

iii. Queries: Queries represent the questions or specific aspects we seek to know or infer about the situation.

iv. Inference Algorithm: The inference algorithm utilizes the probabilistic models and the available evidence to determine the answers to the queries. It performs the necessary computations and reasoning steps to make probabilistic inferences.

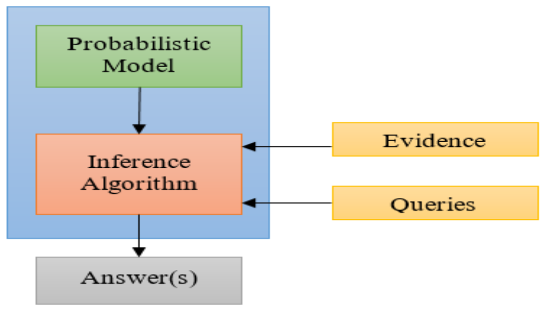

Fig. 4. Probabilistic Reasoning System Model [14]

PRSs offer flexibility and find applications in three key areas [14]: (1) Predicting future events based on the available knowledge and evidence, (2) Inferring causes of an event that enables reasoning about the underlying causes behind observable phenomena, and (3) Learning from past events to improve their predictive capabilities for the future.

By incorporating historical data and observations, the system can refine its probabilistic models and enhance its accuracy. Two important tools commonly used in PRS are Bayesian Networks (or belief networks) and Hidden Markov Models [14].

1) *Figaro*

Based on the investigations we have conducted, over 50 probabilistic programming languages have been introduced in the past decade. The diverse applications of these languages have led research teams in various fields to develop different probabilistic programming languages [25]. Major companies like Google, Amazon, and Microsoft utilize these programs for various applications such as video recommendations, software troubleshooting, cyber-attack prediction, and more [14]. Some notable PPL projects include seismic analysis as a global seismo-acoustic bulletin [26], cognitive sciences research [27], malware detection in systems [28], evaluating PPLs for simulating quantum correlations [15] and using probabilistic graphical models (PGMs) for solving biological network problems [29].

Figaro[3] is a prominent example of a PPL. Its name is derived from Wolfgang Amadeus Mozart's opera, "The Marriage of Figaro." Figaro is implemented as a library in Scala and has been under development since 2009 by Avi Pfeffer and his colleagues. It is compatible with IDEs like IntelliJ IDEA. Practical projects using Figaro include space object identification, target tracking, malware analysis, and soil drainage prediction, among others, developed by Pfeffer and his team.

One of the strengths of PPLs is their ability to model complex probabilistic graphical problems that are challenging to solve using conventional methods. As shown in Fig. 4, to utilize a PPL like Figaro, users need to develop a probabilistic graphical model in collaboration with domain experts. These models often take the form of directed graphs with parent-child nodes or undirected Markov graphs, with or without hidden layers. Observations are then applied to the model, and one or more inference algorithms are used to provide answers to specific queries. Fig. 5 illustrates the steps involved in using Figaro as a probabilistic programming language.

Figaro employs various elements, with the two main types being Atomic and Compound. Typically, queries in Figaro revolve around future event probabilities or causal relationships.

Figaro, as a probabilistic programming language, can be executed in two ways: line by line or within a main method. However, there are certain limitations when constructing models in Figaro. For instance, some functions have constraints on the number of inputs or variables they can handle.

[3] The contents of this section are derived from [14].

```
- Importing libraries
- Main loop:
  - Defining and/or declaring probabilistic and non-
    probabilistic variables
  - Making probabilistic graphical model via functions and
    methods
  - Applying observations and/or conditions and/or
    constraints
  - Calling inference algorithm method via libraries
  - Applying inference algorithm to queries
  - Printing or storing results
- End of loop
```

Fig. 5. Figaro Programming Language Model

Figaro incorporates various elements to define probabilistic models. Some of the important elements are as follows: (1) Atomic and discrete elements: These include elements like Select and Flip. For example, Flip (0.4) represents a variable that, in each program run, has a 0.4 probability of being true and a 0.6 probability of being false, (2) Atomic and continuous elements: These include elements like Uniform and Normal. For instance, Uniform (10,30) represents a variable that generates a decimal number within the range of [10, 30] with equal probability, (3) Compound and discrete elements: Figaro also provides compound elements with discrete values, such as Select and Flip, and (4) Compound and continuous elements: Similar to atomic elements, compound elements can also have continuous distributions. Functions like Uniform and Normal can be used, where the input is a probabilistic value rather than a fixed number. For example, a = Uniform (0,1) and b = Flip (a) define a variable 'a' that generates a decimal number within the range of [0,1] with equal probability. The value of 'a' then serves as the input to the function 'b', which outputs true with a probability equal to 'a'.

Other important elements in Figaro are used to establish relationships between variables in the probabilistic graph model. Elements like Chain and Apply are employed to connect variables in directional graphs and define parent-child relationships. Conditional Probabilistic Distribution (CPD) and RichCPD are used to determine the probabilistic relationship between multiple variables.

Figaro provides two categories of inference algorithms: factored and sampling, each comprising multiple algorithms. The Variable Elimination (VE) algorithm is an accurate factored method that infers based on the moral graph and systematically eliminates variables. Belief Propagation (BP) is another algorithm that is faster than VE but less accurate. It operates using a message passing algorithm. Sampling Algorithms (SA) utilize sampling theories and central limit theorems to provide approximate inference with adjustable accuracy. They are generally faster than factored methods. Figaro incorporates additional elements, functions, and methods, which are described in detail in [14].

IV. Proposed Method: Linear Block Code Decoding by PPL

In this section, our objective is to showcase the transformation of a linear block coding problem into a PGM. We will also elucidate how the structure of linear block codes aligns with the framework of probabilistic problems. Furthermore, we have demonstrated how the Figaro language can implement this PGM model as a PPL.

Based on Section *III* and the block diagram of the digital telecommunications system in Fig. 1, the input data to the channel encoder block is a bit string $[d_1\, d_2 \ldots d_k]$, where d_i takes on values of 0 or 1 for i ranging from 1 to k. The output of this block is denoted as $[c_1\, c_2 \ldots c_N]$, where c_i takes on values of 0 or 1 for i ranging from 1 to N. Here, N and k are integers, with $k < N$. For convenience, we consider d_i and c_i to be binary. At this point, we will ignore the modulator and demodulator blocks. The coded data passes through the channel, where additive noise is introduced. The receiver string is represented as $[\widetilde{c_1}\, \widetilde{c_2} \ldots \widetilde{c_N}]$, where $\widetilde{c_i} = c_i + n_i$ for i ranging from 1 to N, and n_i represents the channel noise.

In the receiver, the bit string $[\widetilde{d_1}\, \widetilde{d_2} \ldots \widetilde{d_k}]$ is extracted from the bit string $[\widetilde{c_1}\, \widetilde{c_2} \ldots \widetilde{c_N}]$ using decoding methods. Techniques that directly utilize the $[\widetilde{c_1}\, \widetilde{c_2} \ldots \widetilde{c_N}]$ bit strings are referred to as soft-decoding methods. On the other hand, hard-decoding techniques present the received bit string as the correct alphabet of coded data. Soft-decoding methods often come with higher costs but offer increased accuracy [1].

One notable feature that distinguishes various coding methods is their ability to detect and correct errors. By incorporating modulator and demodulator blocks into this model, the $[c_1\, c_2 \ldots c_N]$ bit string in the transmitter is sent as a single-bit or multi-bit. For instance, in BPSK modulation, the data is modulated and transmitted one by one, while in QPSK modulation, two bits are modulated and demodulated together [2].

As discussed in Section *II*, BER value is utilized to assess and compare the performance of different coding methods. In coding, where the $[c_1\, c_2 \ldots c_N]$ bit string is a linear combination of $[d_1\, d_2 \ldots d_k]$, its probabilistic graphical model can be represented as shown in Fig. 6.

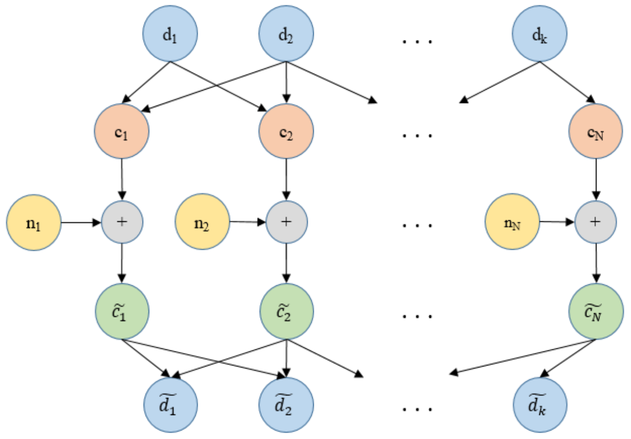

Fig. 6. Probabilistic graphical model of linear block coding

From the receiver's perspective, the challenge lies in extracting the $[\widetilde{d_1}\, \widetilde{d_2} \ldots \widetilde{d_k}]$ bit string from the received $[\widetilde{c_1}\, \widetilde{c_2} \ldots \widetilde{c_N}]$ bit string. If $\widetilde{d_i} = d_i$ for all i, then there are no errors in the receiver. However, if any $\widetilde{d_i}$ does not match its original value, an error has occurred. Without coding, it would be impossible to detect such errors if the $\widetilde{d_i}$s were to encounter errors while crossing the channel.

From the perspective of PRSs and based on Fig. 6, the problem entails establishing a probabilistic graphical model between d_i and $\tilde{c_i}$, while also determining the probabilistic model of the noise. The observed $\tilde{c_i}$s serve as the basis for calculating the probability of d_i occurrences. In other words, the problem can be expressed as finding $p(d_i|\underline{\tilde{\boldsymbol{c}}})$, where $\underline{\tilde{\boldsymbol{c}}}$ represents the observed $[\tilde{c_1}\ \tilde{c_2}\ \dots\ \tilde{c_N}]$. If $\boldsymbol{d}$ is a binary variable, the relationship $\tilde{d_i} = Round\left(p\left(d_i|\underline{\tilde{\boldsymbol{c}}}\right)\right)$ is established.

Based on Fig. 6, and utilizing Fig. 3, the probabilistic graphical model for the aforementioned example is constructed. This model encompasses the input bits of the encoder, its output, channel noise, and the received bits at the receiver. Fig. 7, displays the complete PGM model for this simple Hamming (7,4) code.

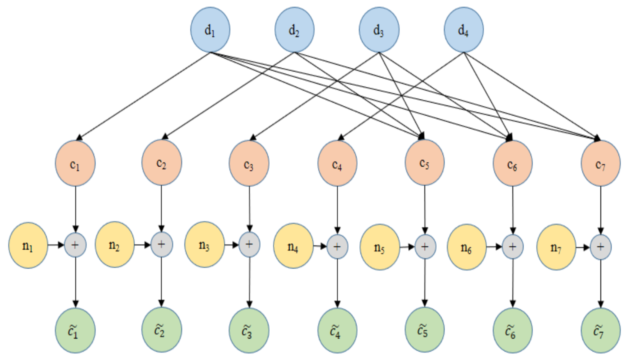

Fig. 7. PGM model of Hamming code in Fig. 3

If we want to explain the proposed methods based on Fig. 4 blocks (section *III*, part *B*, cases i to iv) and Fig. 7, it goes as follows:

i: The probabilistic model is equivalent to the relationship between the input bits (what the sender has transmitted) and the received bits. For example, in Fig. 7, $\tilde{c_5}$ at the receiver is a combination of the first three bits at the sender and then combined with noise ($\tilde{c_5} = d_1 + d_2 + d_3 + n_5$). The sample code we have written for this bit in the Figaro language is as follows:

```
val y1=Select(0.5->1,0.5->0)
...
val y4=Select(0.5->1,0.5->0)
val p1 = Apply(y1,y2,y3,(a:Int , b:Int , c:Int)=>(a+b+c)%2)
val p2 = Apply(y1,y3,y4,(a:Int , b:Int , c:Int)=>(a+b+c)%2)
val p3 = Apply(y1,y2,y4,(a:Int , b:Int , c:Int)=>(a+b+c)%2)
val Ebn0 = 0.125
val channel1 = Normal(0,Ebn0)
...
val channel7 = Normal(0,Ebn0)
val chr1 = Apply(y1,channel1,(a:Int , b:Double)=>a+b)
...
val chr7 = Apply(p3,channel7,(a:Int , b:Double)=>a+b)
```

In this code, the “Select function” chooses a value between 0 and 1 with equal probability. The “Apply” function is used to combine multiple variables. The “Normal” function represents the normal distribution, with its input equal to Ebn0 (i.e. N(0,Ebn0)). In this code, “y1” to “y4” correspond to d_1 to d_4 and “channeli” is equivalent to n_i and "chri" is equivalent to $\tilde{c_i}$.

This part of the code in Figaro represents the probabilistic model. For simplicity, we did not include the definition of the required libraries and some functions in this code. The complete codes are provided in the supplementary file of this paper.

ii: Our observations are equivalent to the bits received at the receiver, i.e., the bit string $\tilde{\mathbf{c}}$. In Figaro language, it is represented as follows:

```
def threshold(d: Double) = d > 0.5
val r1 = Apply(chr1,threshold)
...
val r7 = Apply(chr7,threshold)
r1.observe(true or false)
...
r7.observe(true or false)
```

In this part of the code, first, a function named “threshold” is defined, and this function is applied to the variable “chri”. The variables “ri” are defined to apply the above function to “chri”. The function “r1.observe” is used to determine what the observed bit was at the receiver. In this example, if the observed bit at the receiver is equal to 0, the “false” will be inserted, and if it is equal to 1, the “true” will be inserted. Please note that the program can be written in a different way, where instead of true/false, we can use the values 0 and 1. In that case, some definitions and functions such as “threshold” will change.

iii, iv: The query we perform involves determining the probability that the input bit has a specific value, such as 0 or 1, based on the probabilistic model we constructed in the first part and the observations we have. If the probability value is greater than 0.5, the input bit is considered equivalent to the queried bit, otherwise, its negation will be the answer. See the code snippet below. The code presented below illustrates three distinct approaches to inference. Depending on the specific application, any of these three methods can be utilized. Here, we have provided all three methods to offer a clearer insight into Figaro.

```
//Inference with Belief Propagation
val iteration_BP = 50
val algorithm1 = BeliefPropagation(iteration_BP, y1)
algorithm1.start()
println("x1 from BP: " + algorithm1.probability(y1, 1))
algorithm1.kill()
//Inference with Sampling Algorithm
val samples = 50
val salgorithm1 = Importance(samples, y1)
salgorithm1.start()
println("x1 from SA : " + salgorithm1.probability(y1, 1))
salgorithm1.kill()
//Inference with Variable Elimination
println("x1 from VE : " + VariableElimination.probability(y1, 1))
```

In this part, three different reasoning methods are observed. The first method is the Belief Propagation (BP) method, which is executed for 50 iterations. In this example, the question asked is the probability of the input bit being equal to 1. Another method is the Sampling Algorithm, where we solve the problem using 50 samples. The last method is Variable Elimination, which we briefly explained in previous sections.

To provide further explanation, we first formulated a probabilistic model of the problem. Then, based on the value of Ebn0, we added noise. Next, we assumed that we observed a bit stream at the receiver. Based on the observation, the question asked is the probability of the input bit number 1 (or any) being

1 (or 0). If this probability is greater than 0.5, the input bit is considered to be 1 (or 0). If it is less than 0.5, the input bit is considered to be 0 (or 1). If it is exactly 0.5, we can predefine a value for it.

As is common in simulating the calculation of BER, the program is executed for a large number of different inputs to compute its value.

V. Implementation and Results

In this section, the applicability and practicality of using PPL in decoding linear block codes are illustrated through simulations conducted with Figaro. Several linear block codes, including Hamming (7,4), Hamming (15,11), LDPC (14,8), Convolutional (16,5) (or Convolutional (2,1,3)), and Polar (16,8), are chosen to demonstrate how PPL can decode these codes with Hamming distances of 3, 3, 3, 6, and 4, respectively. As previously mentioned, it should be noted that Convolutional codes are not inherently block codes. However, they can be considered block codes when they operate on fixed-length blocks of input data.

In these simulations for the BSC channel, the Hamming (7,4) code is implemented using three inference methods: Variable Elimination (VE), Simulated Annealing (SA), and Belief Propagation (BP), as implemented in Figaro. For the BAWGN channel with BPSK modulation, all the mentioned codes are simulated using the same inference methods. The assumptions made during these simulations are as follows:

- The generator matrix $\boldsymbol{G}$ for Hamming (7,4) is considered as shown in Fig. 2. The $\boldsymbol{G}$ matrix for Hamming (15,11) is obtained from [30] , for LDPC (14,8) and Convolutional (16,5) from [1], and for Polar (16,8) from [31].
- No modulation is used in the BSC channel, while BPSK modulation is used in the BAWGN channel. The probability of error in BPSK modulation is calculated using (1).
- The bits' occurrence probability is assumed to be equal and set to 0.5 in binary mode (See section *IV*, "Select" function descriptions). Additionally, these bits are assumed to be independent of each other.
- The hard-decoding method is selected for implementation.

We have used IntelliJ IDEA to compile Figaro codes. We have included the sample codes that we have written in Figaro language as supplementary files attached to the paper. Mathematical calculations and the generation of BER diagrams are performed using Wolfram Mathematica software. As you know, the horizontal axis of the BER curve represents the SNR, signal energy, or probability of error, while the vertical axis indicates the bit error rate.

A. *Binary Symmetric Channel*

Fig. 8, displays the BER diagram for the probability of bit error ranging from 0.05 to 0.1 in the BSC channel for Hamming (7,4) code. The diagram compares four different modes: decoding using Variable Elimination (VE), decoding using Belief Propagation (BP) with 25 iterations, decoding not used, and decoding using Sampling Algorithm (SA) with 250 samples. Based on Fig. 8, it is evident that the use of Hamming decoding with VE and BP methods in PPL does not outperform the case where decoding is not utilized.

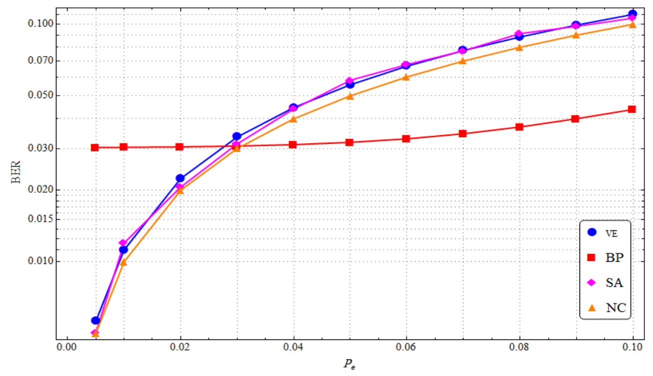

Fig. 8. BER diagram for comparison between the three inference methods in BSC channel for Hamming (7,4) decoding and uncoded data

The results obtained from the PPL methods align with the previous methods [32]. However, it is important to note that the BP method exhibits a significant advantage when the bit error exceeds 0.03. On the other hand, it is highly inefficient for values lower than 0.03.

Fig. 9, showcases the impact of changing the number of iterations on the BER value in the BP method, with a channel error probability of 0.04. Table I, which corresponds to Fig. 9, reveals that increasing the iteration count in the BP method from 5 to 30 leads to a decrease in the BER value from 0.2 to 0.014. This implies that higher iteration numbers in the BP method generally contribute to enhanced coding accuracy. Although Hamming (7,4) is one of the simplest codes, and there exist very simple and low-complexity methods for decoding it, we have used this simple code to demonstrate the impact of various inference parameters used in PRS models.

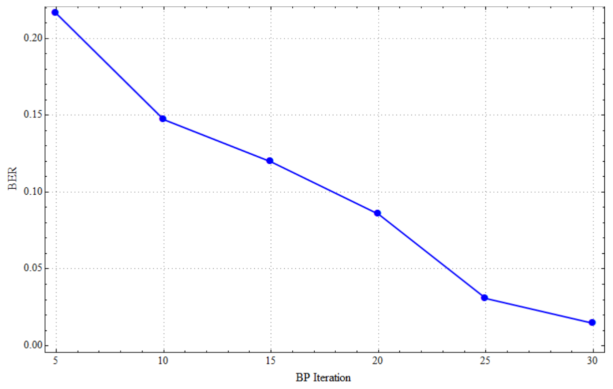

Fig. 9. The effect of increasing iteration in the BP algorithm for BSC channel for Hamming (7,4) decoding per probability of error 0.04

TABLE I
Data Related to Fig. 9

Iterations	BER
5	0.216231
10	0.14722
15	0.11968
20	0.08565
25	0.03076
30	0.01459

B. AWGN Channel

In order to demonstrate the performance of using PPL in decoding linear block codes, simulations are conducted in the AWGN channel with BPSK modulation. The SA inference algorithm is employed, and the BER is calculated for different sampling values and compared with the exact value obtained from (1). Fig. 10, provides a comparison between the exact value and the SA method for two sampling values, 2.5×10^6 and 10^5. It is evident that increasing the number of samples leads to improved accuracy. Furthermore, it demonstrates that PPL is capable of accurately calculating the BER for linear block coding with very good precision.

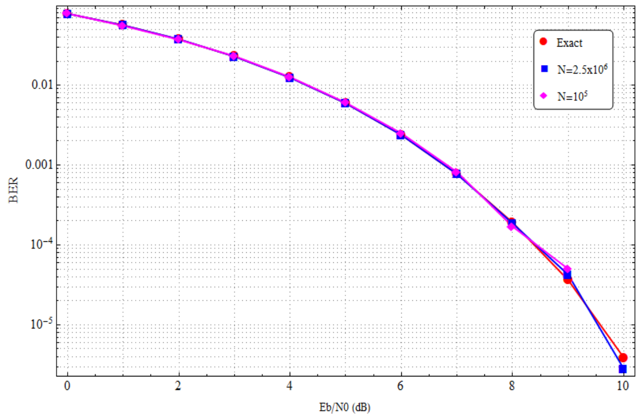

Fig. 10. Comparison of BER in AWGN channel with BPSK modulation by SA method with 2.5×10^6 samples (blue), 10^5 samples (magenta), and exact value (red) for uncoded data

Fig. 11, illustrates the BER diagram in the AWGN channel with BPSK modulation for various decoding schemes, along with a comparison with existing methods. The considered decoding schemes are uncoded data transmission, Hamming (7,4) with VE inference (with an Abstraction value of 500) and SA method (with 2000 samples), and Hamming (15,11) with VE inference. The results are compared with (7,4) for the VE method (with an Abstraction value of 500) and the SA method (with 2000 samples), compared with [33,34] and [30], respectively. The comparison demonstrates that the proposed method yields acceptable results, with the BER values in the proposed method and existing methods being close to each other.

Fig. 12, presents a comparison of different decoding schemes with their common decoding methods in the AWGN channel with BPSK modulation. The inference method employed in this case is VE in PPL. The considered decoding schemes are uncoded data transmission, LDPC (14,8), Convolutional (16,5), and Polar (16,8). The results are compared with [1], [1], and [31], respectively. The comparison shows that the proposed method provides comparable results, with the BER values in the proposed method and existing methods being close to each other (indicated by the dashed lines).

As evident from Fig. 10 to 12, our proposed method does not demonstrate superiority in terms of BER compared to the existing methods. We conducted these simulations to illustrate that while using PPL alongside its potential advantages, it can still provide similar results to the existing methods. This can allow us to confidently utilize this method.

Fig. 13, demonstrates the effect of the number of samples on the BER in the SA method. The figure shows that as the number of samples increases, the BER value decreases. These values are simulated for an Eb/N0 value of 4dB. According to the figure, as the number of samples increases from 10 to 1500, the BER value decreases from 0.022 to 0.013.

Finally, Fig. 14, displays a comparison involving the BP method with 10 and 30 iterations, as well as uncoded transmission for Hamming (7,4). The figure highlights that neither of these two modes shows an advantage over uncoded transmission, but increasing the number of iterations leads to improved results. While increasing the number of iterations beyond 50 brings the performance of this inference method closer to conventional methods, iterations in the range of 10 to 30 are typical in traditional methods [35,36]. Furthermore, this result indicates that in addition to selecting an appropriate inference method and determining its parameters, different decoding methods - such as the one we have proposed - can yield unacceptable responses compared to other methods.

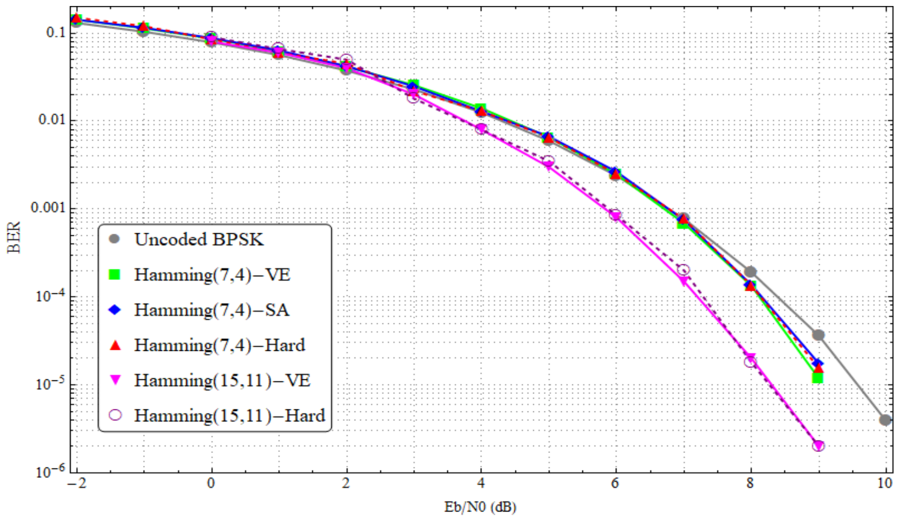

Fig. 11. Comparison of BER for some decoding methods with uncoded (BPSK modulation and AWGN channel). Dashed lines are placed for existing methods, also, "Hard" stands for existing

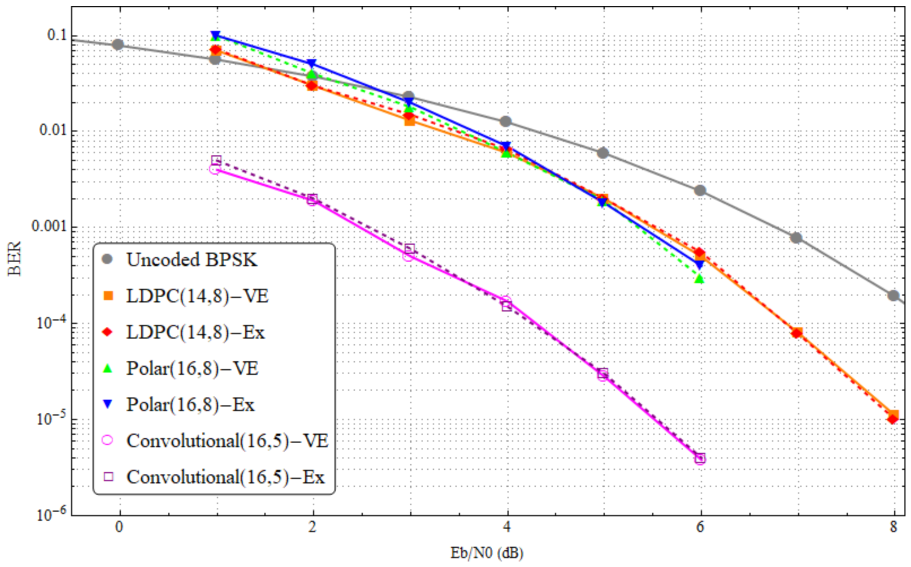

Fig. 12. The BER Comparison; Proposed method (thick lines) with existing (dashed lines); Decoding methods: LDPC(14,8), Polar(16,8), Convolutional(16,5); BPSK modulation and AWGN channel); Ex stands for Existing method.

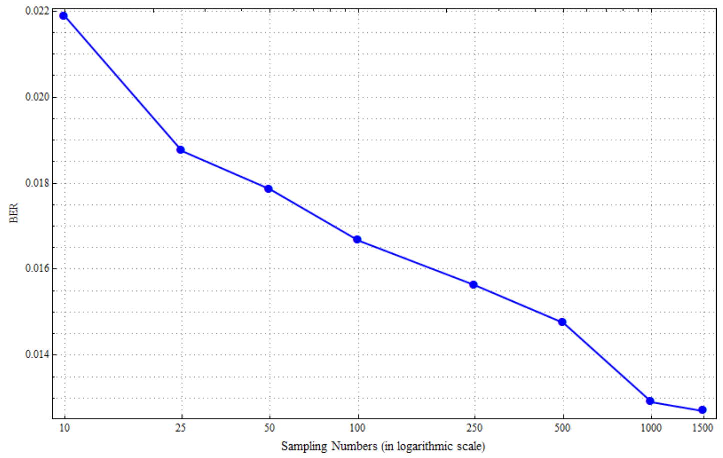

Fig. 13. The effect of the SA method sampling number in BER for Hamming (7,4) decoding (Eb/N$_0$ = 4dB)

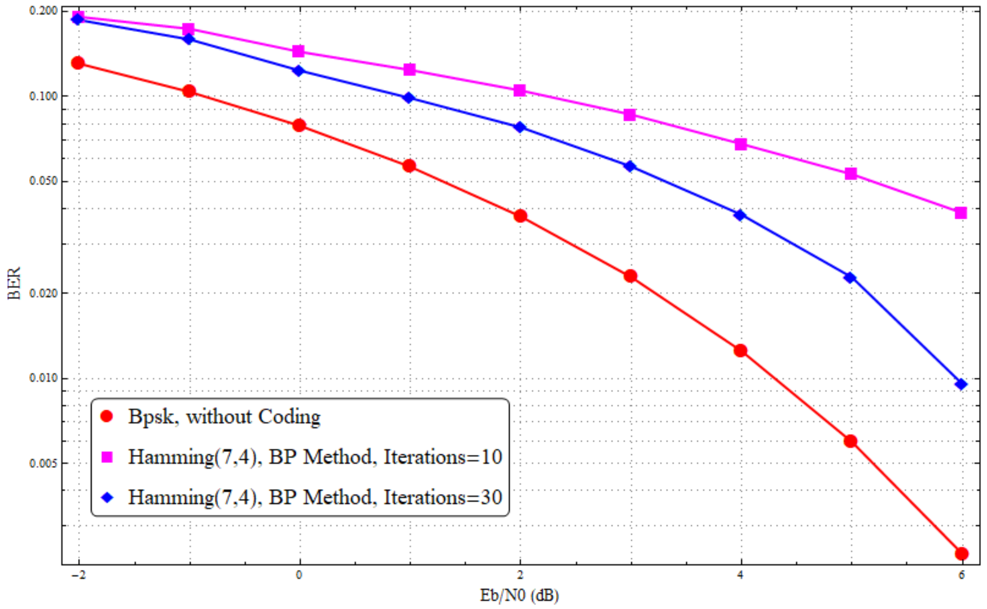

Fig. 14. The effect of the number of iterations of the BP method on the BER for Hamming (7,4)

VI. Advantages and Disadvantages

In the preceding section, we conducted an implementation for decoding linear block codes utilizing Figaro and showcased that employing PPLs can yield comparable outcomes to existing methods. In this section, we will delve into the advantages and disadvantages of employing PPL for decoding. By considering the benefits of this approach, we strongly believe that the utilization of PPL can be justified. Furthermore, with the ongoing advancement of these languages in the future, they have the potential to garner increased attention and recognition. Finally, we present both advantages and disadvantages in Table II.

A. Advantages

Some advantages of the proposed method are described below.

1) Utilization of different inference methods

PPL languages offer support for various inference methods, such as VE, BP, SA, and others. Each of these methods has distinct advantages in different applications. For instance, SA methods tend to be more effective in graph structures containing loops, while both VE and BP methods exhibit higher accuracy in loop-free graphs. Additionally, as discussed in the previous section, different inference methods perform differently under varying channel conditions. Comparatively, existing decoding methods typically employ one or two optimized and fixed methods at the receiver [9]. Therefore, incorporating PPL in the receiver enables the selection of decoding methods based on prevailing conditions. Further explanation is that since the proposed method is based on probabilistic programming, we can write a separate code for each method. For example, for a specific decoding technique, after defining the probabilistic model and variables, we can utilize various inference methods. Each method is written in a separate code, and the appropriate program will be selected based on different conditions.

2) The ability to choose between hard decoding or soft decoding methods based on channel conditions

Conventional methods often implement either soft-decoding or hard-decoding exclusively [10]. Soft-decoding methods offer greater accuracy and lower error probabilities than hard-decoding methods under similar channel conditions, albeit at the cost of increased computational complexity and latency. Using PPL can enable us to have the freedom to choose between these two methods. Since this approach is based on programming, we can write programs for both methods and, based on the channel conditions, use one of them. For example, when we have a higher SNR value, we can use the hard method, and for lower values, we can use the soft method.

3) Implement a wide range of coding methods

As demonstrated in Section *IV*, PPL can implement various decoding techniques, including linear block codes, that can be modeled as probabilistic graphical models. Traditional decoding methods typically focus on implementing one or two decoding methods at the receiver [9,11]. Consequently, communication between the transmitter and the receiver adheres to a specific predetermined standard, utilizing a limited set of coding methods. By incorporating PPL, a receiver can execute multiple decoding methods based on the prevailing conditions. This means that different transmitters can utilize different coding methods to send data, and the receiver can decode the received data by invoking the appropriate program without requiring any hardware modifications. In other words, using PPL allows us to have a wide range of linear block codes along with various inference methods at the receiver side. For each coding method, there exists a separate program, and when that coding method is employed, the corresponding program will be called and executed. For instance, weaker coding can be used when the channel is in good condition, while stronger codes can be employed when the channel conditions are unfavorable, potentially resulting in energy savings.

4) Ability to set inference algorithm parameters

In traditional designs, decoding methods typically have fixed parameters that are predetermined before implementation. However, in PPL methods, the parameters of the inference algorithm can be adjusted. For instance, in the BP method, parameters such as the number of iterations and execution time can be fine-tuned. Similarly, in the SA method, the number of samples can be adjusted. The adjustability of these parameters enhances the flexibility of the receiver, allowing it to adapt to different channel conditions. For example, in poorer channel conditions, the number of iterations or samples can be increased, while in better conditions, these parameters can be decreased.

5) Use machine learning methods to optimize parameter setting

In the previous paragraphs, we discussed the adjustability of PPL parameters such as the number of iterations, the number of samples, the choice between soft or hard decoding, coding type, inference algorithm type, and so on. We also mentioned the ability to set appropriate parameters based on channel conditions. In the receiver, channel information is continuously calculated. By employing machine learning methods and utilizing the information on various channel conditions, algorithm parameters can be optimized. This allows for the selection of the best parameters based on the prevailing conditions. Some research and activities in the field of decoding involve artificial intelligence and machine learning [17,37–41]. Based on the channel conditions and the relationship between parameters, a probabilistic graphical model can be created to determine optimal parameter settings. This generates a probabilistic graphical model problem that can be inferred using PPL. Additionally, methods such as clustering or classification can be employed to examine channel conditions and allocate suitable parameters. Since this approach is based on probabilistic programming, it is capable of executing machine learning-based methods alongside it.

6) A tool for estimating channel condition

Various methods exist for estimating channel state information. For example, in [42] a new multi-stage detector for robust signal and spectrum sensing in cognitive radio is introduced, and in [43], authors propose the modified Newton's (MN)-based Improved Animal Migration Optimization (IAMO) algorithm in MIMO-OFDM systems for channel estimation. In wireless telecommunications systems such as cooperative computation offloading in mobile edge computing systems, which often employ powerful servers without energy limitations [44], PPL can serve as an effective tool for estimating different channel conditions in various operational modes, regardless of its involvement in decoding. In this

TABLE II.
Comparison of Advantages and Disadvantages of the Proposed Method to the Other Existing

No	Parameter	Proposed Method	Other Existing Methods
1	Multi inference method	Yes	Up to 2 methods
2	The ability to choose between hard decoding or soft decoding methods based on channel conditions	Yes	Only soft or only hard method or hybrid method
3	Various decoding method implementation	Yes	Up to 2 methods
4	Setting inference algorithm parameters	Yes	Use only default values
5	Use alongside machine learning	Yes	Possible not yet used
6	Estimating channel condition	Yes	Only with preambles
7	Adjusting the initial value of probability of bits	Yes	Usually, equivalent prob.
8	Large generating matrix ($\boldsymbol{G}$)	Not yet able	Some methods are able
9	Online use	Poor	Optimized
10	Energy consumption	More than existing	Usually optimized

scenario, the receiver (Access Point or Base Station) receives a pilot signal at specified intervals, applies various decoding methods to it, and sends the results to the transmitter via the downlink. Subsequently, the transmitter can utilize these results to select the appropriate decoding method.

7) Adjust the initial probability value of bit or symbol occurrence

PPL provides the capability to determine the initial probability value for variables. In telecommunications, different symbols have varying probabilities of occurrence. Traditional decoding methods often assume equal probability for all symbols. However, as the received data in the receiver is constantly changing, more accurate probabilities of symbol occurrences can be obtained by employing statistical methods and storing relevant information. The accuracy of initial probabilities for variables directly impacts the effectiveness of inference algorithms. Consequently, the more accurate the initial probabilities of the variables, the more accurate the inference algorithms will be. For further explanation of setting initial values, please refer to the code provided in section IV. In that code, functions like "Select" can be adjusted with an initial value. In that code, the probability values of 0 and 1 are considered equal. In applications where these values are not equal, the use of PPL can be employed to compute and initialize each of the bits.

B. Disadvantages

The proposed method has some limitations, which are explained below:

1) Limitations in the development of probabilistic graphs

PPLs, such as Figaro, have limitations when it comes to developing probabilistic graphs. For instance, in Figaro, methods like Apply or Chain have a maximum limit of five inputs [14]. As a result, analyzing graphs with large parent-child relationships (i.e., a large $\boldsymbol{G}$ matrix) becomes challenging. To address this, nested loops must be used, which can slow down the compile speed.

2) Compiling speed

The compiling speed of programs in PPLs depends on the skills of the programmer and the optimized code structures. With proficient programming and optimized code, programs can be compiled at a high speed. However, in channel coding applications in telecommunication systems, we have observed that the speed of PPL programs is slower compared to common methods such as FPGA or ASIC [5,24]. This can limit their usability in certain online applications that require processing times within the range of 0.5 to 50 milliseconds. For example, in the simulations we conducted, the compiling speed for Hamming (15,11) decoding using the SA inference method with 2000 samples is approximately 3 milliseconds. This processing was performed using an Intel Core i5 CPU with a frequency of 2.5 GHz and 6 GB of RAM. This compiling speed is roughly equivalent to 4 Kbps.

3) Requirement for powerful servers in high-speed data transfer applications

In high-speed applications involving data transfer, PPL data decoding necessitates the use of powerful servers. However, due to the limitations of mobile phones, it is currently not feasible to employ this method on the downlink side, as mobile devices may lack the necessary computational capabilities to handle the demands of PPL-based decoding.

4) Higher energy consumption

Despite the mentioned benefits of PPLs, they consume more energy in the decoding of communication systems compared to existing methods. Consequently, their use may not be cost-effective in scenarios where there are energy constraints, such as in wireless sensor networks [45]. This can be attributed to the fact that current methods often involve the design of optimal hardware circuits, which are more energy-efficient in practical applications.

Please note that these disadvantages should be considered within the context of the specific implementation and application of PPLs. Ongoing research and development efforts aim to address these limitations and enhance the efficiency and practicality of PPLs in various communication fields.

C. The Summary of Advantages and Disadvantages Comparison to the Other Existing Methods

Table II shows the summary of the advantages and disadvantages of the proposed method compared with the other existing methods.

VII. Conclusion and Future Works

This paper explores the use of probabilistic programming languages (PPL) for channel data decoding in telecommunications, specifically for linear block coding. It describes how a linear block coding problem can be transformed into a probabilistic graphical model, and how PPL can be utilized for decoding, as illustrated through simulations using the Figaro programming language. Simulations have shown that with skilled programming, the proposed method can achieve results similar to existing decoding methods. The proposed PPL-based decoding method is shown to be able to dynamically adjust parameters based on channel conditions, offer flexible decoding approaches, and leverage machine learning techniques, making it well-suited for scenarios without energy or real-time constraints. The paper also suggests future research to further enhance PPL-based probabilistic graphical models and apply them to wireless channel estimation.

References

[1] Lin S, Costello DJ (2004) Error Control Coding: Fundamentals and Applications, Pearson-Prentice Hall.

[2] Proakis J, Salehi M (2007) Digital Communications, 5th Edition, McGraw-Hill Science/Engineering/Math.

[3] MacKay DJC (2003) Information theory, inference and learning algorithms, Cambridge university press.

[4] Ferraz O, Subramaniyan S, Chinthala R, Andrade J, Cavallaro JR, Nandy SK, Silva V, Zhang X, Purnaprajna M, Falcao G (2022) A Survey on High-Throughput Non-Binary LDPC Decoders: ASIC, FPGA, and GPU Architectures. IEEE Commun Surv Tutorials 24, pp. 524–556.

[5] Hasan FS, Mosleh MF, Abdulhameed AH (2021) FPGA implementation of LDPC soft-decision decoders based DCSK for spread spectrum applications. Int J Electr Comput Eng 11, pp. 4794–4809.

[6] Kumar N, Kedia D, Purohit G (2023) A review of channel coding schemes in the 5G standard. Telecommun Syst 83, pp. 423–448.

[7] Ali MM, Hashim SJ, Chaudhary MA, Ferré G, Rokhani FZ, Ahmad Z (2023) A Reviewing Approach to Analyze the Advancements of Error Detection and Correction Codes in Channel Coding With Emphasis on LPWAN and IoT Systems. IEEE Access 11, pp. 127077–127097.

[8] Sunny J, Ameenudeen PE, Kumar RH (2022) Design of Machine learning based Decoding Algorithms for Codes on Graph. In Proceedings - 2022 IEEE Silchar Subsection Conference, SILCON 2022 IEEE, pp. pp. pp. 1–7.

[9] Luo FL, Zhang CJ (2017) Signal processing for 5G: Algorithms and implementations, John Wiley & Sons, Ltd, Chichester, UK.

[10] Wang J, Tang C, Huang H, Wang H, Li J (2021) Blind identification of convolutional codes based on deep learning. Digit Signal Process A Rev J 115, pp. 103086.

[11] Li S, Zhou J, Huang Z, Hu X (2021) Recognition of error correcting codes based on CNN with block mechanism and embedding. Digit Signal Process A Rev J 111, pp. 102986.

[12] Guo J, Cao C, Shi D, Chen J, Zhang S, Huo X, Kong D, Li J, Tian Y, Guo M (2021) Matching Pursuit Algorithm for Decoding of Binary LDPC Codes. Wirel Commun Mob Comput 2021, pp. 1–5.

[13] Singels R (2016) The application of Probabilistic Graphical Models to Raptor codes over Binary Input Memoryless Symmetric Channel models.

[14] Ścibior A, Ghahramani Z, Gordon AD (2016) Practical probabilistic programming with monads, Manning Publications Co.

[15] Obeid AK, Bruza PD, Wittek P (2019) Evaluating probabilistic programming languages for simulating quantum correlations. PLoS One 14, pp. e0208555.

[16] Yadav A, Kakde S, Khobragade A, Bhoyar D, Kamble S (2018) LDPC Decoder ' s Error Performance over AWGN Channel using. Int J pure Appl Math 118, pp. 3875–3879.

[17] Ly A, Yao YD (2021) A review of deep learning in 5G research: Channel coding, massive MIMO, multiple access, resource allocation, and network security. IEEE Open J Commun Soc 2, pp. 396–408.

[18] Carlson AB, Crilly PB, Rutledge JC (2002) Communication systems : an introduction to signals and noise in electrical communication, McGraw-Hill.

[19] Kumar RD, Vishvaksenan KS (2020) Interference cancellation in cognitive radio-based MC-CDMA system using pre-coding technique. J Supercomput 76, pp. 1–15.

[20] Goldsmith A (2005) Wireless communications, Cambridge University Press.

[21] Tanenbaum AS, Wetherall DJ (2013) Computer Networks, Pearson.

[22] Fanari L, Iradier E, Bilbao I, Cabrera R, Montalban J, Angueira P, Seijo O, Val I (2022) A Survey on FEC Techniques for Industrial Wireless Communications. IEEE Open J Ind Electron Soc 3, pp. 674–699.

[23] Tomlinson M, Tjhai CJ, Ambroze MA, Ahmed M, Jibril M (2017) Reed–Solomon Codes and Binary Transmission. In, Tomlinson M, Tjhai CJ, Ambroze MA, Ahmed M, Jibril M, eds. Springer International Publishing, Cham, pp. pp. pp. 167–179.

[24] Fanari L, Iradier E, Bilbao I, Cabrera R, Montalban J, Angueira P (2021) Comparison between different channel coding techniques for ieee 802.11be within factory automation scenarios. Sensors 21, pp. 7209.

[25] Krapu C, Borsuk M (2019) Probabilistic programming: A review for environmental modellers. Environ Model Softw 114, pp. 40–48.

[26] Le Bras R, Arora N, Kushida N, Mialle P, Bondár I, Tomuta E, Alamneh FK, Feitio P, Villarroel M, Vera B, Sudakov A, Laban S, Nippress S, Bowers D, Russell S, Taylor T (2021) NET-VISA from Cradle to Adulthood. A Machine-Learning Tool for Seismo-Acoustic Automatic Association. Pure Appl Geophys 178, pp. 2437–2458.

[27] Lake BM, Salakhutdinov R, Tenenbaum JB (2015) Human-level concept learning through probabilistic program induction.

[28] Ruttenberg B, Kellogg L, Pfeffer A (2016) Probabilistic Programming for Malware Analysis. CoRR abs/1603.0,.

[29] Farnoudkia H, Purutçuoglu V (2019) Copula Gaussian graphical modeling of biological networks and Bayesian inference of model parameters. Sci Iran 26, pp. 2495–2505.

[30] Alabady SA, Mohd Salleh MF, Al-Turjman F (2018) LCPC error correction code for IoT applications. Sustain Cities Soc 42, pp. 663–673.

[31] Koike-Akino T, Wang Y (2021) Protograph-Based Design for QC Polar Codes. In IEEE International Symposium on Information Theory - Proceedings IEEE, pp. pp. pp. 593–598.

[32] Sadlier RJ, Humble TS (2016) Superdense Coding Interleaved with Forward Error Correction. Quantum Meas Quantum Metrol 3,.

[33] Sarnin SS, Kadri N, Mozi AM, Wahab NA, Naim NF (2010) Performance analysis of BPSK and QPSK using error correcting Code through AWGN. ICNIT 2010 - 2010 Int Conf Netw Inf Technol pp. 178–182.

[34] Giordano AA, Levesque AH (2015) BER performance of BPSK in AWGN with a Hamming code. In Modeling of digital communications systems using Simulink Wiley, pp. pp. pp. 175–180.

[35] Karimi-Lenji A, Houshmand M, Zarmehi F (2017) A high-performance belief propagation decoding algorithm for codes with short cycles. Int J Commun Syst 30, pp. 1–8.

[36] Old J, Rispler M (2023) Generalized Belief Propagation Algorithms for Decoding of Surface Codes. Quantum 7, pp. 1037.

[37] Habib S, Mitchell DGM (2023) Reinforcement Learning for Sequential Decoding of Generalized LDPC Codes. In 2023 12th International Symposium on Topics in Coding (ISTC) IEEE, pp. pp. pp. 1–5.

[38] Fang M (2023) An Improved Min-Sum Polar Code Decoding Algorithm. In 2023 3rd Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS) IEEE, pp. pp. pp. 655–658.

[39] Huang L, Zhang H, Li R, Ge Y, Wang J (2020) AI Coding: Learning to Construct Error Correction Codes. IEEE Trans Commun 68, pp. 26–39.

[40] Qin Z, Fei Z, Huang J, Wang Y, Xiao M, Yuan J (2023) Reinforcement-Learning-Based Overhead Reduction for Online Fountain Codes With Limited Feedback. IEEE Trans Commun 71, pp. 3977–3991.

[41] Song D, Ren J, Wang L, Chen G (2022) Designing a Common DP-LDPC Code Pair for Variable On-Body Channels. IEEE Trans Wirel Commun 21, pp. 9596–9609.

[42] Jeevangi S, Jawaligi S, Patil V (2022) Deep Learning-based SNR Estimation for Multistage Spectrum Sensing in Cognitive Radio Networks. J Telecommun Inf Technol.
[43] Venkateswarlu C, Rao NV (2022) Optimal channel estimation and interference cancellation in MIMO-OFDM system using MN-based improved AMO model. J Supercomput 78, pp. 3402–3424.
[44] Khazali A, Bozorgchenani A, Tarchi D, Shayesteh MG, Kalbkhani H (2023) Joint Task Assignment, Power Allocation and Node Grouping for Cooperative Computing in NOMA-mmWave Mobile Edge Computing. IEEE Access 11, pp. 93664–93678.
[45] Naik C (2022) Computational Intelligence Algorithms For Energy Optimization Problems In Wireless Sensor Networks.