Generalized Joint Shuffled Scheduling Decoding Algorithm for the JSCC System Based on Protograph-LDPC Codes

XU, Zhiping; WANG, Lin; HONG, Shaohua; CHEN, Guanrong

Published in:
IEEE Access

Published: 01/01/2021

Document Version:
Final Published version, also known as Publisher’s PDF, Publisher’s Final version or Version of Record

License:
CC BY

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1109/ACCESS.2021.3112740

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher's copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.
Generalized Joint Shuffled Scheduling Decoding Algorithm for the JSCC System Based on Protograph-LDPC Codes

ZHIPING XU\textsuperscript{1,2}, (Member, IEEE), LIN WANG\textsuperscript{1,2}, (Senior Member, IEEE), SHAOHUA HONG\textsuperscript{1,3}, (Member, IEEE), AND GUANRONG CHEN\textsuperscript{4}, (Life Fellow, IEEE)

\textsuperscript{1}Department of Communication Engineering, Xiamen University, Xiamen 361005, China
\textsuperscript{2}National Institute for Data Science in Health and Medicine, Xiamen University, Xiamen 361005, China
\textsuperscript{3}Shenzhen Research Institute, Xiamen University, Shenzhen 518057, China
\textsuperscript{4}Department of Electrical Engineering, City University of Hong Kong, Hong Kong

Corresponding author: Lin Wang (wanglin@xmu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61671395, in part by the Natural Science Foundation of Fujian Province under Grant 2020J01003, and in part by the Natural Science Foundation of Guangdong Province under Grant 2018A030313710.

The associate editor coordinating the review of this manuscript and approving it for publication was Zesong Fei.

ABSTRACT The joint shuffled scheduling decoding (JSSD) algorithm can reduce the decoding complexity of the joint source-channel coding system (JSCC) based on double protograph low-density parity-check (P-LDPC) codes. However, the JSSD algorithm will not work when the linking matrix between check nodes (CNs) of the source P-LDPC and variable nodes (VNs) of the channel P-LDPC is adopted in such a system, and this linking matrix has a significant influence on the system performance. In this paper, a generalized joint shuffled scheduling decoding (GIJSSD) algorithm is designed to work for the system, and the JSSD algorithm can be regarded as a special case of this algorithm. Simulations show that the proposed GIJSSD algorithm can reduce the decoding complexity with performance improvement when compared with the joint belief-propagation (JBP) algorithm.

INDEX TERMS Joint source-channel coding, P-LDPC code, shuffled decoding.

I. INTRODUCTION

Shannon’s classical separation principle is a milestone of information theory, which states that joint data compression and channel coding cannot bring gains and the source coding and channel coding can be optimized separately to keep the separate source-channel coding (SSCC) optimality [1, 2]. However, when considering finite-length communications, the residual redundancy left by the source code may not be utilized efficiently to improve the error-correcting performance at the channel decoder in the SSCC system. Therefore, the joint source-channel coding (JSCC) system has drawn increasing attention through effective utilization of the residual redundancy. The JSCC system have good error-correcting performance [2], complexity [3], [4] and transmission delay [5], and these advantages promote the JSCC system for applications in image processing [6], video transmissions [7] and so on. The JSCC system, where one low-density parity-check (LDPC) code [8] is used for source compression and one LDPC code is used for channel error correction, was proved to perform well in practical applications by utilizing the joint Tanner graph on the decoder side, named the double LDPC (D-LDPC) JSCC system [2]. As a subclass of multi-edge type LDPC codes, the protograph LDPC (P-LDPC) codes are constructed from a protograph, which is a Tanner graph with a relatively small number of nodes [9]. P-LDPC codes gain more advantages, such as simpler and faster encoding speed, when compared with LDPC codes. The capacity-approaching constructions of P-LDPC codes were discussed in [10] and the rate-compatible codes were designed for various applications [11]–[13]. The P-LDPC codes were designed over non-standard channels in [14] and introduced to bit-interleaved coded modulation [15], [16]. The JSCC system adopted P-LDPC codes as source code and channel code in [17]–[19], named the double P-LDPC (DP-LDPC) JSCC system. The DP-LDPC JSCC system can be seen as an evolution of the D-LDPC JSCC system, which gains the benefits of the...
D-LDPC JSCC system and inherits the excellent properties of P-LDPC codes.

A. RELATED WORKS AND MOTIVATION

In recent years, considerable research efforts have been devoted to investigating the DP-LDPC JSCC system. They mainly analyze two stages, one is the initial exploration stage and the other is the development stage.

In the initial exploration stage, the impact of the source statistic on the performance of the system was studied in [17]. It was observed that this system can obtain better bit error rate (BER) with lower source entropy. This observation promoted the invention of the source protograph extrinsic information transfer (PEXIT) tool, which can be used to calculate the source decoding thresholds. Through the source PEXIT (SPEXIT) tool, a matching criterion was proposed to coordinate the relation between the source entropy and the source coding rate [17].

In the development stage, the research focus was mainly on the code designs from different perspectives over different channels. The source code was optimized to obtain lower error floor [18], [28]. The channel code was redesigned to improve water-fall performance, for which the joint PEXIT (JPEXIT) tool was proposed to analyze the performance [19]. The linking matrix connecting variable nodes (VNs) of the channel code and check nodes (CNs) of the source code, named type-I linking matrix, and the linking matrix connecting CNs of the channel code and VNs of the source code, named type-II linking matrix, were studied in [20]–[22]. The type-II linking matrix was introduced in [20], and further study showed that the type-II linking matrix has a significant influence on the system performance [22]. The code pair of source code and channel code were jointly optimized in [23], [29].

Moreover, three or four components among source code, channel code, type-I linking matrix and type-II linking matrix were jointly optimized over the additive white Gaussian noise (AWGN) channel by binary phase shift keying (BPSK) modulation [25], or over the Rayleigh fading channel [26] by binary differential chaos shift keying (DCSK) modulation [27].

With more in-depth research works on the DP-LDPC JSCC system, research interests were gradually shifted to the study of low decoding complexity design, which is desirable by practical applications of this system. The works mentioned above all adopted the joint belief-propagation (JBP) algorithm, also known as the joint flooding decoding algorithm. The JBP algorithm has a high decoding complexity therefore is not suitable for low-power practical applications. The decoding complexity of LDPC codes is mainly determined by two factors: one is the number of edges in the parity-check matrix, and the other is the maximum number of iterations [34]. The first factor is determined by code design, when the structures of LDPC codes are designed and the number of edges is fixed. Thus, reducing the number of iterations provides an alternative method to reduce the decoding complexity. As is well known, scheduling decoding algorithms can efficiently reduce the number of iterations in the SSCC system. There are various scheduling methods, such as shuffled scheduling method [32]–[41], layer scheduling method [36]–[38], informed dynamic scheduling method [39], [40], and so on, but they were all designed for the SSCC system, not suitable for the JSCC system.

Recently, a joint shuffled scheduling decoding (JSSD) algorithm was proposed for the DP-LDPC JSCC system, which was the first work to apply the scheduling decoding algorithm to this system [41]. However, the JSSD algorithm only considered three components including source code, channel code and type-I linking matrix, while the type-II linking matrix was not taken into account. The JSSD algorithm without considering type-II linking matrix cannot take full advantage of the complete structure of the DP-LDPC JSCC system, since the influence of type-II linking matrix on the performance of the DP-LDPC JSCC system is not negligible [20], [22]. Therefore, it is beneficial for the DP-LDPC JSCC system to have a decoding algorithm to support the shuffled mode with a complete structure including all four components in the system.

B. CONTRIBUTIONS

In this paper, a generalized joint shuffled scheduling decoding (GJSSD) algorithm is designed for the low decoding complexity applications of the DP-LDPC JSCC system. The GJSSD algorithm has a complete structure including all four components and the JSSD algorithm can be regarded as a special case of the GJSSD algorithm when the type-II linking matrix is zero-matrix. The simulation results show that the GJSSD algorithm can obtain nearly two-fifths reduction in decoding complexity and achieve performance improvement as compared with the JBP algorithm. Furthermore, the GJSSD algorithm can obtain a lower error floor than the JSSD algorithm.

The rest of the paper is organized as follows. Section II briefly introduces the DP-LDPC JSCC system model. The proposed GJSSD algorithm is presented in Section III. The simulation results are shown in Section VI. The last section concludes the paper.
II. DP-LDPC JSCC SYSTEM MODEL
For the DP-LDPC JSCC system, a joint base matrix $B_J$ of size $(m_{sc} + m_{cc}) \times (n_{sc} + n_{cc})$ is defined by
\[
B_J = \begin{bmatrix}
B_{sc} & B_{L1} \\
B_{L2} & B_{cc}
\end{bmatrix},
\]
where $B_{sc}$ of size $m_{sc} \times n_{sc}$ represents the source protograph base matrix, $B_{cc}$ of size $m_{cc} \times n_{cc}$ represents the channel protograph base matrix, $B_{L1}$ of size $m_{sc} \times n_{cc}$ represents the connecting edges between the variable nodes (VNs) in the channel protograph and check nodes (CNs) in the source protograph, and $B_{L2}$ of size $m_{cc} \times n_{sc}$ represents the connecting edges between CNs in the channel protograph and VNs in the source protograph, respectively. $B_{L1}$ and $B_{L2}$ are also called type-I and type-II linking base matrices, respectively. The corresponding joint parity-check matrix $H_J$ of size $(M_{sc} + M_{cc}) \times (N_{sc} + N_{cc})$ is obtained by the progressive edge growth (PEG) algorithm [31], as
\[
H_J = \begin{bmatrix}
H_{sc} & H_{L1} \\
H_{L2} & H_{cc}
\end{bmatrix},
\]
where $H_{sc}$ of size $M_{sc} \times N_{sc}$ is the source P-LDPC code, $H_{cc}$ of size $M_{cc} \times N_{cc}$ is the channel P-LDPC code, $H_{L1}$ of size $M_{sc} \times N_{cc}$ represents the linking relationship between CNs in the source P-LDPC code and VNs in the channel P-LDPC codes, and $H_{L2}$ of size $M_{cc} \times N_{sc}$ represents the linking connections between VNs in the source P-LDPC code and CNs in the channel P-LDPC code. $H_{L1}$ and $H_{L2}$ are called type-I and type-II linking matrixes, respectively.

Fig. 1 and Fig. 2 show the system model and the joint Tanner graph, respectively. The point-to-point communication system is considered. At the encoder of the DP-LDPC JSCC system, the sequence of information bits, $s = (s_1, s_2, \ldots, s_t)$, $s_t \in \{0, 1\}$, is generated from a binary independent and identically distributed (i.i.d.) Bernoulli source. The source entropy is calculated by
\[
H(s) = -p \log_2(p)-(1-p)\log_2(1-p),
\]
where $p = \Pr(s_t = 1)$ is the probability of the “1” of the source and $p \neq 0.5$.

Next, the source sequence $s$ is compressed by
\[
b = s \cdot H_{sc}^T.
\]

Then, part of source sequence $s$ and compressed sequence $b$ are combined as a new sequence $[s', b]$, and the codeword sequence $c$ is generated by
\[
c = [s', b] \cdot G_{cc}^{\text{new}},
\]
where $G_{cc}^{\text{new}}$ is the generator matrix of a new parity-check matrix $H_{cc}^{\text{new}} = [H_{L2} \ H_{cc}]$. To that end, the sequence of encoded bits is transmitted over an AWGN channel by BPSK modulation as sequence $x$. At the receiver, the joint source channel decoder (JSCD), which is shown in Fig. 2, is used to recover the sequence of the corrupted information bits $y$ by JBP decoding algorithm. The JSCD works in parallel and exchanges the messages between these two decoders along the edges between the channel decoder and the source decoder.

![Figure 2. Joint Tanner graph of the DP-LDPC JSCC system.](image)

III. THE GENERALIZED JOINT SHUFFLED SCHEDULING DECODING ALGORITHM
Consider the $i$-th iteration of the joint decoder. Assume that the codeword sequence $c$ is transmitted over an AWGN channel with zero mean and variance $\sigma^2$. The maximum number of the decoding iterations is set to $I_{\text{max}}$. Denote the set of all CNs connected to the $n$-th variable node (VN) by $M(n)$. Similarly, denote the set of all VNs connected to the $m$-th check node (CN) by $N(m)$. Moreover, denote $M(n)/m$ as the set $M(n)$ with the $m$-th CN excluded, and $N(m)/n$ as the set $N(m)$ with the $n$-th VN excluded. Also, use $F_{cc}^{(n)}$ $(n = 1, \ldots, N_{cc})$ to represent the channel log-likelihood ratio (LLR) of the $n$-th VN in the channel decoder and $F_{cc}^{(n)}$ $(n = N_{cc} + 1, \ldots, N_{cc} + N_{sc})$ to represent the source LLR of the $n$-th VN in the source decoder.

For illustration, eight types of log-likelihood ratios (LLRs) are defined as follows, which are shown in Fig. 2.

- $\ell_{mn}(0)$ represents the LLR sent from the $m$-th CN to the $n$-th VN and $\ell_{mn}(1)$ represents the LLR sent from the $n$-th VN to the $m$-th CN in the channel decoder.
- $\ell_{mn}(0)$ represents the LLR sent from the $m$-th CN to the $n$-th VN and $\ell_{mn}(1)$ represents the LLR sent from the $n$-th VN to the $m$-th CN in the source decoder.
- $\ell_{cc}(i)$ is the LLR sent from the corresponding CN in the source decoder to the $n$-th VN in channel decoder, and $\ell_{cc}(i)$ is the LLR sent from the $n$-th VN in the channel decoder to the corresponding CN in the source decoder. These two types of LLRs are indexed only by $n$ because each CN in the source decoder is connected to only one single VN in the channel decoder.
- $\ell_{mn}(0)$ is the LLR sent from the $n$-th VN in the source decoder to the $m$-th CN in the channel decoder, and $\ell_{mn}(1)$ is the LLR sent from the $m$-th CN in the channel decoder to the $n$-th VN in the source decoder.

Based on the above definitions, the generalized joint shuffled scheduling decoding algorithm is described as follows.

**Initialization:**
For all $m$ and $n$, set $\ell_{mn}^{(0)} = 0$, $\ell_{mn}^{(1)} = 0$, $\ell_{sc}^{(0)} = 0$, $\ell_{cc}^{(0)} = 0$. For $n = 1, \ldots, N_{cc}$, set $\ell_{mn}^{(1)} = F_{cc}^{(n)} = 2y_n/\sigma^2$, where
\[
y_n = (1 - 2x_n) + G_{n} \quad \text{and} \quad G_{n} \sim N(0, \sigma_n^2).
\]
For \( n = N_{cc} + 1, \ldots, N_{cc} + N_{sc}, \) set \( zsc_{mn}^{(i)} = F_{sc}^{n} = \ln((1 - p))/p). \) 
Set \( i = 1. \)

**Step 1:** LLRs updating.

**Channel decoder:**
1) For \( 1 \leq n \leq N_{cc}, \) and for each \( m \in M(n), \) process the next two steps jointly.

1.1) Horizontal Step: Compute \( \varphi_{mn}^{cc,(i)} \) using Eq. (6), as shown at the bottom of the page.

1.2) Vertical Step: When \( 1 \leq n \leq N_{cc} - M_{cc}, \) compute
\[
\varphi_{mn}^{cc,(i)} = F_{cc}^{n} + \sum_{m' \in M(n)/m} \varphi_{m'n}^{cc,(i)}.
\]  
When \( N_{cc} - M_{cc} + 1 \leq n \leq N_{cc}, \) compute
\[
\varphi_{mn}^{cc,(i)} = F_{cc}^{n} + \varphi_{m'n}^{cc,(i-1)} + \sum_{m' \in M(n)/m} \varphi_{m'n}^{cc,(i)}.
\]  
2) For \( N_{cc} - M_{cc} + 1 \leq n \leq N_{cc} \) and each \( m \in M(n), \) compute
\[
\varphi_{m'n}^{cc,(i)} = F_{cc}^{n} + \sum_{m \in M(n)} \varphi_{mn}^{cc,(i)}. \]  

3) For \( 1 \leq n \leq 1 + N_{cc} + M_{cc}, \) compute \( \varphi_{mn}^{cc,(i)} \) using
\[
\tanh \left( \frac{\varphi_{mn}^{cc,(i-1)}}{2} \right) = \prod_{n' \in N(m)} \tanh \left( \frac{\varphi_{m'n'}^{cc,(i-1)}}{2} \right) \times \prod_{1 < n' \leq N_{cc}} \tanh \left( \frac{\varphi_{m'n'}^{cc,(i-1)}}{2} \right). \]  

**Source decoder:**
1) For \( N_{cc} + 1 \leq n \leq N_{cc} + N_{sc} \) and for each \( m \in M(n), \) process the next two steps jointly.

1.1) Horizontal Step: Compute \( \varphi_{mn}^{sc,(i)} \) using Eq. (11), as shown at the bottom of the page.

1.2) Vertical Step: When \( 1 + N_{cc} \leq n \leq N_{cc} + M_{cc}, \) compute
\[
\varphi_{mn}^{sc,(i)} = F_{sc}^{n} + \sum_{m' \in M(n)} \varphi_{m'n}^{sc,(i)} + \sum_{m' \in M(n)/m} \varphi_{m'n}^{sc,(i)}. \]  
When \( N_{cc} + M_{cc} + 1 \leq n \leq N_{cc} + N_{sc}, \) compute
\[
\varphi_{mn}^{sc,(i)} = F_{sc}^{n} + \sum_{m' \in M(n)/m} \varphi_{m'n}^{sc,(i)}. \]  

2) For \( 1 + N_{cc} \leq n \leq N_{sc} + N_{cc} \) and for each \( m \in M(n), \) compute \( \varphi_{mn}^{sc,(i)} \) using
\[
\tanh \left( \frac{\varphi_{mn}^{sc,(i)}}{2} \right) = \prod_{n' \in N(m)} \tanh \left( \frac{\varphi_{m'n'}^{sc,(i-1)}}{2} \right) \times \prod_{n < n'} \tanh \left( \frac{\varphi_{m'n'}^{sc,(i-1)}}{2} \right). \]  

3) For \( N_{cc} + 1 \leq n \leq N_{cc} + N_{sc} \) and for each \( m \in M(n), \) compute
\[
\varphi_{m'n}^{sc,(i)} = F_{sc}^{n} + \sum_{m' \in M(n)} \varphi_{m'n}^{sc,(i)} + \sum_{m' \in M(n)/m} \varphi_{m'n}^{sc,(i)}. \]  

**Step 2:** Hard decision.
Compute the a posteriori LLRs \( \varphi_{i}^{(i)} \) of the VNs in the channel decoder using (16) for \( n = 1, \ldots, N_{cc} - M_{sc} \) and (17) for \( n = N_{cc} - M_{sc} + 1, \ldots, N_{cc}. \) Set \( \hat{\varphi}_{i} = \varphi_{i}^{(i)} \) where \( \hat{\varphi}_{i} = 0 \) if \( \varphi_{i}^{(i)} \geq 0, \) and \( \hat{\varphi}_{i} = 1 \) otherwise, and
\[
\varphi_{i}^{sc,(i)} = F_{sc}^{n} + \sum_{m' \in M(n)} \varphi_{m'n}^{sc,(i)} + \sum_{m' \in M(n)/m} \varphi_{m'n}^{sc,(i)}. \]  

Compute the a posteriori LLRs \( \varphi_{i}^{sc,(i)} \) of the VNs in the source decoder using (18) for \( n = N_{cc} + 1, \ldots, N_{cc} + M_{cc} \) and (19) for \( n = N_{cc} + M_{cc} + 1, \ldots, N_{cc} + N_{sc}. \) Set \( \hat{\varphi}_{i} = \varphi_{i}^{(i)} \) where \( \hat{\varphi}_{i} = 0 \) if \( \varphi_{i}^{(i)} \geq 0, \) and \( \hat{\varphi}_{i} = 1 \) otherwise, and
\[
\varphi_{i}^{sc,(i)} = F_{sc}^{n} + \sum_{m' \in M(n)} \varphi_{m'n}^{sc,(i)} + \sum_{m' \in M(n)/m} \varphi_{m'n}^{sc,(i)}. \]  

**Step 3:** Stopping condition.
If \( H_{sc} \hat{s} = b \) and \( H_{c} \hat{c} = 0 \) are both satisfied, or \( i = I_{max} \), the iteration will be stopped and go to Step 4. If the conditions are not met, set \( i = i + 1 \) and go to Step 1.

**Step 4:** Output \( \hat{s} \) as the decoded source sequence.

The GJSSD algorithm contains four steps: initialization, LLR-updating process, hard decision, stopping criterion and outputting the decoded source sequence. The major difference between the GJSSD algorithm and the JBP algorithm is the LLR-updating process, but the other three steps are the same [41]. The LLR-updating process in the GJSSD algorithm is sequential, while the LLR-updating process in the JBP algorithm is parallel. The JSSD algorithm can be
regarded as a special case of the GJSSD algorithm with \( B_{L2} = 0 \). Different from the JSSD algorithm, the GJSSD algorithm takes the influences of \( B_{L2} \) into full consideration. The followings are the major differences between the GJSSD algorithm and the JSSD algorithm.

In the initialization process of the GJSSD algorithm, the values \( \ell_{mn}^{cc\rightarrow cc,(0)} \) and the values \( \ell_{mn}^{cc\rightarrow sc,(0)} \), associated with \( B_{L2} \), are initialized as 0 and \( F_{B}^{cc} \), respectively. But, these are not included in the JSSD algorithm. In the LLR-updating process of the GJSSD algorithm, the values \( \ell_{mn}^{cc\rightarrow cc,(t-1)} \) and the values \( \ell_{mn}^{cc\rightarrow sc,(t-1)} \) are involved in the computation of the values \( \ell_{mn}^{cc,(t)} \) in the channel decoder and the values \( \ell_{mn}^{sc,(t-1)} \) in the source decoder, respectively (Eq. (6) and Eq. (12)). Furthermore, the values \( \ell_{mn}^{cc\rightarrow sc,(t)} \) and the values \( \ell_{mn}^{sc\rightarrow cc,(t)} \) are computed in the channel decoder and source decoder, respectively (Eq. (10) and Eq. (15)). However, these calculations are not involved in the JSSD algorithm. With the help of the complete structure including \( B_{sc} \), \( B_{cc} \), \( B_{L1} \) and \( B_{L2} \) in \( B_{J} \), the GJSSD algorithm can make full use of the structure of the DP-LDPC JSCC system to achieve better performance, compared with the JSSD algorithm.

### IV. SIMULATION RESULTS

In this section, simulation results are reported in two parts: performance analysis of the GJSSD algorithm and performance comparison between the GJSSD algorithm and the JBP algorithm. The simulations are all realized with C++ language in Visual Studio 2013. In all simulations, the length of information bits generated by the source is set as 3200 and the parity-check matrix is obtained from the protograph with a lifted factor of 800 by the PEG algorithm. The maximum iteration number \( I_{\text{max}} \) is 30. In the simulation results, the R4JA code with rate-1/4 and rate-1/2 are used as source code, and AR3A code with rate-1/2 is used as channel code, respectively. Their corresponding joint base matrices are given as follows. \( B_{L1} \) and \( B_{L2} \) are contained in \( B_{J} \). Without loss of generality, choose \( a_{ij} = 1 \) (i = 1, 3, 4; j = 5) and the other \( a_{ij} = 0 \) when \( B_{L2} \neq 0 \) in all simulations. The label \((p_{value}, r, L, D_{name})\) in the simulation figures means that the length of source bits is \( L \), with \( R_{total} = r \), which is tested under the decoding method of \( D_{name} \) to decode when \( p = p_{value} \). For example, the label \((0.02, 2, 3200, \text{GJSSD})\) means that the source bits with length 3200 and \( R_{total} = 2 \) are decoded under the decoding method of the GJSSD algorithm, when \( p = 0.02 \). Here, the method used in [41] can be seen as the GJSSD algorithm with \( B_{L2} = 0 \).

\[
B_{J4} = \begin{bmatrix}
3 & 1 & 3 & 1 & 3 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 2 & 1 & 3 & 1 & 3 & 1 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 1 \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & 0 & 0 & 0 & 0 & 1 & 2 & 1 1
\end{bmatrix}
\]

(20)

#### A. PERFORMANCE ANALYSIS OF THE GJSSD ALGORITHM

Fig. 3 shows the bit error rate (BER) performance of the GJSSD algorithm under different conditions, revealing the impacts of the total code rate \( R_{total} \), the length of the source bits and the source entropies. When \( p = 0.02 \), the GJSSD algorithm with \( R_{total} = 1 \) and \( L = 3200 \) achieves coding gains of 1.6 dB and 2.5 dB at \( BER = 1 \times 10^{-6} \), respectively, compared to \( R_{total} = 2 \) with \( L = 12800 \) and \( L = 3200 \). When \( R_{total} = 2 \) and \( L = 3200 \), the GJSSD algorithm with \( p = 0.01 \) has coding gains of 1.7 dB and 4.7 dB at \( BER = 1 \times 10^{-6} \), compared to \( p = 0.02 \) and \( p = 0.03 \), respectively. When \( R_{total} = 2 \) and \( p = 0.03 \), the GJSSD algorithm with \( L = 12800 \) obtains 3.3 dB coding gain at \( BER = 1 \times 10^{-6} \), compared with \( L = 3200 \).

Fig. 4 shows the average number of iterations of the GJSSD algorithm under different conditions. When \( R_{total} = 2 \) and \( L = 3200 \), the GJSSD algorithm with \( p = 0.01 \) converges at lowest \( E_b/N_0 \), followed by \( p = 0.02 \), and \( p = 0.03 \), yielding the highest at the same \( BER \) level. When \( R_{total} = 2 \) and \( p = 0.03 \), the GJSSD algorithm with \( L = 3200 \) and \( L = 12800 \) nearly have the same average number of iterations.
the source decoding threshold, hence the algorithm with cannot. It is because the source statistic is larger than \( B_0 \) with \( B_0 \) of the GJSSD with water-fall region in the moderate-to-low SNR region.

The average number of iterations of the GJSSD algorithm when \( B_{L2} \neq 0 \) (solid line) and \( B_{L2} = 0 \) (dash line), compared to the GJSSD algorithm with \( B_0 \) cannot work. When \( p = 0.03 \), the GJSSD with \( B_{L2} \neq 0 \) can achieve lower error floor than the algorithm with \( B_{L2} = 0 \), with a negligible performance loss of the water-fall region in the moderate-to-low SNR region.

In Fig. 6, one can observe that the average iteration number of the GJSSD with \( B_{L2} \neq 0 \) is the same as the GJSSD with \( B_{L2} = 0 \) when \( p = 0.01 \). When \( p = 0.02 \), the GJSSD with \( B_{L2} \neq 0 \) has a slightly larger iteration numbers than the GJSSD with \( B_{L2} = 0 \). When \( p = 0.03 \), the GJSSD with \( B_{L2} = 0 \) cannot work and oscillation occurs.

Fig. 7 shows the BER performance of the GJSSD algorithm in the DP-LDPC JSCC system and the shuffled decoding algorithm in the SSCC system. One can easily see the superiority of the GJSSD algorithm. For example, the GJSSD algorithm can achieve 2.4 dB coding gain at BER = 2 \times 10^{-6}, compared with the shuffled decoding algorithm in the separated mode.

These simulation results also demonstrate the rationality and validity of the GJSSD algorithm, summarized as follows:

1) When it comes to the error correction performance, the factors in the descending order of degree of influence are \( R_{total} \), source entropies, and the code length. A smaller \( R_{total} \), a lower source entropy, and a longer code length, altogether yields a better BER performance.

2) The convergence speed has a relation with \( R_{total} \) and source entropy, but not the code length. With the same \( E_b/N_0 \), the smaller of the \( R_{total} \) or the source entropy, the faster of the convergence speed.

3) The GJSSD algorithm with \( B_{L2} \neq 0 \) can achieve lower error floor with a little loss of the water-fall region in the moderate-to-low region and a little larger of the iteration number, compared to the GJSSD algorithm with \( B_{L2} = 0 \).

B. PERFORMANCE COMPARISON BETWEEN GJSSD ALGORITHM AND JBP ALGORITHM

Fig. 8 and Fig. 9 show the average number of iterations at different \( E_b/N_0 \) for joint decoder and separated decoder, respectively. In Fig. 8, the GJSSD algorithm can achieve faster convergence than JBP when \( p = 0.01 \), \( p = 0.02 \) and \( p = 0.03 \), with \( B_{L2} \neq 0 \) and \( B_{L2} = 0 \). For example, in Fig. 8(a), the average number of iterations is reduced from 13.88 to 8.68 at \( E_b/N_0 =-1 \) dB when \( p = 0.01 \), from 12.73 to 7.99 at \( E_b/N_0 =1 \) dB when \( p = 0.02 \), and from 9.91 to 6.10 at \( E_b/N_0 =4.5 \) dB when \( p = 0.03 \), with \( B_{L2} \neq 0 \). Details of the convergence speeds of the JBP and the GJSSD algorithm, when \( p = 0.01 \), \( p = 0.02 \) and \( p = 0.03 \), can be found in Table 1, Table 2 and Table 3. In these three Tables, the ratio \( \gamma \) is defined as \( \gamma = \frac{T_{GJSSD}}{T_{JBP}} \), where \( T_{GJSSD} \) and \( T_{JBP} \) denote the iterative numbers of the GJSSD algorithm and.
FIGURE 8. The average numbers of iterations for different types of decoding algorithms under different conditions. The solid line represents the GJSSD algorithm and the dash line represents the JBP algorithm. (a) $B_{L2} \neq 0$; (b) $B_{L2} = 0$.

The JBP algorithm at convergence, respectively. The GJSSD algorithm converges faster than the JBP algorithm if $\gamma < 1$. Moreover, the smaller the value of $\gamma$, the faster the converged speed of the GJSSD algorithm. From the tables, one can find that the ratios of the iteration numbers of GJSSD to JBP at convergence are all nearly 0.6, where $\gamma$ is nearly 0.6, and for SSCC system, $\gamma$ is 0.5 [30].

Fig. 9 shows the unequal convergence rates (UCR) phenomenon [41] in the DP-LDPC JSCC system. One can find that the GJSSD restrains the UCR to a certain extent and there still exist slight UCR in some regions with a high signal-noise ratio (SNR), while the UCR in the JBP algorithm with $B_{L2} \neq 0$ and $B_{L2} = 0$ are obvious.

Fig. 10 shows the comparison of BER performances between the GJSSD algorithm and the JBP algorithm under different conditions. From Fig. 10(a), one can find that the GJSSD algorithm still outperforms the JBP ($B_{L2} \neq 0$) by nearly 0.4 dB at $1 \times 10^{-5}$ BER level when $p = 0.03$, 0.25 dB at $2 \times 10^{-6}$ BER level when $p = 0.02$, 0.2 dB at $1 \times 10^{-6}$ BER level when $p = 0.01$. In Fig. 10(b), one can also find that the GJSSD algorithm obtains coding gains of 0.2 dB at $BER = 1 \times 10^{-6}$ when $p = 0.01$ and achieves lower error.

FIGURE 9. The average numbers of iterations for source decoder (solid line) and channel decoder (dash line) with different types of decoding algorithms under different conditions.

FIGURE 10. Comparison of BER performances between GJSSD algorithm and JBP algorithm under different conditions. The solid line represents the GJSSD algorithm and the dash line represents the JBP algorithm. (a) $B_{L2} \neq 0$; (b) $B_{L2} = 0$. 
The convergence speed comparison of JBP ($B_{12} \neq 0$) and GJSSD by simulations when $p = 0.01$.

<table>
<thead>
<tr>
<th>$E_b/N_0 (dB)$</th>
<th>-2.5</th>
<th>-2.0</th>
<th>-1.5</th>
<th>-1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{JBP}$</td>
<td>23.31</td>
<td>19.50</td>
<td>16.32</td>
<td>13.88</td>
</tr>
<tr>
<td>$T_{GJSSD}$</td>
<td>14.94</td>
<td>12.00</td>
<td>9.98</td>
<td>8.68</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.64</td>
<td>0.62</td>
<td>0.61</td>
<td>0.63</td>
</tr>
</tbody>
</table>

The convergence speed comparison of JBP ($B_{12} \neq 0$) and GJSSD by simulations when $p = 0.02$.

<table>
<thead>
<tr>
<th>$E_b/N_0 (dB)$</th>
<th>0</th>
<th>0.5</th>
<th>1</th>
<th>1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{JBP}$</td>
<td>19.25</td>
<td>15.16</td>
<td>12.73</td>
<td>12.34</td>
</tr>
<tr>
<td>$T_{GJSSD}$</td>
<td>11.39</td>
<td>9.32</td>
<td>7.99</td>
<td>7.77</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.59</td>
<td>0.61</td>
<td>0.63</td>
<td>0.63</td>
</tr>
</tbody>
</table>

The convergence speed comparison of JBP ($B_{12} \neq 0$) and GJSSD by simulations when $p = 0.03$.

<table>
<thead>
<tr>
<th>$E_b/N_0 (dB)$</th>
<th>2.5</th>
<th>3</th>
<th>3.5</th>
<th>4</th>
<th>4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{JBP}$</td>
<td>13.36</td>
<td>12.11</td>
<td>11.13</td>
<td>10.45</td>
<td>9.91</td>
</tr>
<tr>
<td>$T_{GJSSD}$</td>
<td>8.09</td>
<td>7.33</td>
<td>6.79</td>
<td>6.39</td>
<td>6.10</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.62</td>
</tr>
</tbody>
</table>

floor when $p = 0.02$, compared with the JBP algorithm. Compared to the algorithms with $B_{12} \neq 0$, one can see that the GJSSD algorithm can lower the error floor level, and GJSSD can also obtain lower error floor than the JBP with $B_{12} = 0$.

In summary, compared to the JBP algorithm, the GJSSD algorithm can reduce the decoding iteration number by nearly 40% for convergence, and achieve better waterfall performance and lower error floor.

V. CONCLUSION

This paper designs a generalized joint shuffled scheduling decoding (GJSSD) algorithm to fully exploit the complete structure of the DP-LDPC JSCC system. The proposed algorithm’s influential factors, including the code length, the code rate and the source entropy, are comprehensively investigated. Through simulations, it is found that the GJSSD algorithm can reduce decoding complexity by nearly 40 percent in the cases with a moderate-to-high signal-to-noise ratio (SNR) compared to the JBP algorithm, and achieve lower error floor than the JSSD algorithm. In the future, the performance of the proposed decoding algorithm will be studied for specific applications, for example over in-body or on-body channels.

REFERENCES


ZHIPPING XU (Member, IEEE) received the B.S. degree in electronic and information engineering from Jimei University, Xiamen, China, in 2012, and the M.S. degree in integrated circuit engineering from Fuzhou University, Fuzhou, China, in 2017. He is currently pursuing the Ph.D. degree with the Department of Information and Communication Engineering, Xiamen University, Xiamen. His research interests include channel coding, joint source and channel coding/decoding, chaotic coded modulation, and the Internet of Things.