LDPC coded compressive sensing for joint source-channel coding in wireless sensor networks

Jue Chen, Shuai Shao, *Student Member, IEEE*, Tsang-Yi Wang, *Member, IEEE*, Jwo-Yuh Wu, *Member, IEEE*, Chih-Peng Li, *Fellow, IEEE*, Soon Xin Ng, *Senior Member, IEEE*, Robert G. Maunder, *Senior Member, IEEE* and Lajos Hanzo, *Life Fellow, IEEE*

***Abstract*—The novel concept of joint Compressive Sensing (CS) and Low Density Parity Check (LDPC) coding is conceived for Joint Source-Channel Coding (JSCC) in Wireless Sensor Networks (WSNs) supporting a massive number of signals. More explicitly, we demonstrate this concept for a specific scheme, which supports a massive number of signals simultaneously, using a small number of Internet of Things Nodes (IoTNs) based on the concept of CS. The compressed signals are LDPC coded in order to protect them from poor transmission channels. We also propose the new iterative joint source-channel decoding philosophy for exchanging soft *extrinsic* information, which combines CS decoding and LDPC decoding by merging their respective factor graphs. We then characterize this scheme using Extrinsic Information Transfer (EXIT) chart analysis. Our BLock Error Rate (BLER) results show that the proposed iterative joint LDPC-CS decoding scheme attains about 1.5 dB gain at a BLER of 10^{-3} compared to a benchmarker, which employs separate CS and LDPC decoding. Naturally, this gain is achieved at the cost of approximately doubling the complexity of the proposed iterative joint LDPC-CS decoding scheme.**

***Index Terms*—Joint source-channel coding, compressive sensing, LDPC codes, factor graphs, wireless sensor networks, *extrinsic* information transfer charts**

I. Introduction

The Compressive Sensing (CS) technique has been applied in various fields since it was first proposed [1], with Wireless Sensor Networks (WSNs) being an important application. In particular, the Internet of Things Nodes (IoTNs) in WSNs always have constrained energy resources [2], [3], and the challenge of energy-efficient data gathering in WSNs can be mitigated by applying CS [4]–[6]. Here, a vector of signals may be observed by a smaller number of IoTNs according to a sensing matrix, which describes the connectivity among the signals and IoTNs. The observation vector may then be used for reconstructing the signals, provided that the sensing matrix satisfies the Restricted Isometry Property of the l_1-norm (RIP1) [7]. However, in practice, WSNs suffer not only from constrained energy resources but also their unreliable communication channels, which motivates intrinsically amalgamating CS with Joint Source-Channel Coding (JSCC) for their conceived optimization.

Briefly, JSCC combines source coding and channel coding together, where source coding compresses the signal by removing any predictable redundancy, and then the channel coding protects the signal from errors by adding redundancy in a carefully controlled manner [8]. Furthermore, while Shannon's source-channel coding separation theory suggests there is no penalty for using separate source and channel coding, this only holds when unlimited delay and complexity can be afforded, and in the case of stationary channels and sources [9]. If any these assumptions do not apply, then JSCC can offer a benefit by directly transforming the source signal to a coded signal having 'just' the right appropriate amount of redundancy for struggling the required trade-off between compression and error correction. In particular, JSCC has been shown to offer significant coding gain when applied to finite block length transmission and non-stationary channels [10]. Furthermore, JSCC also offers throughput, energy-efficiency and end-to-end latency advantages, while makes it suitable for many applications such as speech, image, and video transmission [11], [12].

In recent years, JSCC has been applied in diverse scenarios. Double LDPC codes were introduced for JSCC in [13], where the first LDPC code was utilized for compressing the signal to complete the source coding, while the second LDPC code was used for protecting the compressed bits as channel coding. Based on this concept, the authors of [14] applied Double-Protograph LDPC (DP-LDPC) codes for JSCC, taking into account the superiority of protograph LDPC codes for efficient encoding and decoding [15]. JSCC has also been employed in numerous WSNs [16]–[18], but, in the conventional approach to signal sensing, the number of IoTNs is typically equal to

L. Hanzo would like to acknowledge the financial support of the Engineering and Physical Sciences Research Council projects EP/P034284/1 and EP/P003990/1 (COALESCE) as well as of the European Research Council's Advanced Fellow Grant QuantCom (Grant No. 789028). Wang's work is supported by MOST of Taiwan under grant 110-2221-E-110-021. J.-Y. Wu's work is supported in part by the Ministry of Science and Technology of Taiwan under grants MOST 108-2221-E-009-025 MY3 and MOST 110-2634-F-009-025, in part by the Higher Education Sprout Project of the National Yang Ming Chiao Tung University and Ministry of Education of Taiwan, and in part by the MOST Joint Research Center for AI Technology and All Vista Healthcare. Prof. Chih-Peng Li would like to acknowledge the financial support from the Ministry of Science and Technology, Taiwan, under grant numbers MOST 108-2218-E-110-014 and MOST 109-2218-E-110-006.

Corresponding Author: L. Hanzo, J. Chen, S. Shao and S.-X. Ng and R.-G. Maunder are with School of Electronics and Computer Science, University of Southampton, SO17 1BJ, UK. (E-mail: lh@ecs.soton.ac.uk; jc7m17@soton.ac.uk; ssecs1994@gmail.com; sxn@ecs.soton.ac.uk; rm@ecs.soton.ac.uk).

T.-Y. Wang is with the Institute of Communications Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan (E-mail: tcwang@mail.nsysu.edu.tw).

J.-Y. Wu is with the Institute of Communications Engineering, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan (E-mail: jywu@cc.nctu.edu.tw).

C.-P. Li is with the Institute of Communications Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, and also with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan (E-mail: cpli@faculty.nsysu.edu.tw).

or higher than the number of signals, implying a high cost of deploying and maintaining the IoTNs. This is in contrast to CS, where the number of IoTNs is typically lower than the number of signals to be monitored, but larger than the number of non-zero elements in a sparse measurement vector. In this way, CS exploits the knowledge that only a few of the signals will be activated any time, hence allowing a smaller number of IoTNs to be adopted. Inspired by the advantages of DP-LDPC codes for JSCC and the particular properties of WSNs that suffer from constrained energy resources and unreliable communication channels, in this paper, we conceive a CS-LDPC scheme that allows a small number of IoTNs to observe a large number of sparse time-varying signals. This is achieved with the aid of factor graphs, by harnessing the concept of CS and exploiting the observations that the sparse signals have a high probability of adopting a value of zero. More specifically, by adopting the concept of CS, fewer IoTNs are required, thus reducing the deployment cost, effort and maintenance. Therefore, the implementation of CS in WSNs has received particular attention [19], [20]. As a complement to this, LDPC codes are employed for protecting the sensing information during transmission over an unreliable wireless channel. Although diverse channel codes have been shown to be beneficial in JSCC [21], [22], LDPC codes are selected here because they are well-known to have strong error-correction performance. Furthermore, both LDPC codes and CS can be represented using factor graphs. Hence, we exploit this factor graph representation to design a novel iterative joint source-channel decoder, which iteratively exchanges soft *extrinsic* information between the CS decoder and LDPC decoder. Furthermore, it may be argued that LDPC codes have become ubiquitous state-of-the-art for channel coding schemes, since they have been adopted both in WiFi [23] and in 3GPP 5G new radio standards [24]. The proposed scheme is capable of simultaneously detecting multiple sparse signals with the aid of a Fusion Center (FC), which represents a data processing center that gathers all data from the distributed IoTNs via wireless communication links and performs iterative decoder processing in order to recover the information. Meanwhile, we introduce an iterative joint LDPC-CS decoding algorithm for the proposed scheme, which is shown to improve the performance compared to separate source and channel decoding.

Iterative decoding techniques have been shown to be beneficial in JSCC schemes [11], [13]. More specifically, the authors of [11], [13] proposed a serially concatenated LDPC-Low Density Generator Matrix (LDGM) code for JSCC. In contrast to these prior works, the proposed scheme has a number of important differences. Firstly, the CS decoder is significantly different from the LDGM decoder of [11], [13] and introduces unique challenges for which we have proposed novel solutions. More specifically, the CS decoder relies on the concept of using a small number of IoTNs to detect a sparse signal among a high number of zero-valued signals. This sparsity introduces a particular probability distribution to the bits that are processed by the CS and LDPC encoders. This regime is in contrast to the LDGM encoder, where all the bits have equiprobable 0 and 1 values. This non-equiprobable probability distribution of the different CS encoders in the proposed scheme creates a particular challenge during the iterative decoding, because the iteratively exchanged LLRs have a non-equiprobable distribution. Furthermore, the IoTNs operate on the basis of the non-linear OR function, in contrast to the XOR function of the LDGM encoder, which presents unique challenges in terms of the operations of the IoTNs. Moreover, the proposed scheme has adopted multiple LDPC decoders for decoding the observations gleaned from the different IoTNs simultaneously, as well as multiple CS decoders to decode information from different Time Slots (TSs) simultaneously. In this way, we enable information provided by different IoTNs and different TSs to be iteratively exchanged, so that they assist each other's recovery. Again, this is in contrast to the LDPC-LDGM scheme of [11], [13], where there is only a single LDPC decoder as well as one LDGM decoder, and there is no information exchange between the different time slots or different LDPC decoders.

TABLE I: Boldly contrasting our contributions to the state-of-the-art

	[11]	[13]	[14]	[17]	[18]	[38]	Proposed
JSCC	√	√	√	√	√	√	√
CS							√
LDPC codes	√	√	√	√		√	√
Factor graphs	√	√	√	√			√
EXIT charts for non-equiprobable bits							√
WSNs				√	√	√	√

To elaborate further, iterative decoding is achieved based on the exchange of *extrinsic* soft information between two or more constituent decoders. Based on this feature, we use Extrinsic Information Transfer (EXIT) charts to analyze our proposed scheme and visualize the iterative exchange of *extrinsic* information between the two decoders [25], [26]. More specifically, EXIT charts are powerful tools of analyzing the convergence behavior of iterative decoding schemes by using a decoding trajectory for intuitively illustrating the convergence process [26]. This allows the verification, characterization and parameterization of iterative decoding schemes at a much lower simulation complexity than that of bit-by-bit full decoding simulations [27]. We boldly and explicitly contrast our contributions to the state-of-the-art in Table I and detail them below:

1) We conceive a new JSCC scheme for the transmission of multiple sparse time-varying signals using a small number of SNs and an intelligent FC. This novel scheme facilitates the accurate sensing of the sparse signals using a low number of IoTNs and energy-efficient communications.
2) Considering the particular structure of the proposed JSCC scheme, we design a new iterative joint source-channel decoding process, which iterates between LDPC decoding and CS decoding using a common factor graph in the FC. Here, LDPC decoding is employed for channel decoding, while CS decoding is employed for source decoding. We demonstrate that the iterations exchanging soft *extrinsic* information between the LDPC decoder and the CS decoder enhances the performance of the scheme by about 1.5 dB at a BLock Error Rate (BLER) of 10^{-3} at the cost of doubling the complexity compared to separate source-channel decoding. In order to facilitate the flexible handling of different signals having different

block lengths for transmission our different channels having different capacities, we adopt the specific quasi-cyclic protograph LDPC code, which has been adopted by the 3GPP 5G new radio [24].

3) In order to confirm and characterize the iterative decoding convergence of the proposed scheme, we propose a novel technique for applying EXIT charts [28], our solution is different from conventional EXIT charts, since we additionally take into account the sparsity of the bits. The associated information is iteratively exchanged, which leads to these bits having an entropy less than 1.

The rest of this paper is organized as follows. Our novel JSCC scheme is proposed in Section II, while its iterative joint source-channel decoder is detailed in Section III. Then, the complexity of the decoding process is characterized in Section IV, whilst the EXIT chart analysis of the novel scheme is described in Section V. Section VI provides simulation results for the proposed scheme and our analysis demonstrates a gain of about 1.5 dB over the benchmarker. Finally, we offer our conclusions in Section VII.

II. System model

This section describes our novel scheme, which performs CS and LDPC coding. The assumptions of this work are listed below:

Assumption 1: In the proposed scheme, it is assumed that each IoTN has knowledge of which signals it is connected to, but not the global knowledge of the complete sensing matrix $\mathbf{\Phi}$. In other words, the IoTNs do not have the knowledge of each other's connectivity. However, the FC is aware of the complete sensing matrix $\mathbf{\Phi}$, i.e., it has the global knowledge of which IoTNs are observing which signals. Meanwhile, we assume that the connections between IoTNs and signals remain unchanged across the series of T TSs. In practice, the knowledge of the connections between the signals and IoTNs could be obtained during the system construction phase in the scenarios such as cooperative spectrum sensing and source localization [29].

Assumption 2: Similar to [29], we assume that exactly K signal elements are non-zero during each TS, but we consider two scenarios concerning the knowledge of the FC pertaining to sparsity K. In the first scenario, the true sparsity level K is known *a priori* to the FC and hence the upper bound of sparsity $K' = K$. In the second scenario, the true sparsity level K is unknown to the FC, but the upper bound of sparsity K' ($1 \leq K \leq K'$) is known in the FC. In a practical CS-based WSN, the sparsity level or the upper bound of the sparsity level may be estimated using residual-based algorithms or cross-validation, which is usually implemented in the FC during the WSNs' training phase [30].

Assumption 3: We assume the presence of a control overhead or a side information channel that establishes connectivity, maintains synchronization, and is used for signal acquisition. For the sake of simplicity, this control overhead or side information channel is not simulated in this work. However, we note that [31] discusses these overheads for controlling synchronization and signal acquisition in FC based WSNs.

Fig. 1 provides a block level diagram of the JSCC scheme. The steps in the first dashed box are completed by the IoTNs, while the steps enclosed in the second dashed box are completed by the FC. Each part of the scheme is described as follows.

- **Sparse signals in our scheme:** A factor graph is shown at the left-hand side of Fig. 1 for characterizing the connectivity between the N signals ($V_1 \dots V_N$) and M IoTNs ($S_1 \dots S_M$). Here, we have fewer IoTNs than signals, $M < N$, which is facilitated by exploiting the sparsity of the time-varying binary sparse signals $\boldsymbol{x}_1$ to $\boldsymbol{x}_N$. In this work, these signals are assumed to be IoTN measurements of physical quantities, such as audio signals, temperature, etc. To elaborate further, the n^{th} ($n \in [1, N]$) Variable Node (VN) V_n can be considered to be a random variable that adopts a particular value at each TS of the n^{th} signal $\boldsymbol{x}_n = [x_{n,1}, \dots, x_{n,t}, \dots, x_{n,T}]$, which is a vector comprising T non-equiprobable bits. In the proposed scheme, each element of each signal vector adopts different values at each of the T TSs, and each signal has the same length T. The value of the n^{th} signal in the t^{th} ($t \in [1, T]$) TS is represented as $x_{n,t} \in \{0, 1\}$. Furthermore, in each TS, we have exactly K of the N signals adopting non-zero value, and $N - K$ signals having a value of zero. In cases where K is significantly smaller than N, we have a sparse signal, which lends itself to compressive sensing. Consider a real-life application such as a tiger habitat with a known number of tigers K, which is divided into N blocks of one-square-meter tiles, where a total of M microphones are installed throughout the habitat. Each microphone is configured to detect tiger activity in a particular subset of neighbouring tiles. In this case, we know that exactly K of these one-square-meter tiles will be occupied at any one time. When the microphones detect the noise made by the tigers in these occupied tiles, they activate the sensors and their transmitters, hence allowing us the track of tigers in a non-invasive way.

- **IoTNs in our scheme:** The M IoTNs compress the N ($N > M$) signals to obtain the M observations. Here, the IoTNs represent sensor nodes, which include microphones, motion sensors, or temperature sensors, for example. Each of the IoTNs S_m ($m \in [1, M]$) observes different number of signals and different combinations of signals. The connectivity between the signals and IoTNs are shown as dotted lines in the factor graph at the left-hand side of the Fig. 1. In the proposed scheme, all of the signals have the same degree K_s, which means that each signal is observed by K_s IoTNs. However, The number of signals observed by a IoTN is referred to as its degree, where the set of degrees is represented by the vector $\boldsymbol{d} = [d_1, \dots, d_m, \dots, d_M]^T$ and $d_m \in \mathbb{Z}^{0+}$. Note that the IoTNs may have different degrees to each other, but under the constraint that $\sum_{m=1}^{M} d_m = NK_s$, where N is the number of signals and K_s is the fixed degree adopted for each signal. Meanwhile, the connectivity between

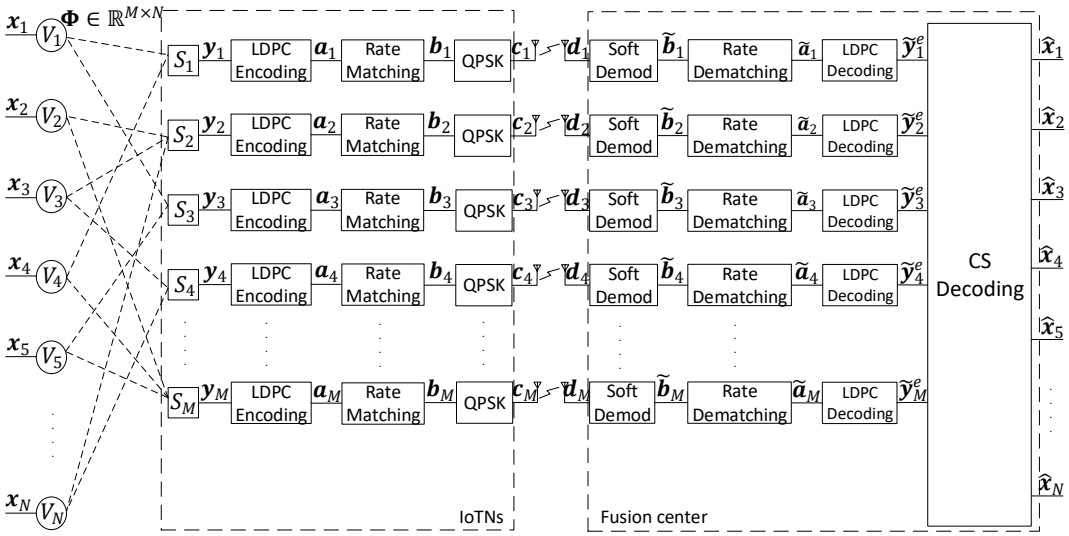

Fig. 1: System model representing the relationship between signals, IoTNs and Fusion Center.

the N signals and M IoTNs may be represented by a sensing matrix $\mathbf{\Phi} \in \{0,1\}^{M\times N}$ having M rows and N columns. Here, each element in this matrix represents a randomly selected edge in a factor graph representation of the connectivity between the signals and IoTNs. More specifically, each row of $\mathbf{\Phi}$ represents the connectivity of a corresponding IoTN, while each column represents the connectivity of a corresponding signal. For instance, in Fig. 1, the first IoTN senses the first signal, giving $\phi_{1,1} = 1$. By contrast, the first IoTN does not sense the second signal, and therefore $\phi_{1,2} = 0$. The use of a sparse sensing matrix to represent the connectivity of the signals and IoNTs has also been considered in [29] and has found application in [32]. In a realistic example, we may have M microphones, each of which observes a different subset of the N audio sources, where the matrix Φ may be used to represent the connectivity between the microphones and sources.

- **Functions performed in the IoTNs:** The output of each IoTN is a binary observation vector $\boldsymbol{y}_m$ having the same length T as the signals, where m is in the range of $1, 2, \ldots, M$. The value of the m^{th} IoTN in the t^{th} TS is represented as

$$y_{m,t} = \mathcal{X}_{m,t}(1) \vee \mathcal{X}_{m,t}(2) \ldots \vee \mathcal{X}_{m,t}(d_m), \qquad (1)$$

where $\boldsymbol{\mathcal{X}}_{m,t}$ represents the set, which includes all signals observed by the m^{th} IoTN in the t^{th} TS. Furthermore, $\vee$ denotes the OR function performed in the IoTNs. An element $y_{m,t}$ of the observation vector is set to 1 only when one or more of the observed signals by that particular IoTN adopt a value of 1 in that TS, according to an OR function. If the observed signals are all zeros, then the output of the connected IoTN will be 0 in that TS. Returning to our practical example of tracing wild animals in their natural habitat, the use of the OR function in the generation of $y_{m,t}$ is reflected by the use of a microphone to detect noise from multiple sources, where the microphone is activated whenever any of the sources is activated, corresponding to an OR function.

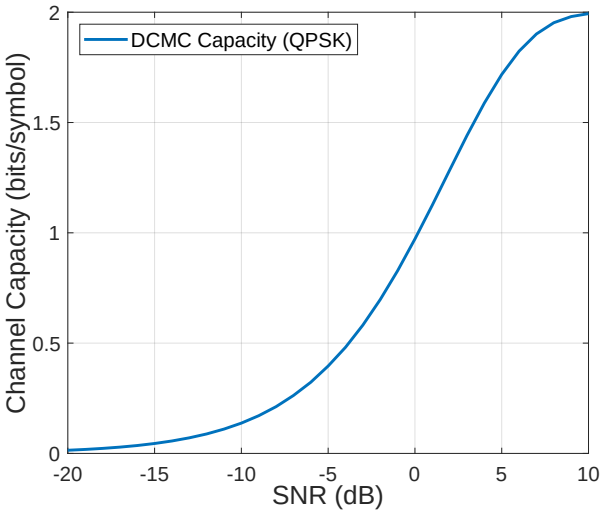

Fig. 2: The channel capacity for the Additive White Gaussian Noise (AWGN) Discrete-input Continuous-output Memoryless Channel (DCMC) using QPSK modulation.

After observing the connected signals, the m^{th} IoTN will individually encode its own observation vector $\boldsymbol{y}_m$ using an LDPC code having a block length of T. Here, we employ the specific LDPC codes and rate matching [1] scheme that was introduced in 3GPP 5G new radio [24], which adopts particular degree distributions for its VNs and CNs. The resultant intermediate bit sequence $\boldsymbol{a}_m$ ($m \in [1, M]$) has a length of U bits, and it is obtained

[1]Rate matching adjusts the coding rate of the LDPC encoder, so that a specified number of LDPC encoded bits are generated, which may be different from the number of bits provided by the native LDPC encoder. Explicitly, repetition may be used when an increased number of encoded bits is desired, while puncturing may be used to reduce a number of encoded bits [33], in this way, the desired coding rate may be selected for closely matching the channel capacity. As a benefit, the channel capacity is never wasted or overloaded.

by encoding the T bits binary observation vector T bits binary observation vector $\boldsymbol{y}_m$ using the parity check matrix $\mathbf{H} \in \{0,1\}^{W \times U}$ of the LDPC code[2]. Following this, rate matching with coding rate $R = \frac{T}{E}$ is applied in order to obtain the binary bit vector $\boldsymbol{b}_m$ having a length of E. Rate matching is followed by Quadrature Phase Shift Keying (QPSK) modulation in order to obtain the In-phase and Quadrature (IQ) vector $\boldsymbol{c}_m$, which is a complex vector having a length $\frac{E}{2}$ and the symbol energy E_0 is normalized to 1.

- **Transmission over a channel:** The IQ vector $\boldsymbol{c}_m$ is transmitted over an Additive White Gaussian Noise (AWGN) channel to obtain the IQ vector $\boldsymbol{d}_m$, which is a complex vector that has the same length as $\boldsymbol{c}_m$. Here, the AWGN channel adds noise to the transmit signal according to a zero-mean, independent and identically distributed (i.i.d.) complex Gaussian distribution, with variance N_0. The resultant of the signal-to-noise ratio (SNR) may be represented by SNR $= \frac{E_0}{N_0}$. The effective throughput of the proposed LDPC-CS scheme is calculated as $\eta = H(x_{n,t})\frac{N}{M}R\log_2(\delta)$, where $H(x_{n,t})$ is the entropy of each element in the signal $\boldsymbol{x}_n$ and each element of $\boldsymbol{x}_n$ has the same entropy value of $H(x_{n,t}) = -P_0 \times \log_2(P_0) - (1-P_0) \times \log_2(1-P_0)$, where $P_0 = \frac{N-K'}{N}$ is the *a priori* probability of a particular signal adopting a value of zero in a particular TS. Furthermore, δ is the modulation order. For transmission over a QPSK modulated AWGN channel, the Discrete-input Continuous-output Memoryless Channel (DCMC) capacity is shown in Fig. 2 as a function of its SNR [34]. The capacity bound of the proposed scheme is the SNR, when the channel capacity is equal to the throughput η.

- **Functions performed in the FC:** In the FC, soft demodulation [35] is employed to transform the received vector $\boldsymbol{d}_m$ into a vector of Logarithmic Likelihood Ratios (LLRs) $\tilde{\boldsymbol{b}}_m$, having a length E. Before decoding the LLRs using LDPC decoding, rate dematching is applied in order to obtain the channel LLR vector $\tilde{\boldsymbol{a}}_m$, which has a length of U. These channel LLRs $\tilde{\boldsymbol{a}}_m$ are entered into the LDPC decoder, which performs iterative decoding in order to obtain a vector of information LLRs $\tilde{\boldsymbol{y}}_m^e$, having the length T. The process of LDPC encoding and decoding is detailed in [36]–[38]. The set of all LDPC-decoded information vectors $\tilde{\mathbf{Y}}^e = [\tilde{\boldsymbol{y}}_1^e, \ldots, \tilde{\boldsymbol{y}}_m^e, \ldots, \tilde{\boldsymbol{y}}_M^e]^T$ are then forwarded to the CS-FC, which decompresses the signal on a TS by TS basis in order to obtain the set of estimated bit sequences $\hat{\mathbf{X}} = [\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_m, \ldots, \hat{\boldsymbol{x}}_N]^T$.

For separate LDPC-CS decoding, the M LLR vectors are forwarded to the CS-FC after LDPC decoding, but there is no further communication between the LDPC decoder and CS-FC after this. Hence, separate LDPC-CS decoding serves as a benchmarker for the proposed scheme, where there are several iterations between the LDPC decoder and CS decoder, as discussed in Section III.

[2]The length of $\boldsymbol{a}_m$ is obtained according to the size of the LDPC parity check matrix $\mathbf{H} \in \{0,1\}^{W \times U}$, where U is the number of natively LDPC-encoded bits and W is given by $U - T'$, and T' is the number of information bits plus some processed zero padding, as required. More specifically, the dimensions of the parity check matrix depend on the length of input information bits T and the choice of the base graph defined in [24]. Furthermore, there are 51 lifting sizes and 2 base graphs in [24], which allows an appropriate value for T' to be selected, which is close to T.

III. Iterative joint LDPC-CS decoding

In this section, we introduce our novel iterative joint LDPC-CS decoding scheme. More specifically, Section III-A provides the top level diagram of the scheme, while Section III-B and Section III-C detail the operations of the LDPC and CS components, respectively.

A. Top level diagram of iterative joint LDPC-CS decoding

Fig. 3 shows the structure of the proposed iterative joint LDPC-CS decoding scheme, which comprises two main parts. The first part is the LDPC decoder outputting the M LLR vectors, which pertain to the LDPC encoded compressed bit sequences. The LDPC decoding allows each of the TSs to help each other's reconstruction indirectly. The other part is the CS decoder, which is used for detecting the M compressed bit sequences, with consideration of the relationships between the M IoTNs and the N signal bit sequences. The CS decoding allows each of the IoTNs to help the reconstruction of each other indirectly. Four different decoders are implemented. More specifically, the Variable Node Decoder I (VNDI) and the Check Node Decoder (CND) of Fig. 3 are used for LDPC decoding, while the Sensor Node Decoder (SND), and VNDII are employed for CS decoding. Each of these decoder components comprises a number of nodes, as shown in Fig. 4. Here, the coordinated dependency exhibits itself among the nodes within a particular layer of a decoder, hence the nodes in the different layers can operate in parallel.

The flow of the iterative joint LDPC-CS decoding process is summarised in Algorithm 1. Here, the LDPC decoding steps comprise initialization, CND update and VNDI update. The LDPC decoder of Fig. 3 has two inputs, where the first input is constituted by the channel LLRs represented by the notation $\tilde{\mathbf{A}} = [\tilde{\boldsymbol{a}}_1, \ldots, \tilde{\boldsymbol{a}}_M]$, which is a matrix comprising M channel LLR vectors $\tilde{\boldsymbol{a}}_1, \ldots, \tilde{\boldsymbol{a}}_M$, where each vector has U, elements $\tilde{a}_{m,1}, \ldots, \tilde{a}_{m,U}$. Each of the M channel LLR vectors is obtained by demodulating and rate-dematching the bit sequence transmitted by the corresponding one of the M IoTNs. Following rate dematching, channel LLRs $\tilde{\mathbf{A}}$ are forwarded to the LDPC decoder. More specifically, the channel LLRs $\tilde{\mathbf{A}}$ are input to the VNs I of VNDI as shown in Fig. 4. Here, VNDI is of dimension $M \times U$ VN I, represented as VI in Fig. 4. The other input of the LDPC decoder of Fig. 3 is the *a priori* LLR matrix $\tilde{\mathbf{Y}}^a \in \mathbb{R}^{M \times T}$, which only contains the knowledge of the non-equiprobable bit probabilities in the first iteration, which will then be updated during the subsequent iterations, when the CS decoder can provide feedback for the LDPC decoder.

In the initialization step, VNDI combines the two sets of input LLRs, namely the channel LLRs $\tilde{\mathbf{A}}$ and the *a priori* LLRs $\tilde{\mathbf{Y}}^a$. The output of VNDI comprises the *extrinsic* LLR matrix $\tilde{\mathbf{M}}^e \in \mathbb{R}^{M \times G}$, which is passed to the CND through

the factor graph connectivity based on the LDPC parity check matrix $\mathbf{H}$, shown in Fig. 4 to create the *a priori* LLR matrix $\tilde{\mathbf{N}}^a \in \mathbb{R}^{M\times G}$ of the CND. Here, G is equal to the number of non-zero elements in the parity check matrix $\mathbf{H}$. In response, the CND generates the *extrinsic* LLR matrix $\tilde{\mathbf{N}}^e \in \mathbb{R}^{M\times G}$, which is passed back through the factor graph connectivity in order to produce the *a priori* LLR matrix $\tilde{\mathbf{M}}^a \in \mathbb{R}^{M\times G}$ as the input of the VNDI. As this point, the VNDI may be operated again having three inputs, namely the channel LLRs $\tilde{\mathbf{A}}$, the *a priori* LLRs $\tilde{\mathbf{Y}}^a$ and the *a priori* LLRs $\tilde{\mathbf{M}}^a$ gleaned from the CND. The detailed operations performed within the CND and VNDI will be discussed in Section III-B.

In summary, LDPC decoding is an iterative decoding process, in which the operation of the VNDI and CND is alternated for i_{LDPC} number of iterations, as shown in Algorithm 1. Following this, VNDI generates an *extrinsic* LLR matrix $\tilde{\mathbf{Y}}^e \in \mathbb{R}^{M\times T}$, which is passed through the factor graph connectivity between VNDI and SND, as shown in Fig. 4.

The output of the VNDI in the LDPC decoder becomes the *a priori* LLRs $\tilde{\mathbf{Z}}^a$ of the SND in the CS decoder. However, the *a priori* knowledge $\mathbf{L} = [\boldsymbol{l}, \ldots, \boldsymbol{l}] \in \mathbb{R}^{M\times T}$ is added into $\tilde{\mathbf{Z}}^a$ before it is entered into the SND, where $\mathbf{L}$ is an LLR matrix, which contains all *a priori* knowledge concerning the M non-equiprobable compressed bit sequences ($\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m, \ldots, \boldsymbol{y}_M$). More specifically, each element of $\mathbf{L}$ characterizes the probability of the corresponding bits in the compressed bit sequences $\boldsymbol{y}_m$ that have a non-zero value.

As in the LDPC decoder, there are also three steps in CS decoding. The first step is initialization, in which the *a priori* LLR matrix $\tilde{\mathbf{\Gamma}}^a \in \mathbb{R}^{M\times F}$ acting as the input of the SND is set to an all zero matrix. Here, F quantifies the number of non-zero elements of the sparse sensing matrix $\mathbf{\Phi}$. Then, the SND of Fig. 3 takes the *a priori* LLRs $\tilde{\mathbf{Z}}^a$ and $\tilde{\mathbf{\Gamma}}^a$ as its inputs and generates the corresponding *extrinsic* LLRs $\tilde{\mathbf{\Gamma}}^e \in \mathbb{R}^{M\times F}$, as it will be described in Section III-C. These *extrinsic* LLRs are then passed through the factor graph connectivity between the SND and VNDII as shown in Fig. 4, and are stored in the sensing matrix $\mathbf{\Phi}$. The reordered LLRs become the *a priori* LLR matrix $\tilde{\mathbf{\Lambda}}^a \in \mathbb{R}^{M\times F}$ of the VNDII. In response, the VNDII generates the matrix of *extrinsic* LLRs $\tilde{\mathbf{\Lambda}}^e$ having the same dimensions as $\tilde{\mathbf{\Lambda}}^a$, as it will be discussed in Section III-C.

Similar to the LDPC decoder, the CS decoder also employs an iterative decoding process, with the *extrinsic* LLR matrix $\tilde{\mathbf{\Lambda}}^e$ becoming an updated *a priori* LLR matrix $\tilde{\mathbf{\Gamma}}^a$ for the SND after reordered. In the case of separate LDPC-CS decoding, we may perform a number of iterations (i_{CS}) within the CS decoder, before the VNDII arrives at its final decision for the estimated bit matrix $\hat{\mathbf{X}} \in \{0,1\}^{N\times T}$. However, in iterative joint LDPC-CS decoding, once the required number of iterations have been performed within the CS decoder, the SND may forward the *extrinsic* LLR matrix $\tilde{\mathbf{Z}}^e$ to the LDPC decoder via the factor graph connectivity between the VNDI and SND as shown in Fig. 4. In this way, there may be iterations between the LDPC and CS decoder, where the LDPC decoder performs its internal iterations, and the CS decoder performs its own internal iterations within each of the system level iterations. The system-level iterations represent the iterations between the LDPC and CS decoder, termed as the LDPC-CS iteration. Following the computation of the required number ($i_{LDPC-CS}$) of LDPC-CS iterations in the iterative joint LDPC-CS decoder, the VNDII of Fig. 3 will obtain its final decision for the estimated bit matrix $\hat{\mathbf{X}} \in \{0,1\}^{N\times T}$ based upon the *a posteriori* LLRs $\tilde{\mathbf{\Gamma}}^a \in \mathbb{R}^{N\times T}$.

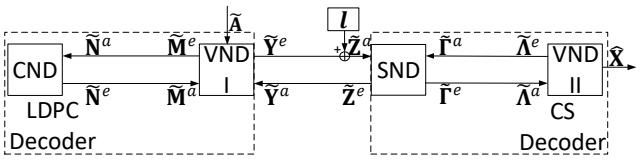

Fig. 3: Block diagram showing the iterative decoding inside the LDPC and CS decoding, as well as between the LDPC and CS decoding.

During the iterative joint LDPC-CS decoding, the initialization steps of the LDPC and CS decoder are performed at the beginning of every LDPC-CS iteration, as shown in Fig. 1. As a result of this, the internal LLRs of the LDPC decoding will be cleared up and the LDPC decoder will start afresh using the new LLRs provided by the CS decoder in each iteration, and vice versa. A practical benefit of this is that less memory is required for storing the internal state between the LDPC-CS iterations [39], [40] and it also prevents the iterative decoder from converging to a local optimum. The detailed operations of the CND, VNDI, SND and VNDII of Fig. 3 are detailed in Section III-B.

Algorithm 1: Iterative joint LDPC-CS decoding algorithm

Input: $\mathbf{H}$, K', $\boldsymbol{d} = [d_1, d_2, \ldots, d_M]^T$, i_{LDPC}, i_{CS}, $i_{LDPC-CS}$, N, M, T, $\mathbf{\Phi}$, $\mathbf{Q}_1, \ldots, \mathbf{Q}_M$, $\boldsymbol{l} = [l_1, \ldots, l_M]^T$, $\tilde{\mathbf{A}}$

Output: Estimated binary signals $\hat{\mathbf{X}}$

```
for i = 1 to i_LDPC-CS do
    Initialization for LDPC decoding;
    for ii = 1 to i_LDPC do
        CND update in the LDPC decoding;
        VNDI update in the LDPC decoding;
    end
    Initialization for CS decoding;
    SND update in the CS decoding;
    for jj = 1 to i_CS do
        VNDII update in the CS decoding;
        SND update in the CS decoding;
    end
end
Perform final decision to obtain the estimated binary signal X̂;
```

B. LDPC decoding

As mentioned in Section III, there are three steps in the LDPC decoding process, namely initialization, CND update and VNDI update. Meanwhile, as shown in Fig. 4, the LDPC decoding process comprises M layers, where each layer processes the information gleaned from one IoTN using a number

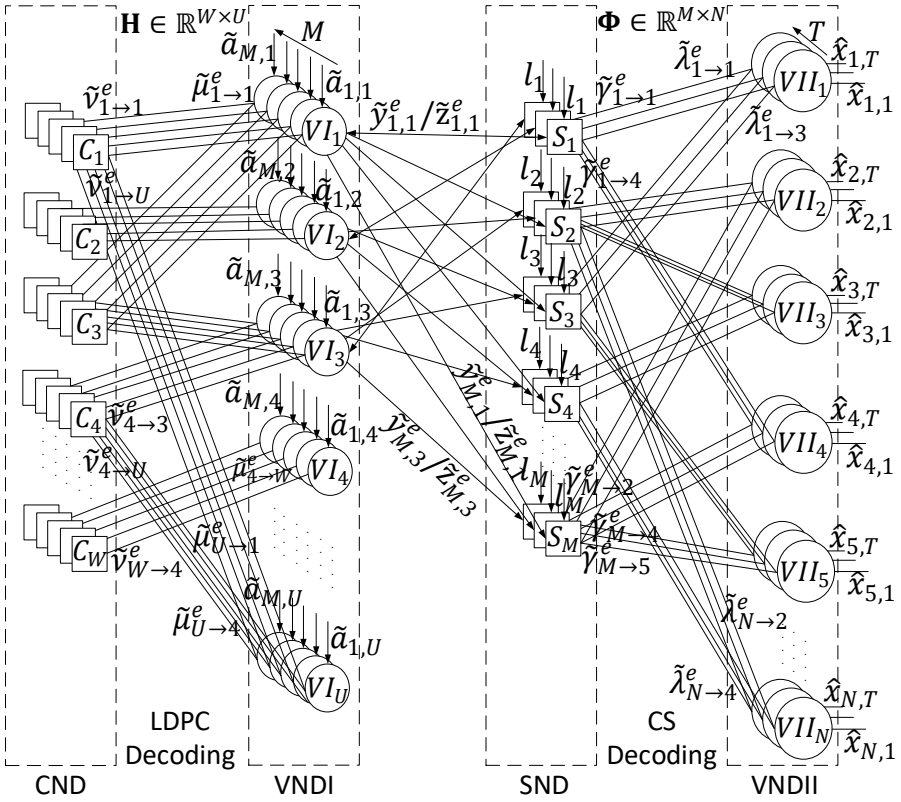

Fig. 4: Factor graph illustrating the connectivity and iterative information exchange between VNDI, CND, SND and VNDII.

of parallel CNs and VNs I. Each of the nodes in a layer assists in the LDPC decoding of a corresponding bit sequence $\boldsymbol{y}_m$, where $m \in [1, M]$ and each of the M layers is operated in parallel. More specifically, every CN of the CND employs the same Sum-Product Algorithm (SPA) operation [41], while each VN in VNDI performs the same sum operation. Hence, in this section, we discuss the CND update and VNDI update based upon the operation of a single CN having an index w ($w \in [1, W]$) and a single VN having an index u ($u \in [1, U]$) in the m^{th} layer.

At the beginning of LDPC decoding, the initialization step is performed, as mentioned in Section III. In this step, the VN having an index u is provided with a channel LLR by the rate dematching block shown in Fig. 1. In each edge between the CND and the VNDI as characterized by the indices (w, u), where we have $h_{w,u} = 1$

$$\tilde{\mu}^e_{u\to w} = \begin{cases} \tilde{a}_{m,u} + \tilde{y}^a_{m,u} & \text{if } 1 \le u \le T \text{ (\textit{i.e.} variable node } u \\ & \text{corresponds to a systematic bit)} \\ \tilde{a}_{m,u} & \text{if } T < u \le U \text{ (\textit{i.e.} variable node } u \\ & \text{corresponds to a parity bit)} \end{cases} \tag{2}$$

where $\tilde{\mu}^e_{u\to w}$ denotes the *extrinsic* LLR that the VN u passes to the CN w, where it becomes the *a priori* LLR $\tilde{\nu}^a_{u\to w}$. Here, $\tilde{a}_{m,u}$ represents the channel LLRs provided for the VN u, while $\tilde{y}^a_{m,u}$ represents the *a priori* LLRs provided by the CS decoder for the VN u in support of LDPC decoding corresponding to the m^{th} IoTN. We initialize $\tilde{y}^a_{m,u} = l_m = \ln\left(\frac{P_{0m}}{1-P_{0m}}\right)$ according to Assumptions 1 and 2. This captures the *a priori* knowledge of the non-equiprobable bit values. More specifically, the m^{th} IoTN can calculate its probability of having zero valued bits $P_{0m} = (P_0)^{d_m}$, where P_0 is the *a priori* probability of having zero values in each TS and d_m is the degree of the m^{th} IoTN. Furthermore, the probability knowledge of all M IoTNs are stored in $\boldsymbol{l} = [l_1, \ldots, l_m, \ldots, l_M]^T$.

Following the completion of LDPC initialization, the CN update is performed. A CN combines the LLRs from its connected VNs using the boxplus operator [41], which is employed as follows when having two arguments

$$\begin{aligned} \tilde{\alpha} \boxplus \tilde{\beta} &= \text{sign}(\tilde{\alpha})\text{sign}(\tilde{\beta}) \min(|\tilde{\alpha}|, \left|\tilde{\beta}\right|) \log(1 + e^{-|\tilde{\alpha}+\tilde{\beta}|}) \\ &\quad - \log(1 + e^{-|\tilde{\alpha}-\tilde{\beta}|}) \\ &\approx \text{sign}(\tilde{\alpha})\text{sign}(\tilde{\beta}) \min(|\tilde{\alpha}|, \left|\tilde{\beta}\right|); \end{aligned} \tag{3}$$

Note that since the boxplus operator is also associative, the function may also be extended to a high number of inputs, as follows for the case of calculating the *extrinsic* LLRs $\tilde{\nu}^e_{w\to u}$ of the w^{th} CN in the CND

$$\tilde{\nu}^e_{w\to u} = \left(\prod_{u' \in \mathcal{V}(w)\setminus u} \text{sign}(\tilde{\nu}^a_{u'\to w}) \right) \times \min_{u' \in \mathcal{V}(w)\setminus u} (\left|\tilde{\nu}^a_{u'\to w}\right|) \tag{4}$$

where $\mathcal{V}(w)$ represents the set of VNs that connect to the w^{th} CN, and $\mathcal{V}(w)\backslash u$ denotes the set $\mathcal{V}(w)$ when excluding the u^{th} VN.

Once the CN update has been completed and the LLRs have been passed to VN I, LDPC decoder will perform the VN I update. For the u^{th} VN in the LDPC decoder, we compute the *extrinsic* LLRs that will be passed to the CND by employing the *a priori* LLR $\tilde{\mu}^a_{u\to w}$ that is received from the w^{th} CN by the u^{th} VN, according to

$$\tilde{\mu}^e_{u\to w} = \begin{cases} \tilde{a}_{m,u} + \tilde{y}^a_{m,u} + \sum_{w'\in\mathcal{C}(u)\backslash w} \tilde{\mu}^a_{w'\to u} & \text{if } 1 \le u \le T \text{(i.e. variable node } u \text{ corresponds to a systematic bit)} \\ \tilde{a}_{m,u} + \sum_{w'\in\mathcal{C}(u)\backslash w} \tilde{\mu}^a_{w'\to u} & \text{if } T < u \le U \text{(i.e. variable node } u \text{ corresponds to a parity bit);} \end{cases} \tag{5}$$

Furthermore, we compute the *extrinsic* LLR $\tilde{y}^e_{m,u}$ that is passed to the CS decoder of Fig. 4 by the u^{th} VN for the m^{th} SN, according to

$$\tilde{y}^e_{m,u} = \tilde{a}_{m,u} + \sum_{w\in\mathcal{C}(u)} \tilde{\mu}^a_{w\to u}, \tag{6}$$

where $\mathcal{C}(u)$ represents the set of CNs that connect to the u^{th} VN, and $\mathcal{C}(u)\backslash w$ denotes the set $\mathcal{C}(u)$ when excluding the c^{th} CN. However, the *extrinsic* LLR $\tilde{y}^e_{m,u}$, which is passed to the SN of Fig. 4, is only computed for the systematic bits, where $1 \le u \le T$.

C. CS decoding

Similar to LDPC decoding, the SND and VNDII of Fig. 3 comprise multiple SNs and VNs. More specifically, CS decoding is constituted of T layers, where each layer corresponds to a different TS. Here, the set of layers are represented by a set of independent factor graphs, where the VNs and SNs of each layer are not connected directly to the VNs and SNs in the other layers. Similar to the CND and VNDI in the LDPC decoder, each SN in the SND processes the inputs provided by its connected VNs. Likewise, each VN in the VNDII processes the inputs provided by the connected SNs of the SND. Since each SN performs the same operations with each other and since each VN performs the same operations with each other, here we describe the SND update based on a single SN having the index m ($m \in [1, M]$) and discuss the VNDII update based on a single VN having the index n ($n \in [1, N]$). Both of these are considered for the case of the t^{th} ($t \in [1, T]$) TS.

As in LDPC decoding, the first step performed by each iteration of the CS decoder is initialization. In order to initialize the m^{th} SN in the t^{th} TS, we set the *a priori* LLR to $\tilde{z}^a_{m,t} = \tilde{y}^e_{m,u} + l_m$, where $\tilde{y}^e_{m,u}$ is the *extrinsic* LLR provided by the u^{th} VN in the m^{th} layer of the LDPC decoder, while l_m is the *a prior* knowledge of the m^{th} IoTN detecting no signals. The SN connects to the u^{th} VN I in the m^{th} layer of the LDPC decoder, which is the m^{th} SN in the t^{th} TS of the CS decoder in the case of $t = u$. In addition to the connection to the VN of the LDPC decoder, the m^{th} IoTN is also connected to the VNs of the CS decoding in VNDII. However, in the initialization step, the *a priori* LLRs $\boldsymbol{\gamma}^a_m$ provided to the m^{th} SN in the t^{th} TS by VNDII are set to zeros.

Following the initialization of the CS decoder, the SN updates are carried out. In [42], we proposed an algorithm for SN updates for the scenario, when the received signals are real values. However, in this treatise the SN updates appropriately modified to fit our scheme associated with time-varying binary signals. According to Assumption 1 in Section II, the m^{th} SN has knowledge of its connections to the various signals and hence all possible permutations of the on-and-off state of the signals can be listed and stored in a matrix $\mathbf{Q}_m$. Let us consider an example based on Fig. 4, in which the SN S_2 connects to the second, third and last VNs, and hence has a degree $d_2 = 3$. According to Assumption 2, we may assume that the FC has knowledge that there are only $K = K' = 2$ non-zero elements in the t^{th} TS. By exploiting this, the SN can be sure that only 0, 1 or 2 of the connected VNs may have non-zero elements, and hence we may determine the permutation matrix $\mathbf{Q}_2$ of S_2, according to

$$\boldsymbol{q}_2 = \begin{matrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{matrix}$$
$$\mathbf{Q}_2 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{7}$$

Each column of $\mathbf{Q}_2$ represents a specific permutation of the on-and-off state of the connected VNs, where the number of columns is given by $\sum_{k=0}^{\min\{K,d_2\}} \binom{d_2}{k}$. Each row represents all possibilities for a specific one of the VN connected to the 2^{nd} SN. Since the SN performs an OR function, the expected output of the SN for different permutations is given by the vector $\boldsymbol{q}_2 = [q_2(1), \ldots, q_2(7)]$, where the output of the SN becomes the input to the LDPC encoder. For example, the first permutation in the matrix $\mathbf{Q}_2$ predicts that in the t^{th} TS, the signals $x_{2,t}$, $x_{3,t}$ and $x_{N,t}$ connected to the m^{th} IoTN all have zero values. In this case, the expected signal output of the IoTN is zero, because it employs the OR function. By contrast, the second permutation predicts that $x_{2,t}$, $x_{3,t}$ have zero values and $x_{N,t}$ has a non-zero value, resulting in an expected signal output value of $q_2(2) = 1$, owing to the action of the OR function applied by the SN. Similarly, the rest of the permutations are analogous and each has an expected IoTN output value of 1. In the proposed scheme, Assumption 1 states that the connections between the signals and IoTNs remain constant across the T TSs, and hence each IoTNs has the same permutation matrix in every TS, even though the particular permutations adopted by the scheme will be changing from TS to TS.

The SN update begins by calculating the probabilities of the permutations in the matrix $\mathbf{Q}_2$ based on the *a priori* LLRs provided by the LDPC decoder and the connected VNs of the VNDII. The probability of the $p^{th} \in [1, P_m = \sum_{k=0}^{\min\{K,d_m\}} \binom{d_m}{k}]$ permutation in the permutation matrix $\mathbf{Q}_m$ is calculated as

$$\eta(\boldsymbol{Q}_m(:,p)) = \begin{cases} \sum_{n'\in\mathcal{N}(m)} \tilde{\gamma}^a_{n'\to m} + \tilde{z}^a_{m,t} & \text{if } q_m(p) = 0 \\ \sum_{n'\in\mathcal{N}(m)} \tilde{\gamma}^a_{n'\to m} & \text{otherwise} \end{cases} \tag{8}$$

where $\mathcal{N}(m)$ denotes the set of the VNs in the CS decoder that connect to the m^{th} SN, having a predicted value of zero in the p^{th} permutation. Here, $\tilde{\gamma}^a_{n'\to m}$ is the *a priori* LLR provided by the n'^{th} VN, while $\tilde{z}^a_{m,t}$ is the *a priori* LLR provided by the LDPC decoder. More specifically, in the example above

based on S_2, the 2^{nd} permutation has $x_{2,t}$ and $x_{3,t}$ predicted to adopt zero values, and hence $\mathcal{N}(m)$ includes the second and third VNs. In this case, the adoption of the second permutation for S_2 in the t^{th} TS results in $Pr_2 = \tilde{\gamma}^a_{2\to2} + \tilde{\gamma}^a_{3\to2}$, where the $\tilde{z}^a_{2,t}$ is not added because we have $q_2(2) = 1$.

After quantifying the permutation probabilities, the *extrinsic* LLRs gleaned from the m^{th} SN by its connected VNs in the LDPC decoder and CS decoder may be calculated as

$$\tilde{\gamma}^p_{m\to n} = \max\left(\boldsymbol{\eta}\left[\boldsymbol{Q}_m(r,p)=0\right]\right) - \max\left(\boldsymbol{\eta}\left[\boldsymbol{Q}_m(r,p)=1\right]\right) \tag{9}$$

$$\tilde{\gamma}^e_{m\to n} = \tilde{\gamma}^p_{m\to n} - \tilde{\gamma}^a_{n\to m} \tag{10}$$

$$\widetilde{z}^p_{m,t} = \eta(1) - \max(\boldsymbol{\eta}(2:P_m)) \tag{11}$$

$$\widetilde{z}^e_{m,t} = \widetilde{z}^p_{m,t} - \widetilde{z}^a_{m,t} \tag{12}$$

where $\boldsymbol{\eta}$ is a vector of length $P_m = \sum_{k=0}^{\min\{K,d_m\}} \binom{d_m}{k}$, which stores the probabilities of all possible permutations for the m^{th} SN in the t^{th} TS. Furthermore, $\boldsymbol{\eta}\left[\boldsymbol{Q}_m(r,p)=0\right]$ denotes the vector that contains the probabilities of the permutations in the permutation matrix $\mathbf{Q}_m$, in which the q^{th} column of $\mathbf{Q}_m$ has a zero value. Similarly, $\boldsymbol{\eta}\left[\boldsymbol{Q}_m(r,p)=1\right]$ denotes a vector that contains the probabilities associated with the specific rows of permutation matrix $\mathbf{Q}_m$, in which the p^{th} column has non-zero values. Here, $\tilde{\gamma}^p_{m\to n}$ and $\tilde{\gamma}^e_{m\to n}$ are the *a posteriori* LLR and *extrinsic* LLR provided by the m^{th} SN for the n^{th} VN of VNDII respectively. Likewise, $\tilde{\gamma}^a_{n\to m}$ is the *a priori* LLR provided by the n^{th} VN in VNDII for the m^{th} SN. However, $\widetilde{z}^p_{m,t}$ and $\widetilde{z}^e_{m,t}$ are the *a posteriori* LLR and *extrinsic* LLR of the m^{th} SN, which are used for performing message passing between the LDPC and CS decoder. Here, $\eta(1)$ represents the first value of $\boldsymbol{\eta}$ and $\boldsymbol{\eta}(2:P_m)$ is the vector containing all values except for the first value of $\boldsymbol{\eta}$. In the case of iterative joint decoding between the LDPC and CS decoder, $\widetilde{z}^e_{m,t}$ is the *extrinsic* LLR provided by the m^{th} SN of the CS decoder for the u^{th} VN of the VNDI gleaned from the LDPC decoder. By contrast, there are no iterations between the LDPC and CS decoder in the case of separate decoding, and so there is not necessary to calculate the *extrinsic* LLRs $\widetilde{y}^e_{m,t}$, $\widetilde{z}^e_{m,t}$ since they are not exchanged between the VNs of VNDI and the SNs.

Following the completion of the SN update, the CS decoder performs an update for VNDII. Here, the CS VNs of VNDII employ the same operations as the LDPC VNs of VNDI. More specifically, the n^{th} VN, computes

$$\tilde{\lambda}^e_{n\to m} = \sum_{m'\in\mathcal{M}(n)\backslash m} \tilde{\lambda}^a_{m'\to n} \tag{13}$$

$$\tilde{\Lambda}^p_{n,t} = \sum_{m\in\mathcal{M}(n)} \tilde{\lambda}^a_{m\to n} \tag{14}$$

where $\mathcal{M}(n)$ denotes the set of SNs that are connected to the n^{th} VN in VNDII, and $\mathcal{M}(n)\backslash m$ represents the set $\mathcal{M}(n)$, when excluding the m^{th} SN. Finally, $\tilde{\Lambda}^p_{n,t}$ is the *a posteriori* LLR of the n^{th} VN in the CS decoder, which is used for deriving the final bit decisions for the t^{th} TS.

The final hard decision is made by exploiting the knowledge that there are only K' non-zero values in each TS, and so we can use the *a posteriori* LLRs for identifying the K' most likely signals to be non-zero. This is achieved by sorting all the *a posteriori* LLRs of VNDII in decreasing order and then selecting those having the most positive values and decoding them as non-zeros.

IV. Complexity analysis

The complexity of the proposed iterative joint LDPC-CS decoding scheme may be quantified in terms of the number of Add, Compare and Select (ACS) operations performed across the successive iterations by the VNDI, CND, SND and VNDII components of Fig. 3. The complexity of each operation of a single VN u both in the VNDI of the LDPC decoder and in the VNDII of the CS decoder may be calculated as [43]

$$\mathcal{C}_{VI_u} = \mathcal{C}_{VII_u} = 3(d_{V_u} - 1); \tag{15}$$

where d_{V_u} is the degree of the VN u. Similarly, the complexity of each operation of a single CN w in the LDPC decoder is given by [44]

$$\mathcal{C}_{C_w} = 3(d_{C_w} - 2). \tag{16}$$

where d_{C_w} is the degree of the CN w. The different nodes in VNDI and CND have different degrees depending on the connectivity of the factor graphs, but, all VNs in VNDII have the same degree K_s.

Similarly, the complexity of each visit to a single SN m having a degree of d_m by calculating Eq. (8) and Eq. (9) is given by

$$\mathcal{C}_{S_m} = 2d_m \times P_m - \sum_{k=0}^{\min\{K,d_m\}} k\binom{d_m}{k}, \tag{17}$$

where $P_m = \sum_{k=0}^{\min\{K,d_m\}} \binom{d_m}{k}$ is the number of permutations considered by the m^{th} IoTN. The overall complexity of the iterative joint decoding considering all the VNs, CNs and SNs may be calculated as

$$\begin{aligned}\mathcal{C}_{LDPC-CS} =& i_{LDPC-CS} \times (i_{LDPC} \times (\sum_{v=1}^{U} \mathcal{C}_{VI_u} + \sum_{c=1}^{W} \mathcal{C}_{C_w}) \\ &+ (i_{CS}+1) \times \sum_{m=1}^{M} \mathcal{C}_{S_m} + i_{CS} \times \sum_{n=1}^{N} \mathcal{C}_{VII_n}).\end{aligned} \tag{18}$$

V. EXIT chart analysis of the proposed LDPC-CS scheme

In this section, we use EXIT charts for analyzing the soft-in/soft-out consequence behavior of the proposed LDPC-CS scheme. We begin by introducing the novel non-equiprobable EXIT chart concept, which may be used for analyzing the non-equiprobable bit sequences that are considered by the proposed scheme. More specifically, Section V-A will discuss the methods of generating LLRs for these non-equiprobable bit sequences, as well as the calculator of the associated Mutual Information (MI). Section V-B will introduce the process of generating the LDPC EXIT functions, while Section V-C will provide the corresponding discussion for the CS EXIT functions. In Section V-D, we characterize the iterative decoding trajectories, which may be plotted in the EXIT charts in order

to characterize the exchange *extrinsic* information between the LDPC and CS decoder in successive iterations. Finally, Section V-E presents our results and discussions of the EXIT charts introduced.

A. EXIT functions for non-equiprobable bit sequence

EXIT charts constitute powerful tools of visializing the convergence behaviour of decoders. They help us validate our decoding scheme at a significantly reduced simulation complexity. Typically, an EXIT chart is characterized as a pair of EXIT functions corresponding to a pair of constituent decoders that engage in iterative *extrinsic* information exchange, as well as a decoding trajectory, which characterizes the progression of the iterative exchange as the iteration index. Here, the quality of the iteratively exchanged *a priori* and *extrinsic* information is quantified by the MI between the corresponding LLR sequences and the corresponding hypothesized bit sequences availing from the transmitter [45]. The *a priori* LLRs are generated artificially as the inputs of the corresponding constituent decoder by relying on our EXIT-chart software to be detailed late in Fig. 5 and 6. However, when the bits in the transmitter do not have equiprobable values of 0 and 1, special measures must be taken. In this section, we use an example for characterizing the relationship between a non-equiprobable bit sequence $\boldsymbol{\omega}$, a corresponding sequence of LLRs $\tilde{\boldsymbol{\omega}}$ and the MI $I(\boldsymbol{\omega};\tilde{\boldsymbol{\omega}})$. When generating the artificial LLRs, we must consider the probability $P_{0\omega}$ of the bit sequence $\boldsymbol{\omega}$ adopting the value 0, according to

$$\tilde{\omega}_i = \frac{\sigma_\omega^2}{2}(1-2\omega_i) + N_\omega + \log\left(\frac{P_{0\omega}}{1-P_{0\omega}}\right), \tag{19}$$

where, N_ω is an independent Gaussian random variable with zero-mean and a variance of σ_ω^2 [26], [45].

The MI between the non-equiprobable bit sequence $\boldsymbol{\omega}$ and the corresponding LLR sequence $\tilde{\boldsymbol{\omega}}$ may be calculated using the so-called histogram-based method of [28]

$$\begin{aligned} I(\boldsymbol{\omega};\tilde{\boldsymbol{\omega}}) = &\sum_{\omega_i=0,1}\int_{-\infty}^{+\infty} f_{\tilde{\boldsymbol{\omega}}|\boldsymbol{\omega}}(\tilde{\omega}_i|\omega_i) f_{\boldsymbol{\omega}}(\omega_i) \\ &\cdot \log_2\left(\frac{f_{\tilde{\boldsymbol{\omega}}|\boldsymbol{\omega}}(\tilde{\omega}_i|\omega_i)}{f_{\tilde{\boldsymbol{\omega}}}(\tilde{\omega}_i)}\right) d\tilde{\omega}_i, \end{aligned} \tag{20}$$

where the integral may be implemented as a summation over histogram bins. Here, each bit of the non-equiprobable $\boldsymbol{\omega}$ obeys Bernoulli distribution, and the probability mass function of the i^{th} element of the bit sequence $\boldsymbol{\omega}$ is

$$f_{\boldsymbol{\omega}}(\omega_i) = (P_{0\omega})^{\omega_i}\,(1-P_{0\omega})^{(1-\omega_i)}. \tag{21}$$

The conditional Probability Density Function (PDF) of the LLR sequence $\tilde{\boldsymbol{\omega}}$ is given by

$$f_{\tilde{\boldsymbol{\omega}}|\boldsymbol{\omega}}(\tilde{\omega}_i|\omega_i) = \frac{e^{-((\tilde{\omega}_i-\sigma_\omega^2/2\times(1-2\omega_i)^2)/2\sigma_\omega^2)}}{\sqrt{2\pi}\sigma_\omega^2}, \tag{22}$$

while that of $\tilde{\boldsymbol{\omega}}$ is given by

$$f_{\tilde{\boldsymbol{\omega}}}(\tilde{\omega}_i) = \int_{-\infty}^{+\infty} f_{\tilde{\boldsymbol{\omega}}|\boldsymbol{\omega}}(\tilde{\omega}_i|\omega_i) f_{\boldsymbol{\omega}}(\omega_i) d_{\omega_i}. \tag{23}$$

For details of the derivation process, please refer to [26].

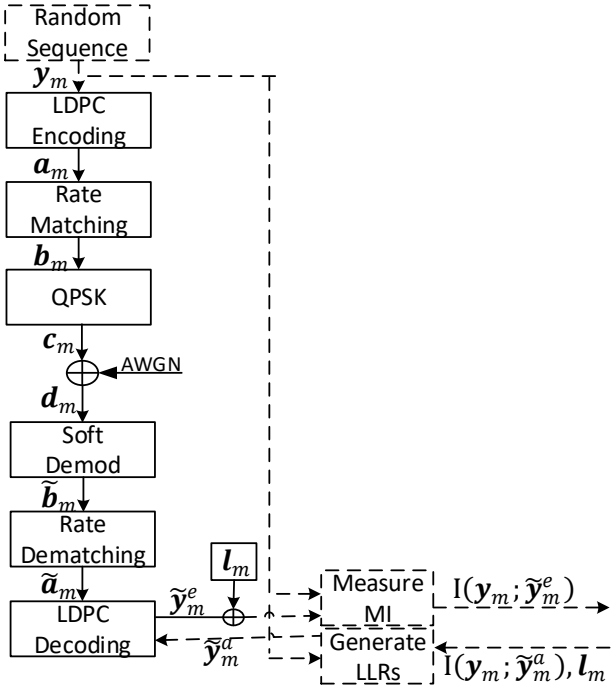

Fig. 5: The schematic depicting the generation of the LDPC EXIT function, where solid lines indicate parts of the real system, while dashed lines represent our EXIT chart software.

B. LDPC EXIT function generation

Fig. 5 depicts the schematic of generating the LDPC EXIT function, which characterizes the MI $I(\boldsymbol{y}_m;\tilde{\boldsymbol{y}}_m^e)$ of the *extrinsic* LLRs $\tilde{\boldsymbol{y}}_m$ output by the LDPC decoder in response to the SNR associated with the channel LLRs $\tilde{\boldsymbol{a}}_m$ and the MI $I(\boldsymbol{y}_m;\tilde{\boldsymbol{y}}_m^a)$ of the *a priori* LLRs $\tilde{\boldsymbol{y}}_m^a$. Here, the process of generating the channel LLRs $\tilde{\boldsymbol{a}}_m$ is similar to that of the discussion in Section II. More specifically, we adopt a random sequence to generate the T elements of the signal vector $\boldsymbol{y}_m$, where each element adopts a value of 0 with the probability P_{0m}. The observation vector $\boldsymbol{y}_m$ comprising T binary IoTN observations is LDPC encoded and rate matched in order to obtain the encoded bit vector $\boldsymbol{b}_m$, which comprises E bits. Following this, QPSK modulation is applied in order to obtain the IQ vector $\boldsymbol{c}_m$ comprising $\frac{E}{2}$ IQ symbols. As shown in Fig. 5, we simulate the transmission of this IQ vector through an AWGN channel, in order to obtain the received IQ vector $\boldsymbol{d}_m$. This is then QPSK demodulated in order to obtain the LLR vector $\tilde{\boldsymbol{b}}_m$, which comprises E number of LLRs. Next, rate dematching is performed in order to obtain the channel LLR vector $\tilde{\boldsymbol{a}}_m$ comprising T elements, before performing LDPC decoder, as discussed in Section III-A and shown in Fig. 4. More specifically, the channel LLRs $\tilde{\boldsymbol{a}}_m$ are provided for the VNs of VNDI, as shown in Fig. 4. Meanwhile, the LDPC decoding is also provided with the *a priori* LLR vector $\tilde{\boldsymbol{y}}_m$ comprising T elements, which is furnished by the CS decoder when employed in the complete system diagram of Fig. 3. However, in the Fig. 5, instead of generating the *a priori* LLR vector $\tilde{\boldsymbol{y}}_m$ artificially based on Eq. (19), we create it as a function of the observation vector $\boldsymbol{y}_m$, so that the LLR vector has the MI of $I(\boldsymbol{y}_m;\tilde{\boldsymbol{y}}_m^a)$. When generating the *a priori* vector

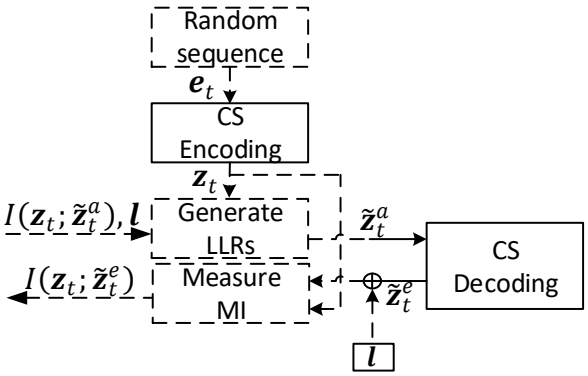

Fig. 6: Schematic of generating the CS EXIT function, where solid lines are used for the parts adopted for the real system and dashed lines represent our EXIT chart software.

$\tilde{\boldsymbol{y}}_m^a$, the knowledge of the probability of each element of $\boldsymbol{y}_m$ adopting a value of 0 is considered. More specifically, the LLR vector $\boldsymbol{l}_m = [l_m, l_m, \ldots, l_m]^T$ is employed, which has the same T number of elements as the signal vector $\boldsymbol{y}_m$, so that $\tilde{\boldsymbol{y}}_m^a$ should include knowledge of P_{0m}. As discussed in Section III-B, we adopt $l_m = \log\left(\frac{P_{0m}}{1-P_{0m}}\right)$, where P_{0m} is the probability of an element in the signal vector $\boldsymbol{y}_m$ having a zero value. Having provided the two inputs for the LDPC decoder of Fig. 5, we perform the operations described in Section III-B, which includes initialization, CN update and VN I update. In response, the vector of T *extrinsic* LLRs $\tilde{\boldsymbol{y}}_m^e$ is generated and the LLR vector $\boldsymbol{l}_m$ is also added to the *extrinsic* LLRs $\tilde{\boldsymbol{y}}_m^e$ before using Eq. (20) for quantifying the corresponding MI. Finally, having calculated the *extrinsic* MI of each SN individually, the final *extrinsic* MI may be calculated as the average value of these M *extrinsic* MI values.

C. CS EXIT function generation

Fig. 6 illustrates our software used for generating the CS EXIT function. As shown in Fig. 4, CS decoding is operated in layers, where each layer processes a different TS using an identical set of operations. Hence, the schematic of Fig. 6 characterizes the processing in the t^{th} ($t \in [1, T]$) layer of the CS decoding, with all other layers are operating in the same way. As for the LDPC EXIT function, the final *extrinsic* MI is given by the average of the *extrinsic* MI values obtained by each of the layers. The generation of the CS EXIT function begins by generating a random sequence $\boldsymbol{e}_t$ comprising N number of bits, which corresponds to the set of signal elements generated in the t^{th} TS, and hence corresponds to the t^{th} column of $\mathbf{X}$. The random sequence $\boldsymbol{e}_t$ is constructed using CS encoding, and then the resultant compressed signal $\boldsymbol{z}_t = [z_{t,1}, \ldots, z_{t,m}, \ldots, z_{t,M}]^T$ is comprised of M bits corresponding to the M observations in the t^{th} TS. The compressed sequence $\boldsymbol{z}_t$ is the input of the LLR function, which generates a sequence $\tilde{\boldsymbol{z}}_t^a$ comprised of M artificial LLRs using Eq. (19), where the MI is given by $I(\boldsymbol{z}_t; \tilde{\boldsymbol{z}}_t^a)$. During the process of generating the LLRs, the LLR constants $\boldsymbol{l} = [l_1, \ldots, l_m, \ldots, l_M]^T$ are added into the *a priori* LLRs $\tilde{\boldsymbol{z}}_t^a$ in order to capture the probability of each compressed bit adopting the value 0, as shown in Fig. 6. Given that the elements of the compressed signal $\boldsymbol{z}_t$ are derived from M observations in the t^{th} TS, and because the probabilities of the corresponding bit to adopt the value of 0 are different, each element l_m of the vector $\boldsymbol{l}$ typically adopts a different value. The artificially generated *a priori* LLRs $\tilde{\boldsymbol{z}}_t^a$ are then entered into the CS decoder, which operates as discussed in Section III-C, and characterized by the structure shown in Fig. 4. This comprises a set of SNs and VNs II, where the CS decoding includes initialization, SN update and VN II update. Finally, the *extrinsic* MI $I(\boldsymbol{z}_t; \tilde{\boldsymbol{z}}_t^e)$ is given by the measured MI function, which compares the compressed bit value $\boldsymbol{z}_t$ to the M *extrinsic* LLRs in the sequence $\tilde{\boldsymbol{z}}_t^e$, where the constant values of $\boldsymbol{l}$ are added in, as shown in Fig. 6 and characterized using Eq. (20).

D. Iterative decoding trajectory

In order to illustrate the iterative exchange of *extrinsic* information between the LDPC and CS decoder, their EXIT functions may be plotted in the same EXIT chart by swapping the axes of CS decoding curves. This is because the *extrinsic* output of the LDPC decoder becomes the *a priori* input of the CS decoder, and vice versa, as seen in Fig. 3. The iterative exchange of *extrinsic* information between the LDPC and CS decoder may be visualized as a staircase-shaped trajectory evolving between the two EXIT functions [45]. Here, the MI calculation using the generation of the trajectory is the same as that used for characterizing the EXIT functions employing Eq. (20).

Each step in the staircase-shaped trajectory represents a specific iteration between the LDPC and CS decoder improving the MI step-by-step. In cases where the EXIT chart tunnel between the LDPC and CS EXIT functions is open, the staircase shaped iterative decoding trajectory succeeds in evolving between the two EXIT functions towards the top right corner of the EXIT chart. In this way, we may quantify the number of iterations required for reliable decoding as the number of the steps in the trajectory within the open EXIT tunnel. By contrast, in case of a closed EXIT chart tunnel, the trajectory is unable to reach the upper right-hand corner. In contrast to the conventional EXIT charts, the upper right-hand corner of the LDPC-CS EXIT function is not at the $(1, 1)$ coordinate, since the *extrinsic* information represents non-equiprobable bit values in the proposed scheme. Instead, it is given by the entropy of the associated non-equiprobable bit sequence. In the proposed system, we have M different non-equiprobable bit sequences each having different entropies, and so the coordinate of the top right corner of the EXIT chart is given by the mean of all M entropies. Here, the entropy of the t^{th} ($t \in [1, T]$) bit in the non-equiprobable bit sequence in $\boldsymbol{y}_m$, where the probability of a particular bit adopting a value of 0 is given by P_{0m}, may be calculated as

$$H(y_{m,t}) = -P_{0,m} \times \log_2(P_{0,m}) - (1 - P_{0,m}) \times \log_2(1 - P_{0,m}). \quad (24)$$

E. EXIT chart results

The EXIT charts presented in this section characterize the trade-off between decoding performance and complexity as a function of the number of iterations performed during LDPC decoding (i_{LDPC}), CS decoding (i_{CS}) and iterative LDPC-CS decoding ($i_{LDPC-CS}$). Explicitly, performing too few iterations leads to poor performance, while performing too many iterations leads to excessive complexity. Hence, the EXIT charts offer an insightful complement to BLER results, since the simulations are much quicker to complete and provide insight into the internal information exchange of the receiver. The MI of both the EXIT functions and trajectories in the EXIT charts are calculated using Eq. (20).

Fig. 7 portrays the LDPC EXIT curves for different numbers of iterations (i_{LDPC}), for the case of using $N = 500$ signals, $M = 150$ IoTNs, $T = 500$ TSs, $K = K' = 5$ non-zero values in each TS and where each VN in Fig. 1 has a degree of $K_s = 5$. Fig. 7 shows that as the number of iterations (i_{LDPC}) increases, the LDPC EXIT function evolves upwards, which is characteristic of high-quality *extrinsic* LLRs and creates a wide open EXIT chart tunnel. However, it may be observed that there are diminishing returns for each additional iteration, with very little extra benefit offered after performing $i_{LDPC} = 16$ iterations. Hence, we set $i_{LDPC} = 16$ as a stop criterion for LDPC decoding to strike an attractive performance vs. complexity trade-off. Fig. 7 also characterizes the CS EXIT functions for various numbers of iterations (i_{CS}). In contrast to the LDPC EXIT function, when the number of iterations (i_{CS}) is increased within the CS decoder, the CS EXIT function moves downwards because of the inverted coordinate system used for the CS decoding. Again, there are diminishing returns for more than $i_{CS} = 3$ iterations. Hence, we recommend the selection of $i_{CS} = 3$ iterations as the stopping criterion of the CS decoder, in order to strike an attractive performance vs. complexity trade-off.

Furthermore, EXIT charts are also capable of characterizing the *extrinsic* information exchange between the LDPC and CS decoder. Fig. 8 characterizes the EXIT functions of the proposed iterative joint LDPC-CS scheme for different channel SNRs, which shows that as the SNR is increased, the tunnel between the LDPC and CS EXIT functions becomes wider. This open tunnel between the two EXIT functions suggests potential converge towards a low BLER. Hence, by determining the lowest SNR at which the EXIT chart tunnel becomes open, we may predict the lowest SNR at which a low BLER may be obtained. As shown in Fig. 8, when the channel SNR is -6 dB, the tunnel between the LDPC and CS EXIT functions is closed, but when the SNR is increased to about -5.2 dB, the EXIT tunnel reaches its threshold between the opening and closing. In cases where the SNR is -4.4 dB, the tunnel is clearly open. This may be confirmed by plotting a trajectory between the LDPC and CS EXIT functions, which is measured after each iteration. The trajectory may characterize the iterative exchange of *extrinsic* LLRs between the CS and LDPC decoder in successive iterations. As the trajectory progresses upwards to the top right in each successive step towards the top right-hand corner of the EXIT chart, the decoding performance becomes better and better. However, if the EXIT function of the CS and LDPC decoder intersect, then the trajectory will be prevented from reaching the top right-hand corner, indicating that iterative decoding convergence to a low BLER is not possible at the corresponding channel SNR. As shown in Fig. 8, when the EXIT tunnel is open at an SNR $= -4.4$ dB, the staircase-shaped trajectory reaches the top

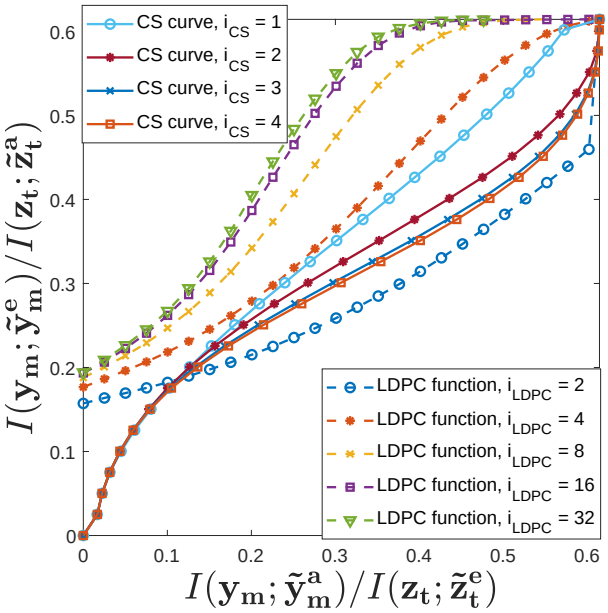

Fig. 7: LDPC/CS EXIT functions for various numbers of iterations during LDPC-CS decoding, for the case of $N = 500$ signals, $M = 150$ IoTNs, sparsity level $K = K' = 5$, the degree of each signal $K_s = 5$, $T = 500$ TSs, coding rate $R = \frac{1}{3}$ and SNR $= -4.4$ dB, when using QPSK modulation for communication over AWGN channels.

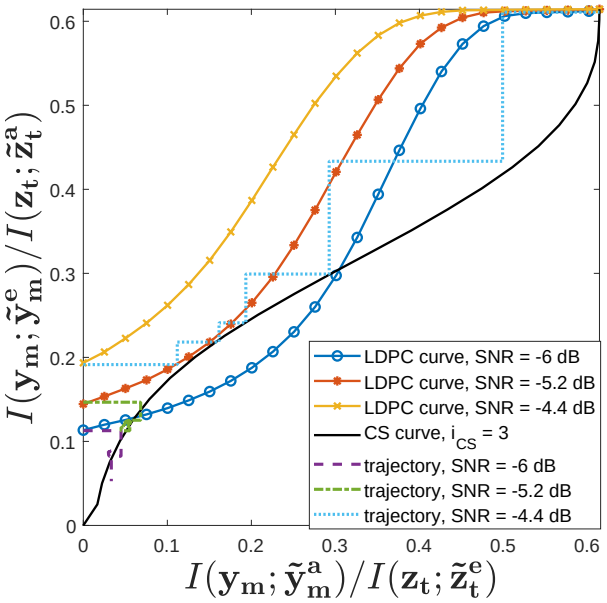

Fig. 8: LDPC/CS EXIT functions and iterative decoding trajectories for various numbers of iterations during LDPC-CS decoding, for the case of $N = 500$ signals, $M = 150$ IoTNs, sparsity level $K = K' = 5$, the degree of each signal $K_s = 5$, $T = 500$ TSs, coding rate $R = \frac{1}{3}$, when using QPSK modulation for communication over AWGN channels.

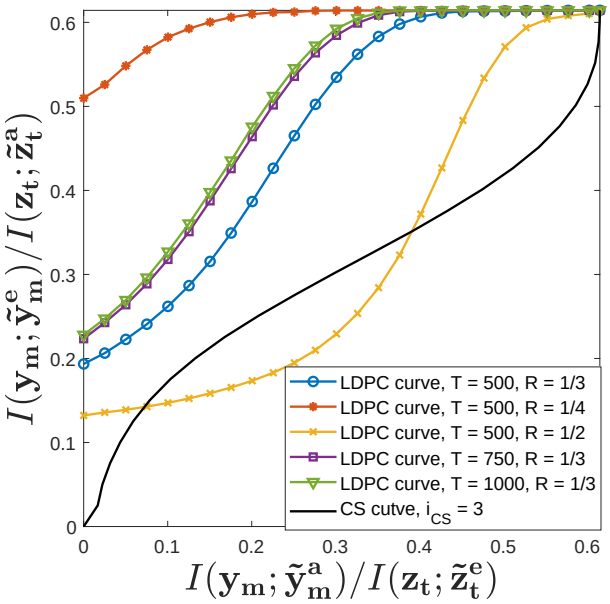

Fig. 9: LDPC/CS EXIT functions for various coding rates and number of TSs T, for the case of $N = 500$ signals, $M = 150$ IoTNs, sparsity level $K = K' = 5$, the degree of each signal $K_s = 5$ and SNR $= -4.4$ dB, when using QPSK modulation over AWGN channels.

right corner, suggesting that a low BLER may be obtained. By contrast, when the tunnel is closed at an SNR of -6 dB, the trajectory converges early and becomes unable to pass through between the intersecting EXIT functions, hence a low BLER is prevented. At an SNR of -5.2 dB, Fig. 8 shows that the EXIT tunnel is very narrow and the trajectory is unable to pass through the tunnel. In addition to predicting whether or not iterative decoding converge to a low BLER is possible, the trajectory can also predict how many iterations between the LDPC and CS decoder are needed. In the example where SNR is -4.4 dB, the trajectory suggests that $i_{LDPC-CS} = 6$ iterations constitutes an attractive stopping criterion for achieving a low BLER. Although, there is a small mismatch between the EXIT functions and trajectories in Fig. 8, we may observe a broad agreement between the various trajectories and EXIT functions.

Fig. 9 characterizes the impact of different block lengths T and coding rates R upon the EXIT function of the LDPC decoder. More specifically, Fig. 9 shows that as the LDPC coding rate is reduced, while maintaining the same block length of $T = 500$ and SNR $= -4.4$ dB, the MI of the LLRs is increased. As a result, the EXIT tunnel opens as the coding rate is reduced, increasing the probability of successful decoding, as expected. Fig. 9 also characterizes the influence of different block lengths on the EXIT functions, while maintaining the same coding rate of $R = \frac{1}{3}$ and SNR $= -4.4$ dB. As expected, a longer LDPC block length gives superior BLER performance, and this becomes particularly evident upon increasing the block length from $T = 500$ to $T = 750$. However, Fig. 9 also shows that there is a diminishing return beyond this block length, namely for $T = 1000$.

VI. SIMULATION RESULTS

As a complement to the EXIT charts provided in Section V-E, this section characterizes the BLER performance of the

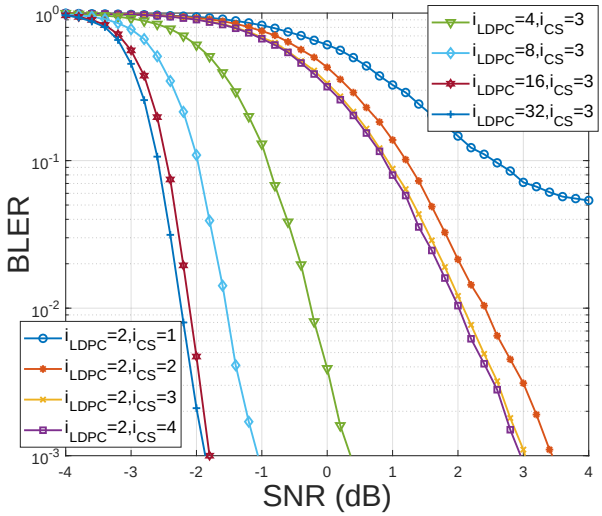

Fig. 10: BLER results for the benchmarker employing separate LDPC-CS decoding, for the case where $N = 500$ signals, $M = 150$ IoTNs, $K = K' = 5$, the degree of each signal $K_s = 5$, $T = 500$ TSs and coding rate $R = \frac{1}{3}$, when using QPSK modulation for communication over AWGN channels.

proposed LDPC-CS scheme, which employs iterative joint LDPC and CS decoding components. More specifically, the BLER plots presented in this section demonstrate that our proposed scheme outperforms the separate LDPC/CS benchmarker at a slightly increased complexity.

Each of the plots in this section characterizes the impact of a different one of the three iteration parameters, namely i_{LDPC}, i_{CS} and $i_{LDPC-CS}$, as defined in Section III-A. Fig. 10 considers the benchmarker, where there are no iterations between the LDPC and CS decoder. Following the discussions of Fig. 10, we introduce Fig. 11, which compares the separate decoding based benchmarker to our proposed iterative joint LDPC-CS decoder, where both adopt a similar complexity. Later, Fig. 12 characterizes how the number of iterations ($i_{LDPC-CS}$) performed between the LDPC and CS decoder influence the performance of the proposed iterative joint decoding scheme. Finally, Fig. 14 characterizes the impact of coding rate and block length upon the performance of the proposed iterative joint decoding scheme. Throughout these investigations, we set the number of signals to $N = 500$, the number of IoTNs to $M = 150$, the sparsity level to $K' = 5$ and each signal is observed by $K_s = 5$ IoTNs.

Fig. 10 characterizes the BLER performance of the benchmarker for an LDPC block length spanning $T = 500$ TSs at a coding rate of $R = \frac{1}{3}$. In order to characterize the influence of the number of the iterations i_{CS} performed within the CS decoder, we maintain a constant number of $i_{LDPC} = 2$ iterations within the LDPC decoder. As shown in Fig. 10, the BLER performance of the benchmarker is improved significantly upon increasing i_{CS} in CS decoding from 1 to 3. However, when the number of iterations is increased beyond $i_{CS} = 3$, the BLER only has minor further improvements. This observation corresponds to that, which may be made based on the CS EXIT functions of Fig. 7, increasing diminishing returns upon increasing the number of iterations beyond $i_{CS} = 3$. Similarly, in order to characterize the effect of the number of iterations i_{LDPC} performed

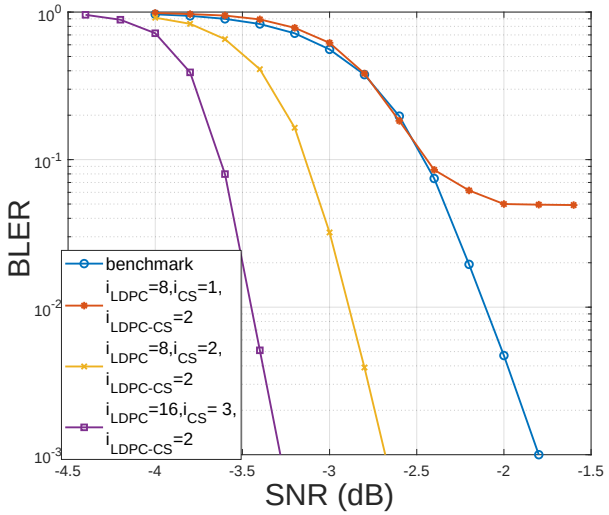

Fig. 11: BLER results for the benchmarker employing separate LDPC-CS decoding and the proposed iterative joint LDPC-CS decoding, with consideration of the complexity of the decoding processes, for the case of $N = 500$ signals, $M = 150$ IoTNs, sparsity level $K = K' = 5$, the degree of each signal $K_s = 5$, $T = 500$ time slots and coding rate $R = \frac{1}{3}$, when using QPSK modulation for communication over AWGN channels.

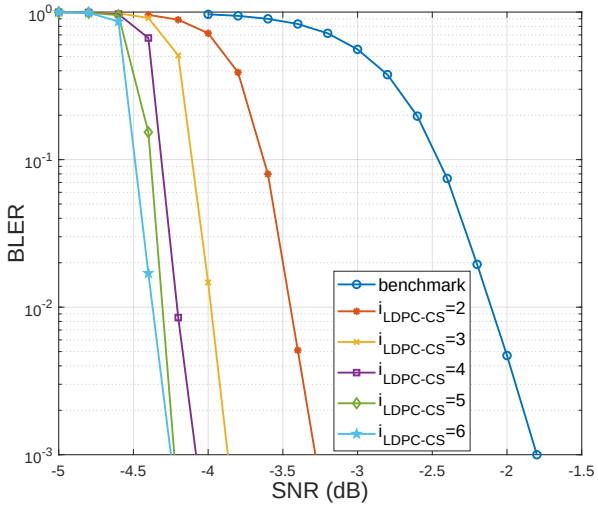

Fig. 12: BLER results of iterative joint LDPC-CS decoding, for the case of $N = 500$ signals, $M = 150$ IoTNs, sparsity level $K = K'5$, the degree of each signal $K_s = 5$, $T = 500$ time slots, coding rate $R = \frac{1}{3}$, iterations inside LDPC decoding $i_{LDPC} = 16$, and iterations inside CS decoding $i_{CS} = 3$, when using QPSK modulation for communication over AWGN channels.

within the LDPC decoder, we use a constant number of CS decoding iterations of $i_{CS} = 3$. As shown in Fig. 10, the BLER performance increases significantly upon increasing the number of LDPC iterations i_{LDPC} towards $i_{LDPC} = 16$, but offers diminishing returns beyond this point. This observation corresponds to that which may be drawn from the LDPC EXIT functions of Fig. 7. Motivated by these observations, In order to strike an attractive performance vs. complexity trade-off, we recommend the selection of $i_{LDPC} = 16$ and $i_{CS} = 3$ for the benchmarker employing separate LDPC-CS decoding. Using Eq. (18), we may quantify the complexity of the benchmarker as 1.6527×10^{11} ACS operations.

Fig. 11 compares the BLER performance of the benchmarker and of the proposed iterative joint LDPC-CS decoding scheme for different combinations of the iterations having different complexity, which may be compared that of our benchmarker. For example, the combination of $i_{LDPC} = 8$, $i_{CS} = 1$ and $i_{LDPC-CS} = 2$ gives a benchmarker complexity of 1.6527×10^{11} ACS operations according to Eq. (18). In this case, the performance of the proposed iterative joint decoding does not offer an improvement over the benchmarker, and suffer from an error floor, owing to the selection of $i_{CS} = 1$, as characterized in Fig. 10. However, when we increase i_{CS} to 2 in the proposed iterative joint LDPC-CS decoding scheme, the error floor is removed and we may achieve about 0.9 dB SNR gain at BLER $= 10^{-3}$ compared to the benchmarker, when adopting $i_{LDPC} = 8$ and $i_{LDPC-CS} = 2$. In this case, however, the complexity of the proposed iterative joint decoder is 2.4765×10^{11} ACS operations, which is about 1.5 times that of the benchmarker. Furthermore, if we set the number of the iterations within the LDPC and CS decoder to be the same as the benchmarker, namely to $i_{LDPC} = 16$, $i_{CS} = 3$, while setting $i_{LDPC-CS} = 2$, the complexity of the proposed iterative joint LDPC-CS decoder becomes twice that of the benchmarker, but with the benefit achieving about 1.5 dB of gain.

Fig. 12 characterizes the influence of the number of the iterations $i_{LDPC-CS}$ performed between the LDPC and CS decoder, when adopting $i_{LDPC} = 16$ and $i_{CS} = 3$. As shown in Fig. 12, the BLER performance is improved as the number of iterations $i_{LDPC-CS}$ is increased, although diminishing returns may be observed as the number of iterations increases towards $i_{LDPC-CS} = 6$. This observation corresponds to the trajectory of Fig. 8, which suggests that 6 iterations are sufficient for achieving iterative decoding convergence to a low BLER when we have SNR$= -4.4$ dB. The capacity bound in this case is about -8.75 dB based on Section II. Furthermore, as shown in Fig. 8, the EXIT tunnel opening SNR bound is around -5.2 dB.

In Fig. 13, we consider two scenarios. In the first scenario, the true sparsity level K is known to the FC. In this case, the upper bound of sparsity is $K' = K$. In the second scenario, the true sparsity level K is unknown, but the upper bound of sparsity K' is known to the FC, where $1 \leq K \leq K'$. In this case, we select the value of true sparsity K in each time slot using a uniform distribution in the range spanning from 1 to K'. Furthermore, in this case, we define the BLER as the particular fraction of blocks in which not all non-zero entries of the signal are identified, i.e. the particular fraction of blocks in which there are some false positives. As seen in Fig. 13, when the sparsity level K is lower than the upper bound K', the BLER is superior to the case of known sparsity K is to be exactly equal to the upper bound K'. The explanation for this counter-intuitive result is that while it is useful to have exact knowledge of the sparsity level, this benefit is outweighed by a stronger effect. Explicitly, having a lower value of K results in fewer non-zero entries, and hence in easier successful decoding. Fig. 13 demonstrates that as the sparsity level K is increased from 5 to 10, the BLER performance is degraded and exhibits an error floor when $M = 150$. In this case, employing more IoTNs to observe the signal and obtain more information

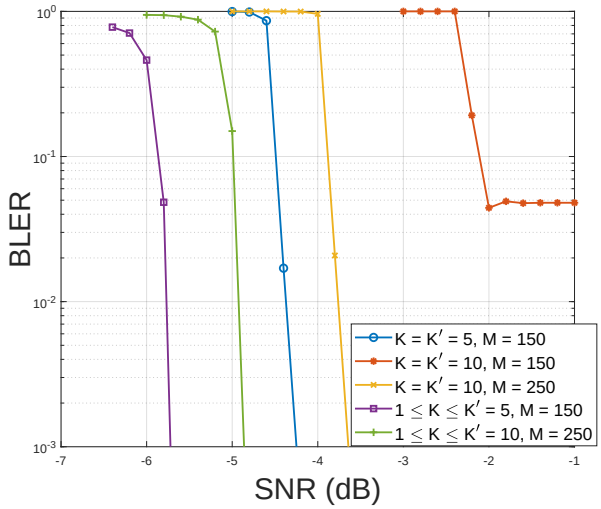

Fig. 13: BLER results of the proposed iterative joint LDPC-CS decoding scheme, when employing different sparsity K and number of IoTNs M, for the case of $N = 500$ signals, the degree of each signal $K_s = 5$, $T = 500$ time slots, coding rate $R = \frac{1}{3}$, iterations inside LDPC decoding $i_{LDPC} = 16$, iterations inside CS decoding $i_{CS} = 3$ and iterations between LDPC decoding and CS decoding $i_{LDPC-CS} = 6$, when using QPSK modulation for communication over AWGN channels.

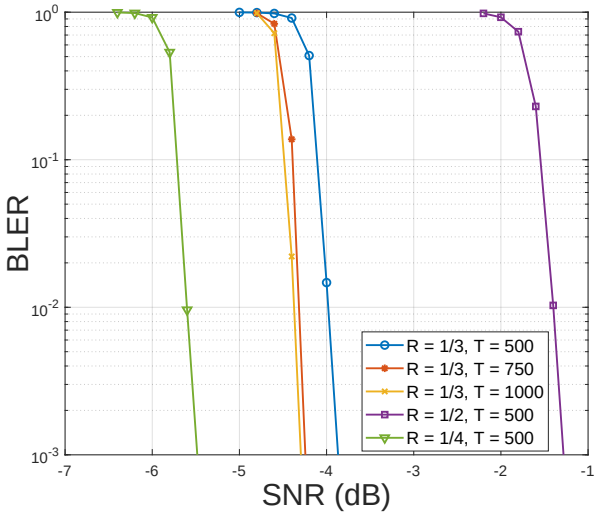

Fig. 14: BLER results of the proposed iterative joint LDPC-CS decoding, when using different block lengths T and coding rates R, for the case of $N = 500$ signals, $M = 150$ IoTNs, sparsity level $K = K' = 5$, the degree of each signal $K_s = 5$, when using QPSK modulation for communication over AWGN channels.

can help improve the decoding performance and eliminate the error floor, as shown in Fig. 13 when increasing M to 250.

Fig. 14 characterizes the BLER of the proposed iterative joint LDPC-CS decoding scheme for different block lengths and coding rates. Here, we adopt iteration numbers of $i_{LDPC} = 16$, $i_{CS} = 3$ and $i_{LDPC-CS} = 3$ in all cases. These results indicate that superior BLER performance is obtained for longer blocks at the same coding rate, with diminishing returns as the long block length is increased beyond a certain threshold. Furthermore, Fig. 14 shows that lower coding rates lead to superior BLER performance when the same block length is adopted, as it may be expected. These results also confirm the accuracy of our EXIT chart analysis in Section V-E.

VII. Conclusions

A novel joint CS and LDPC coding scheme was proposed for JSCC in WSNs, which can support the detection of massive number of signals, while using a relatively small number of IoTNs. Here, the adoption of source coding ensures energy efficient communication, while the use of channel coding protects the transmitted signals during transmission over realistic channels. More specifically, JSCC is implemented using the proposed iterative joint LDPC-CS decoding scheme, which we have shown to offer beneficial performance advantages over separate LDPC-CS decoding scheme at the cost of doubling the complexity. More specifically, the proposed iterative joint LDPC-CS decoding offers a gain about 1.5 dB at BLER of 10^{-3}, at the cost of doubling the complexity. Furthermore, we carried out the associated EXIT chart analysis for characterizing the iterative exchange of LLRs pertaining to the non-equiprobable bits encountered in the proposed iterative joint LDPC-CS decoding scheme. We demonstrated that the EXIT charts offer good performance predictions for the proposed iterative joint LDPC-CS decoding scheme.

References

[1] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006. [Online]. Available: https://doi.org/10.1109/TIT.2006.871582

[2] J. Huang and B. Soong, "Cost-aware stochastic compressive data gathering for wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1525–1533, 2019.

[3] P. Wu, F. Xiao, H. Huang, C. Sha and S. Yu, "Adaptive and Extensible Energy Supply Mechanism for UAVs-Aided Wireless-Powered Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 9201-9213, Sept. 2020.

[4] X. Li, X. Tao, and Z. Chen, "Spatio-temporal compressive sensing-based data gathering in wireless sensor networks," *IEEE Wireless Communications Letters*, vol. 7, no. 2, pp. 198–201, 2018.

[5] G. Kuldeep and Q. Zhang, "Design Prototype and Security Analysis of a Lightweight Joint Compression and Encryption Scheme for Resource-Constrained IoT Devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 65-181, 1 Jan.1, 2022.

[6] D. Lin, W. Min and J. Xu, "An Energy-Saving Routing Integrated Economic Theory With Compressive Sensing to Extend the Lifespan of WSNs," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7636-7647, Aug. 2020.

[7] S. Jafarpour, W. Xu, B. Hassibi, and R. Calderbank, "Efficient and robust compressed sensing using optimized expander graphs," *IEEE Transactions on Information Theory*, vol. 55, no. 9, pp. 4299–4308, Sep. 2009.

[8] Q. Chen, F. C. M. Lau, H. Wu and C. Chen, "Analysis and Improvement of Error-Floor Performance for JSCC Scheme Based on Double Protograph LDPC Codes," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14316-14329, Dec. 2020.

[9] S. Vembu, S. Verdu, and Y. Steinberg, "The source-channel separation theorem revisited," *IEEE Transactions on Information Theory*, vol. 41, no. 1, pp. 44–54, 1995.

[10] L. Pu, Z. Wu, A. Bilgin, M. W. Marcellin, and B. Vasic, "LDPC-based iterative joint source-channel decoding for jpeg2000," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 577–581, 2007.

[11] S. Liu, L. Wang, J. Chen, and S. Hong, "Joint component design for the JSCC system based on DP-LDPC codes," *IEEE Transactions on Communications*, pp. 1–1, 2020.

[12] Ó. Fresnedo, P. Suárez-Casal, and L. Castedo, "Transmission of spatio-temporal correlated sources over fading multiple access channels with DQLC mappings," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5604–5617, 2019.

[13] M. Fresia, F. Perez-Cruz, H. V. Poor, and S. Verdu, "Joint source and channel coding," *Signal Processing Magazine, IEEE*, vol. 27, pp. 104 – 113, 12 2010.

[14] C. Chen, L. Wang, and F. C. M. Lau, "Joint optimization of protograph LDPC code pair for joint source and channel coding," *IEEE Transactions on Communications*, vol. 66, no. 8, pp. 3255–3267, 2018.

[15] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 876–888, 2009.

[16] O. Simeone, "Source and channel coding for homogeneous sensor networks with partial cooperation," *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1113–1117, 2009.

[17] N. Deligiannis, E. Zimos, D. M. Ofrim, Y. Andreopoulos and A. Munteanu, "Distributed Joint Source-Channel Coding With Copula-Function-Based Correlation Modeling for Wireless Sensors Measuring Temperature," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4496-4507, Aug. 2015.

[18] V. Sadhu, X. Zhao and D. Pompili, "Energy-Efficient Analog Sensing for Large-Scale and High-Density Persistent Wireless Monitoring," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6778-6786, Aug. 2020.

[19] Z. Zhang, T. Jung, S. Makeig, and B. D. Rao, "Compressed sensing for energy-efficient wireless telemonitoring of noninvasive fetal ECG via block sparse bayesian learning," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 2, pp. 300–309, 2013.

[20] S. Li, L. D. Xu, and X. Wang, "Compressed sensing signal and data acquisition in wireless sensor networks and Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2177–2186, Nov. 2013.

[21] S. Feizi and M. Medard, "A power efficient sensing/communication scheme: Joint source-channel-network coding by using compressive sensing," *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2011, pp. 1048-1054.

[22] M. A. M. Izhar, A. J. Aljohani, S. X. Ng and L. Hanzo, "Distributed Joint Source Coding and Trellis Coded Modulation for Symbol-Based Markov Sources," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4031-4041, May 2018.

[23] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput," *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)* , vol., no., pp.1-565, 29 Oct. 2009.

[24] "Physical layer procedures for data (Release 15)," November 2017 [23-06-2022]. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/38_series/38.212/

[25] M. El-Hajjar and L. Hanzo, "EXIT charts for system design and analysis," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 127–153, 2014.

[26] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, 2001.

[27] S. X. Ng, O. R. Alamri, J. Kliewer and L. Hanzo, "Near-capacity turbo trellis coded modulation design based on EXIT charts and union bounds," in *in IEEE Transactions on Communications*, vol. 56, no. 12, pp. 2030-2039, December 2008.

[28] S. X. Ng, J. Wang, M. Tao, L. Yang and L. Hanzo, "Iteratively Decoded Variable Length Space-Time Coded Modulation: Code Construction and Convergence Analysis," *IEEE Transactions on Wireless Communications*, vol. 6, no. 5, pp. 1953-1963, May 2007.

[29] M. Yang, J. Wu, T. Wang, R. G. Maunder, and R. Gau, "Fusion-based cooperative support identification for compressive networked sensing," *IEEE Wireless Communications Letters*, vol. 9, no. 2, pp. 157–161,2020.

[30] J. W. Choi, B. Shim, Y. Ding, B. Rao, and D. I. Kim, "Compressed sensing for wireless communications: Useful tips and tricks," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1527–1550, 2017.

[31] W. Li and H. Dai, "Distributed detection in wireless sensor networks using a multiple access channel," *IEEE Transactions on Signal Processing*, vol. 55, no. 3, pp. 822–833, 2007.

[32] J. J. Meng, W. Yin, H. Li, E. Hossain and Z. Han, "Collaborative Spectrum Sensing from Sparse Observations in Cognitive Radio Networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 2, pp. 327-337, February 2011.

[33] L. Yu, X. Wang and J. Liu, "An Improved Rate Matching Algorithm for 3GPP LTE Turbo Code," *2011 Third International Conference on Communications and Mobile Computing,* 2011, pp. 345-348.

[34] L. Hanzo, S. X. Ng, T. Keller, W. Webb, *Quadrature Amplitude Modulation*, John Wiley & Sons, Ltd, 2004.

[35] Y. Rosmansyah, "Soft-demodulation of QPSK and 16-QAM for turbo coded WCDMA mobile communication systems," 2003, pp. 54–55.

[36] N. Bonello, S. Chen, and L. Hanzo, "Construction of regular quasi-cyclic protograph LDPC codes based on Vandermonde matrices," *IEEE Transactions on Vehicular Technology*,vol. 57, no. 4, pp. 2583-2588, July 2008.

[37] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[38] C. Tseng and J. Tarng, "Low-complexity and piecewise systematic encoding of non-full-rank QC-LDPC codes," *IEEE Communications Letters*, vol. 19, no. 6, pp. 897–900, 2015.

[39] S. A. Hashemi, C. Condo, F. Ercan, and W. J. Gross, "Memory-efficient polar decoders," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 7, no. 4, pp. 604–615, 2017.

[40] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5165–5179, 2015.

[41] S. Shao et al., "Survey of Turbo, LDPC, and Polar Decoder ASIC Implementations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2309-2333, thirdquarter 2019.

[42] J. Chen, T. Wang, J. Wu, C. Li, R. G. Maunder, S. Ng, L. Hanzo, "Factor Graphs for Support Identification in Compressive Sensing Aided Wireless Sensor Networks," *IEEE Sensors Journal*, vol . 21, no. 23, pp. 27195-27207, 2021.

[43] M. P. C. Fossorier, M. Mihaljevic and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Transactions on Communications*, vol. 47, no. 5, pp. 673-680, May 1999.

[44] M. Baldi, G. Cancellieri, A. Carassai and F. Chiaraluce, "LDPC codes based on serially concatenated multiple parity-check codes," *IEEE Communications Letters*, vol. 13, no. 2, pp. 142-144, February 2009.

[45] A. Movahed, M. C. Reed, N. Aboutorab, and S. E. Tajbakhsh, "EXIT chart analysis of turbo compressed sensing using message passing dequantization," *IEEE Transactions on Signal Processing*, vol. 64, no. 24, pp. 6600–6612, 2016.