Received April 20, 2020, accepted May 4, 2020, date of publication May 7, 2020, date of current version May 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992933

Computing Algorithms for LDPC Coded Internet-of-Things

SHANCHENG ZHAO
College of Information Science and Technology, Jinan University, Guangzhou 510632, China
e-mail: zhaoday@mail2.sysu.edu.cn

This work was supported in part by the NSF of China under Grant 61871201, Grant 61772234, and Grant 61501206, and in part by the Provincial NSF of Guangdong Province under Grant 2017A030313371.

ABSTRACT Low-density parity-check (LDPC) codes are widely employed in communication systems. We focus on the computing of messages at the sink node of internet-of-things (IoT). As opposed to decoding all the messages, we consider the case that the sink node is interested in computing a linear transformation of the messages. We assume that all the IoT devices are identical. We first present three representations of the considered systems, based on which three multistage computing algorithms are proposed, which are decoding-computing (DC) algorithm, computing-decoding (CD) algorithm, and computing-decoding-computing (CDC) algorithm. Secondly, we show that the considered system admits a compact normal graph representation, based on which a joint computing algorithm is proposed. Thirdly, we present numerical results to show the advantages of the proposed algorithms. Numerical results show that the optimality of the proposed algorithms depends on the channel conditions and the computing functions. Numerical results also show that the joint computing algorithm has the best performances for a variety of scenarios. Finally, we present a simulation-based optimization procedure to design finite-length LDPC codes for the joint computing algorithm.

INDEX TERMS LDPC codes, linear superposition, joint computing algorithm, internet-of-things.

I. INTRODUCTION

Internet-of-things (IoT) has recently emerged as a promising enabling technique for a wide class of applications [1]–[7]. For IoT-aided sensing networks, the sink node may not be interested in the original messages. For example, the sink node may be interested in the modulo-sum of the original messages. This requires improved design of the IoT transmission schemes. Particularly, cross-layer design may be required. The classical example of computing channel is the two-way relay channel with physical-layer network coding [8].

As a class of capacity-approaching codes, low-density parity-check (LDPC) codes have been widely adopted in emerging wireless communication systems to achieve reliable information transmission [9]–[11]. Since the power and the computing capability of the nodes in IoT are limited, new codes and modulations should be developed [1], [6], [12]. Furthermore, IoT may be used to support various applications with differing latency and reliability requirements. These requirements pose further challenges on the implementation of IoT.

Superposition is a common phenomenon in communication systems. Superposition is introduced artificially in modulation to achieve the channel capacity [13]. In power domain non-orthogonal multiple access, superimposed signals are generated deliberately to improve the spectral efficiency [14]. In IoT, signals of different sensing nodes are superimposed at the sink node. The classical channel model which characterizes this phenomenon is the Gaussian interference channels (GIFC). In GIFC, the sink node is only interested in the messages of its intended transmitter. In modulo-sum computing [15]–[18], the sink node is interested in the modulo-sum of the messages. It can be seen that, instead of decoding all the messages, the receiver is interested in decoding a linear transformation of the messages. In [18], the authors analyzed the distance spectrum of coded Gaussian two-way relay channels with binary input. In [16], the authors developed a framework to optimize finite-length LDPC codes for two-way relay channels. In [19], the authors presented a class of

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

invertible-subset LDPC codes with improved error correction capacity for orthogonal frequency division multiplexing systems. In [20], decoding algorithms for LDPC coded GIFC were presented. The optimization of LDPC codes for GIFC was considered in [21]. To the best knowledge of the authors, very few works in the literature focused on the design of *computing algorithms* for LDPC coded IoT networks.

This paper focuses on the design of computing algorithms for IoT which employs LDPC codes for transmission. We assume that all IoT devices are the same, which means that they employ the same binary LDPC code in the physical layer. We first present three representations of the considered systems, based on which three multistage computing algorithms are proposed, which are decoding-computing (DC) algorithm, computing-decoding (CD) algorithm, and computing-decoding-computing (CDC) algorithm. Secondly, we show that the considered system admits a compact normal graph representation, based on which a joint computing algorithm is proposed. Thirdly, we consider the applications of the proposed algorithms to Gaussian interference channels (GIFC) and modulo-sum computing. Numerical results show that the optimality of the proposed algorithms depends on the channel conditions and the computing functions. Numerical results also show that the joint computing algorithm reveals the best performances for a variety of scenarios. Finally, we present a simulation-based optimization procedure to design finite-length LDPC codes for the joint computing algorithm. Numerical results show that for the considered system performance improvements are obtained by optimizing the LDPC codes.

The rest of this paper is organized as follows. In Section II, the system model is presented. In Section III, we present the normal graphs and the corresponding iterative multistage computing algorithms. Section IV presents the joint computing algorithm. Applications of the proposed algorithms are presented in Section V and the finite-length design of LDPC codes is given in Section VI. Section VII concludes this paper.

II. SYSTEM DESCRIPTION AND PROBLEM STATEMENT

A. BINARY LDPC CODES

A binary LDPC code $\mathcal{C}[n, k]$ is defined by a sparse binary parity-check matrix $\mathbf{H}$. The length of the code is n and the dimension of the code is k. There are two different ways to specify an LDPC code ensemble, the edge degree distribution perspective and the node degree distribution perspective. For simplicity, the node degree distribution is used in this paper. We use Λ_i to denote the fraction of degree-i variable nodes and Γ_j to denote the fraction of degree-j check nodes. To write these in a compact form, we have

$$\Lambda(x) = \sum_{i=1}^{N} \Lambda_i x^i \ \text{ and } \ \Gamma(x) = \sum_{i=1}^{M} \Gamma_i x^i.$$

In this paper, $\Lambda(x)$ is referred to as the variable-node degree distribution and $\Gamma(x)$ is referred to as the check-node degree

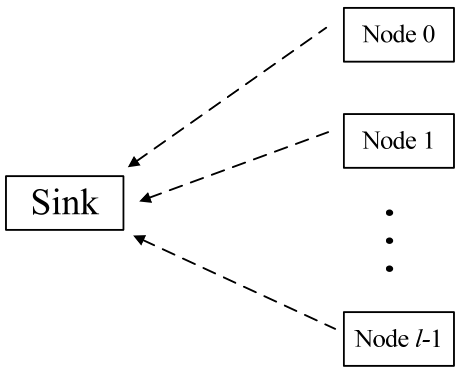


FIGURE 1. The considered system with l IoT sensing nodes and a sink node.

distribution. Obviously, we have $\sum_{i=1}^{N} \Lambda_i = 1.0$, $\sum_{i=1}^{M} \Gamma_i = 1.0$, $\Lambda_i \geq 0$ $(1 \leq i \leq N)$, and $\Gamma_j \geq 0$ $(1 \leq j \leq M)$. The variable-node degree distribution can be viewed as an element of an $(N-1)$-dimension manifold in the N-dimension space and hence can be represented by an $(N-1)$-dimensional vector $\underline{\Lambda} = (\Lambda_1, \Lambda_2, \cdots, \Lambda_{N-1})$.

B. THE LDPC CODED LINEAR SUPERPOSITION SYSTEM

Assume that there are l IoT devices who want to communicate with the sink node, see Fig. 1 for reference. The l IoT devices are assumed to be identical. Particularly, these l IoT devices employ the same LDPC codes for error correction and the same modulation for transmission. The i-th device wants to transmit the sequence $\underline{u}^{(i)}$ of length k to the IoT sink node. To achieve reliable transmission, the sequence $\underline{u}^{(i)}$ is encoded by the LDPC code before transmission. The encoding output is denoted as $\underline{c}^{(i)} = (c_0^{(i)}, c_1^{(i)}, \cdots, c_{n-1}^{(i)})$. For modulation, the binary phase shift keying (BPSK) is used. For the i-th codeword, we use $\underline{x}^{(i)}$ to denote the i-th modulated sequence for transmission. At the sink node, the received signal is $\underline{y} = (y_0, y_1, \cdots, y_{n-1})$, where

$$y_t = \sum_{0 \leq i \leq l-1} h_i x_t^{(i)} + z_t = \sum_{0 \leq i \leq l-1} h_i \alpha_t^{(i)} (1 - 2c_t^{(i)}) + z_t. \quad (1)$$

The channel coefficients are $h_0, h_1, \cdots, h_{l-1}$. The z_t is the mean zero Gaussian noise with variance $\sigma^2 = N_0/2$. For simplicity, we assume that the channel coefficients h_i's are time-invariant. However, we allow the the amplitudes $\alpha_t^{(i)}$'s to be time-varying. We impose the following power constraint on $P^{(i)}$, which represents the power constraint of the i-th IoT device.

$$\frac{1}{n} \sum_{0 \leq t \leq n-1} \left(\alpha_t^{(i)}\right)^2 \leq P^{(i)}. \quad (2)$$

For simplicity, we set $\sigma^2 = 1$.

Obviously, we have $\underline{c}^{(i)} \mathbf{H}^T = \underline{0}$, for $0 \leq i \leq l-1$. Let $c_t = (c_t^{(0)}, c_t^{(1)}, \cdots, c_t^{(l-1)})^T$, which is composed of the coded bits at time t. Based on c_t's, we can define the following matrix.

$$\underline{c} = (c_0, c_1, \cdots, c_{n-1}) = \begin{pmatrix} c_0^{(0)} & c_1^{(0)} & \cdots & c_{n-1}^{(0)} \\ c_0^{(1)} & c_1^{(1)} & \cdots & c_{n-1}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ c_0^{(l-1)} & c_1^{(l-1)} & \cdots & c_{n-1}^{(l-1)} \end{pmatrix}. \quad (3)$$

It can be easily checked that $\underline{c}\mathbf{H}^T = (c_0, c_1, \cdots, c_{n-1})\mathbf{H}^T = \mathbf{0}$. Let τ denote a linear transformation from $\mathbb{F}_2^l$ to $\mathbb{F}_2^{l'}$. Define $\underline{v} = \tau(\underline{c}) \overset{\Delta}{=} (\tau(c_0), \tau(c_1), \cdots, \tau(c_{n-1}))$. In this paper, we consider the following general problem.

How to compute $\underline{v} = \tau(\underline{c})$ at the IoT sink node from the received sequence $\underline{y}$?

Note that we may not interested in recovering all the message sequences. However, if this is the case, we have $\tau = \mathbf{I}$. It can be seen that the above problem is very general. We present in the following several approaches to achieve this goal.

- The straightforward solution is to decode $\underline{c}$ first and then compute $\tau(\underline{c})$. Algorithms follow this procedure will be referred to as *decoding-computing algorithms*. To decode $\underline{c}$, we require the likelihoods of its components, which are computed as

$$f_o(y_t|c_t) = \frac{1}{\sqrt{2\pi}} \exp(-(y_t - \phi_t(c_t))^2/2), \quad c_t \in \mathbb{F}_2^l, \quad (4)$$

  where $\phi_t : \mathbb{F}_2^l \mapsto \mathbb{R}$ is determined by

$$\phi_t(c_t) = \sum_{0 \le i \le l-1} h_i \alpha_t^{(i)} (1 - 2c_t^{(i)}),$$

  for $c_t \in \mathbb{F}_2^l$ and $0 \le t \le n-1$. The problem of the decoding-computing algorithm is that when the channel is bad we cannot achieve the reliably recovery of $\underline{c}$.
- The second solution first computes the likelihoods and then decodes. Particularly, we first compute the likelihoods for $v_t = \tau(c_t)$ as

$$f_\tau(y_t|v_t) \propto \sum_{\substack{c_t \in \mathbb{F}_2^l \\ \tau(c_t) = v_t}} f_o(y_t|c_t), \quad v_t \in \mathbb{F}_2^{l'} \quad (5)$$

  for $0 \le t \le n-1$. Then, these likelihoods are used to initialize the computing algorithms for an l'-level LDPC coded system. For simplicity, we will refer this algorithm as *computing-decoding algorithm*.
- The last solution is called the computing-decoding-computing solution. In this solution, we first recover a linear transform of $\underline{c}$ by the linear mapping $\tilde{\tau}$, then compute $\tau(\underline{c})$ from the decoding results. To ensure correct computing, we require that the linear mapping $\tilde{\tau}$ is invertible. Different from the decoding-computing algorithm, the computing-decoding-computing algorithms are initialized with the following likelihoods

$$f_{\tilde{\tau}}(y_t|w_t) = f_o(y_t|\tilde{\tau}^{-1}(w_t)), \quad w_t \in \mathbb{F}_2^l \quad (6)$$

  for $0 \le t \le n-1$. Interestingly, our simulation results indicate that the computing-decoding-computing algorithm has the best performances in certain scenarios.

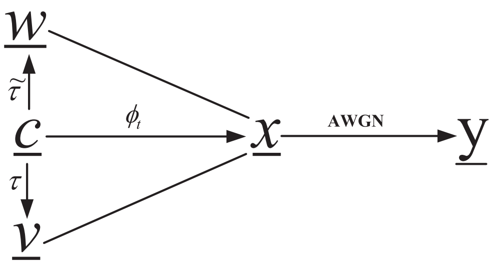


FIGURE 2. The relationships among the involved variables.

To make it more clear, we have illustrated the relationships of all the involved variables in Fig. 2. The decoding-computing algorithms follow the route $\underline{y} \to \underline{x} \to \underline{c} \to \underline{v}$, the computing-decoding algorithms follow the route $\underline{y} \to \underline{x} \to \underline{v}$, whereas the computing-decoding-computing algorithms follow the route $\underline{y} \to \underline{x} \to \underline{w} \to \underline{c} \to \underline{v}$.

III. ITERATIVE MULTISTAGE COMPUTING ALGORITHMS

In this section, several iterative multistage computing algorithms are presented to recover $\tau(\underline{c})$. The LDPC code $\mathcal{C}[n, k]$ specified by the parity-check matrix $\mathbf{H}$ is used. A high-level normal graph for the considered system is shown in Fig. 3 (a). In a normal graph, edges are used to indicates the random variable and nodes are used to indicate local constraints on the random variables. For example, the $\ominus$ nodes in Fig. 3 represent the equality constraint. Following the convention, we use probability mass function (pmf) to denote the message of a random variable. For example, for a random variable Z, its associated message is represented by the real vector P_Z. If a vector of messages is needed, we use the notation $P_{\underline{Z}}(\underline{z})$, where $\underline{Z}$ is a random vector. To simplify the description of the algorithm, we use $P_{\underline{Z}}^{\mathcal{A} \to \mathcal{B}}(\underline{z})$ to denote the message from node $\mathcal{A}$ to node $\mathcal{B}$, where $\underline{Z}$ represents the random vector on the edge from $\mathcal{A}$ to $\mathcal{B}$.

In the following, we present three multistage computing algorithms, which are decoding-computing (DC) algorithm, computing-decoding (CD) algorithm, and the computing-decoding-computing (CDC) algorithm. These algorithms depends on different normal graphical representations of the considered system, which are presented in Fig. 3. Fig. 3 (a) corresponds to the most straightforward representation. The DC algorithm works over the normal graph shown in Fig. 3 (a). Here, we omit the details of the DC algorithm since it is highly similar to the CD algorithm in Algorithm 1. We point out the the DC algorithm closely follows the well-known iterative multistage decoding algorithm. Note that the DC algorithm attempts to recover $\tau(\underline{c})$ by first recovering $\underline{c}$ and then applying the linear mapping τ. This algorithm has been shown in the literature to be ineffective for coded two-way relay channels.

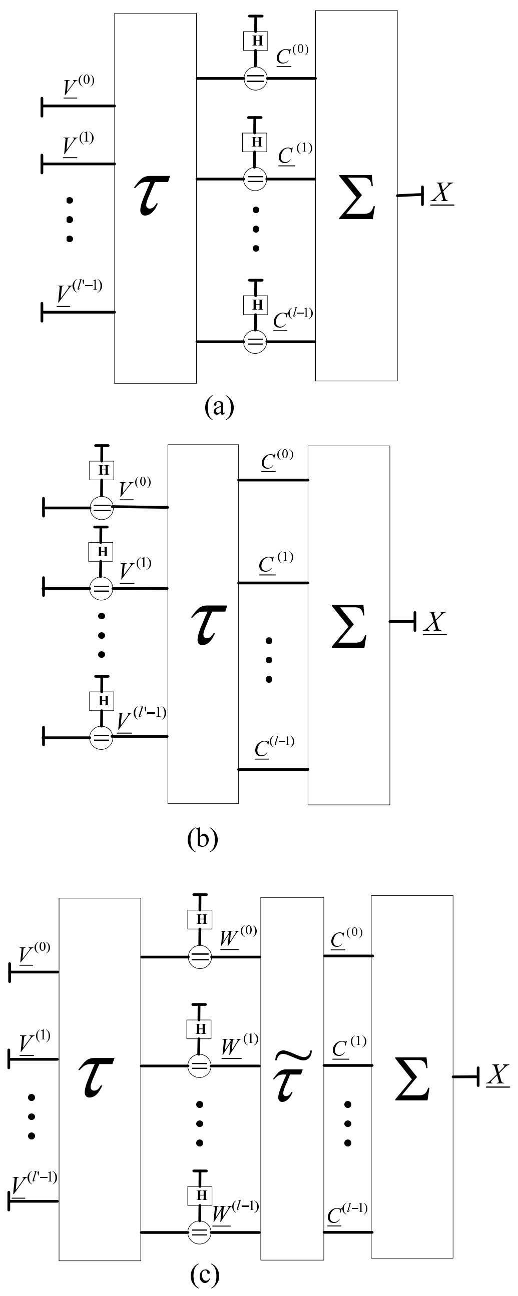


FIGURE 3. **Normal graphical representations for different computing algorithms: (a) decoding-computing algorithm, (b) computing-decoding algorithm, and (c) computing-decoding-computing algorithm.**

A. THE CD ALGORITHM

We derive the basic principle underlying the CD algorithm in this subsection. Note that τ is a linear mapping, hence, we have $\underline{v}^{(i)}\mathbf{H}^T = \underline{0}$, where $\underline{v}^{(i)}$ is composed of the i-th components of $v_0, v_1, \cdots, v_{n-1}$. With this result, we know that the binary vector $\underline{v}^{(i)}$ is a codeword of the LDPC code. As a result, we may first compute the likelihood associated with v_t, then use the iterative decoder to decode the message v_t. We present in Fig. 3 (b) another normal graphs of the considered system. The CD algorithm works by exchanging messages over this normal graph. The procedure of CD algorithm closely follows the procedure of DC algorithm, except the initialization. For completeness, we present in the following the detailed procedure of the CD algorithm. During the decoding, the global iteration is executed with maximum I_{max} iterations and the LDPC decoder is executed with maximum K_{max} iterations.

Algorithm 1 The CD Algorithm

```
• Initialization: Set $P^{\mathbf{H}\to\Sigma}_{\underline{V}^{(i)}}(\underline{v}^{(i)} = 0) = P^{\mathbf{H}\to\Sigma}_{\underline{V}^{(i)}}(\underline{v}^{(i)} = 1) = 0.5$ $0 \le i \le l' - 1$..
• Computing: Compute the likelihoods $f_\tau(y_t|v_t)$ according to (5).
• Decoding: While $K < K_{max}$
    1) Compute $P^{\Sigma\to\mathbf{H}}_{\underline{V}^{(i)}}(\underline{v}^{(i)})$ as $P^{\Sigma\to\mathbf{H}}_{V_t^{(i)}}(m) \propto \sum_{v_t\in\mathbb{F}_2^l,\, v_t^{(i)}=m} f_\tau(y_t|v_t)\prod_{j\neq i} P^{\mathbf{H}\to\Sigma}_{V_t^{(j)}}(v_t^{(j)})$, for $m \in \mathbb{F}_2$.
    2) Execute the LDPC decoder to compute $P^{\mathbf{H}\to\Sigma}_{\underline{V}^{(i)}}(\underline{v}^{(i)})$. The maximum iteration of the LDPC decoder is $I_{max}$, which is pre-specified.
    3) The messages $P_{\underline{V}^{(i)}}(\underline{v}^{(i)})$ for decision is computed as
           $$P_{V_t^{(i)}}(v_t^{(i)}) \propto P^{\mathbf{H}\to\Sigma}_{V_t^{(i)}}(v_t^{(i)})\, P^{\Sigma\to\mathbf{H}}_{V_t^{(i)}}(v_t^{(i)}),$$
       and the decision is made as
           $$\hat{v}_t^{(i)} = \arg\max_{v_t^{(i)}\in\mathbb{F}_2} P_{V_t^{(i)}}(v_t^{(i)}).$$
    4) If $\hat{\underline{v}}$ is a codeword, decoding stops and a success is delivered; otherwise set $K = K + 1$.
• Output: Output the decoding results.
```

Note that if the linear mapping τ is determined by the $l \times l$ identity matrix, the CD algorithm and the DC algorithm are the same. For the CD algorithm, we first map the l length-n message sequences into l' length-n message sequences. Note that the decoding complexity of CD algorithm is lower than that of the DC algorithm, since l' is smaller than l. Furthermore, since DC algorithm may not work in low SNR region, we expect the CD algorithm to perform better than the DC algorithm.

B. THE CDC ALGORITHM

In the subsection, we present the computing method based on computing-decoding-computing. Let $\tilde{\tau}$ denote an invertible linear transform. In the CDC algorithm, the linear transform $\tilde{\tau}(\underline{c})$ is recovered first and then the intended function $\tau(\tilde{\tau}^{-1}\tilde{\tau}(\underline{c}))$ is recovered. We require that after applying the linear transform $\tilde{\tau}$ we are able to find the intended message from $\tilde{\tau}(\underline{c})$. The invertible functions satisfying this requirements depend on the properties of τ. It is difficult to give a general description of which $\tilde{\tau}$ satisfies the requirement. The first candidate of $\tilde{\tau}$ is the identity mapping, with which the CDC algorithm degenerates to the DC algorithm. Based on

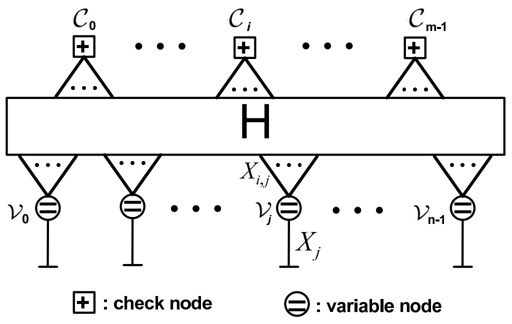


FIGURE 4. **A joint normal graphical realization of linear LDPC coded superposition system.**

$\tilde{\tau}$, we can define the following matrix

$$\underline{w} = (w_0, w_1, \cdots, w_{n-1}) \triangleq \tilde{\tau}(\underline{c}) = \begin{pmatrix} \underline{w}^{(0)} \\ \underline{w}^{(1)} \\ \vdots \\ \underline{w}^{(l-1)} \end{pmatrix}.$$

It can be easily checked that each row of the above matrix is a codeword of the LDPC code. This motivates us to present the CDC algorithm. We present in Fig. 3 (c) the normal graph by viewing $\underline{w}^{(t)}$ as a codeword of the LDPC code. The CDC algorithm is implemented over this normal graph by exchanging messages.

The decoding procedure of the CDC algorithm is similar to that of the CD algorithm. For space limitation, we omitted the details of the decoding procedure of the CDC algorithm. Note that the decoding complexity of the CDC algorithm depends on the invertible linear transform $\tilde{\tau}$. To achieve low-complexity, we should design the $\tilde{\tau}$ appropriately. Other than complexity, other important aspects should also be considered. For example, for a given channel condition, how to determine the optimal linear map $\tilde{\tau}$ to achieve the best performance. This problem is more involved and is not considered here. Furthermore, we should jointly design the local iteration number and the global iteration number to achieve the trade-offs between performance and complexity. We point out that we may treat the CD algorithm and the CDC algorithm as message pre-processing techniques. Hence, they can be used in the joint computing algorithm in Section IV.

IV. JOINT COMPUTING ALGORITHM

Recall that $\underline{c} = (c_0, c_1, \cdots, c_{n-1})$. As the c_t is a length-l binary vector, it can be viewed as a finite filed element of $\mathbb{F}_q$, where $q = 2^l$. Hence, we have $\underline{c}\mathbf{H}^T = (c_0, c_1, \cdots, c_{n-1})\mathbf{H}^T = \underline{0}$, which means that $\underline{c}$ is a codeword the code specified by the parity-check matrix $\mathbf{H}$. This observation motivates us to represent the considered system with the normal graph given in Fig. 4. It can be seen that this normal graph is completely different from those given in Section III. To decode, we operate the iterative message processing algorithm over this normal graph. For completeness, we present in the following the details of the joint algorithm.

The most important problem of the joint algorithm is that its decoding complexity grows exponentially with the number of IoT devices. Hence, the joint algorithm is applicable only for IoT systems with small l. On the other hand, the complexities of the iterative multistage algorithms in Section III grow linearly with l. Hence, they can be used for IoT systems with large l. Note that the computing-decoding-computing technique can also be incorporated into the joint algorithm to reduce the decoding complexity.

Algorithm 2 The Joint Algorithm

```
• Initialization: The original likelihoods are used to initialize the decoder as $P_{V_t}^{|\to \mathcal{V}_t}(x) \propto f_o(y_t|x)$. The maximum number of iteration is selected as $K_{max}$.
• Iteration: For $K = 1, 2, \cdots, K_{max}$
    1) Compute $P_{X_{i,j}}^{\mathcal{C}_i \to \mathcal{V}_j}(x)$ as
       $$P_{X_{i,j}}^{\mathcal{C}_i \to \mathcal{V}_j}(x) = \sum_{x+\sum_{k\neq j} x_{i,k}=0} (\prod_{k\neq j} P_{X_{i,k}}^{\mathcal{V}_k \to \mathcal{C}_i}(x_{i,k})), \quad (7)$$
       for $x \in \mathbb{F}_q$.
    2) Compute $P_{X_{i,j}}^{\mathcal{V}_j \to \mathcal{C}_i}(x)$ as
       $$P_{X_{i,j}}^{\mathcal{V}_j \to \mathcal{C}_i}(x) \propto P_{V_j}^{|\to \mathcal{V}_j}(x) \prod_{k\neq i} P_{X_{k,j}}^{\mathcal{C}_k \to \mathcal{V}_j}(x), \quad (8)$$
       for $x \in \mathbb{F}_q$.
    3) Detections:
       – Compute
         $$P_{X_j}(x_j) \propto P_{X_j}^{|\to \mathcal{V}_j}(x_j) \prod P_{X_{k,j}}^{\mathcal{C}_k \to \mathcal{V}_j}(x_j), \quad (9)$$
         for $x_j \in \mathbb{F}_q$.
       – Compute the intended message $\underline{V} = \tau(\underline{C})$
         $$P_{V_j}(v_j) = \sum_{\substack{x_j \in \mathbb{F}_2^l \\ \tau(x_j)=v_j}} P_{X_j}(x_j) \quad (10)$$
         for $v_j \in \mathbb{F}_2^{l'}$.
       – The decisions are made as
         $$\hat{v}_j = \arg \max_{v_j \in \mathbb{F}_{2^{l'}}} P_{V_j}(v_j). \quad (11)$$
       – If $\hat{\underline{v}}$ is a codeword, decoding stops and a success is delivered.
    4) Increment $K$ by one.
• Output: Output the decoding results.
```

V. NUMERICAL RESULTS

Obviously, the linear transformation τ, the code, and the channel shall influence the performances of the proposed algorithms. In this section, we present numerical results to show the performances of different algorithms in computing different functions. Particularly, we consider the GIFC and modulo-sum computing. In our simulations, the additive Gaussian noise channel is assumed. The modulation is assumed to be the BPSK. The LDPC codes is decoded with the sum-product algorithm.

A. GAUSSIAN INTERFERENCE CHANNELS

In this subsection, we investigate the applications of the proposed algorithms in Gaussian interference channels. Consider the two-user symmetric Gaussian interference channels. We assume that a (3,6)-regular LDPC code constructed by the PEG algorithm [22] is employed. The girth of this code is 6. Due to the symmetry of the two users, we only consider the first pair of users. The first receiver serves as the sink node and the two transmitters serve as the sensing nodes. For fairness, we impose the constraint that $P^{(0)} = P^{(1)} = P/2$. The linear transformation computed by the sink node is $\tau = [1\ 0]$. The DC algorithm, CD algorithm, CDC algorithm, and the joint computing algorithm are employed for decoding. For CDC algorithm, the invertible mapping $\tilde{\tau}$ is specified by the matrix

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

It can be seen that we can obtain $\tau(\underline{c})$ once $\tilde{\tau}(\underline{c})$ is correctly recovered. Moderate interference and strong interference are considered in our simulations, which are $(h_0^2 = 1, h_1^2 = 0.5)$ and $(h_0^2 = 1, h_1^2 = 0.75)$. We present in Fig. 5 and Fig. 6 the simulation results. It can be seen from these curves that

1) For $h_1^2 = 0.5$, at BER $= 10^{-4}$, the CDC algorithm has the best error performance, as in CDC algorithm the influence of the interference is mitigated in the computing procedure. In other algorithms, the influence of interference is not reduced.
2) For $h_1^2 = 0.75$, the best performance is achieved by the joint algorithm. Particularly, at BER $= 10^{-4}$, at least 0.4 dB performance gain is obtained.

B. MODULO-SUM COMPUTING

Consider an IoT network with two sensing nodes and a sink node. The messages of the sensing nodes are coded with a rate 1/3 LDPC code $\mathcal{C}_{\text{Kite}}[12288, 4096]$ [23]. The sink node is interested in computing the modulo-2 sum of the messages. That is, the sink node wants to compute $\underline{c}^{(0)} \oplus \underline{c}^{(1)}$ and the corresponding linear transform is defined by $\tau = [1\ 1]$. This case appears when the sink node serves as the point for information exchanging among the IoT devices. Following existing works, we assume that $h_0^0 = h_1^2 = 1.0$. Similarly, to achieve fairness, we impose that $P^{(0)} = P^{(1)} = P/2$. The joint computing algorithm and the CD algorithm are implemented for computing. We present in Fig. 7 the bit error rates of these two algorithms. In our simulations, both the CD algorithm and the joint computing algorithm are implemented with $K_{max} = 200$. It can be seen that the joint algorithm performs about 0.2 dB better than the CD algorithm at BER $= 10^{-4}$.

We further consider an IoT network with three sensing nodes and a sink node. Similarly, the messages of the sensing nodes are coded with the rate 1/3 LDPC code $\mathcal{C}_{\text{Kite}}[12288, 4096]$ and the sink node is interested in computing the modulo-2 sum $\underline{c}^{(0)} \oplus \underline{c}^{(1)} \oplus \underline{c}^{(2)}$. Similarly, we assume that $h_0^0 = h_1^2 = h_2^2 = 1.0$. For fairness, we assume that $P^{(0)} = P^{(1)} = P^{(2)} = P/3$. The joint computing algorithm and the CD algorithm are implemented for computing. The simulation results are given in Fig. 8. In our simulations, both the CD algorithm and the joint computing algorithm are implemented with $K_{max} = 200$. It can be seen that the joint computing algorithm performs about 0.1 dB better than

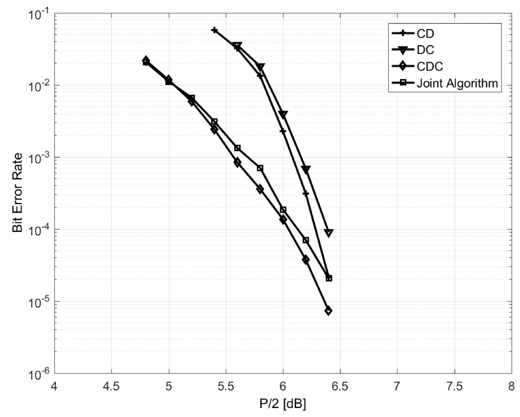


FIGURE 5. Bit error rates of different algorithms in GIFC with moderate interference $h_1^2 = 0.5$.

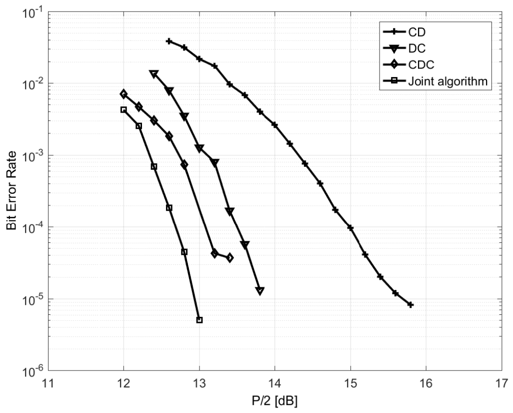


FIGURE 6. Bit error rates of different algorithms in GIFC with strong interference $h_1^2 = 0.75$.

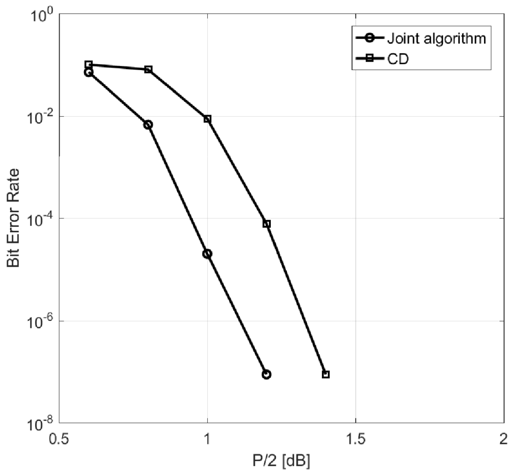


FIGURE 7. Bit error rates of the different algorithms for computing the modulo-sum.

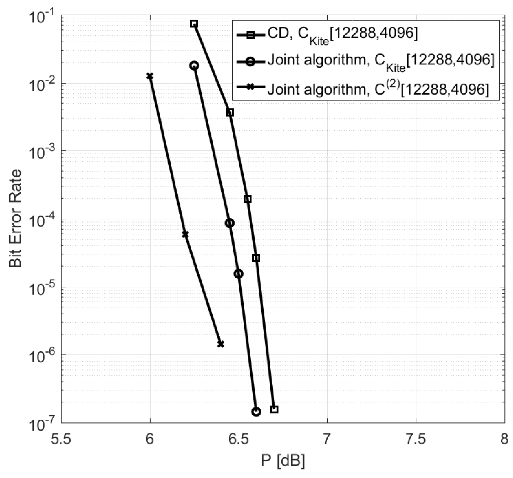


FIGURE 8. Bit error rates of the different computing algorithms and different LDPC codes.

the CD algorithm at BER $= 10^{-4}$. Note that the decoding complexity of the joint computing algorithm is higher than that of the CD algorithm.

VI. OPTIMIZATION OF FINITE-LENGTH LDPC CODES FOR THE JOINT ALGORITHM

It can be seen from the simulation results in Section V that for a variety of scenarios the best performance is achieved by the joint algorithm. This implies the wide applicability of the joint algorithm. In IoT with mission-critical applications, the latency is of great importance. For these applications, we should employ short-length LDPC codes. On one hand, it can be seen that when computing is considered, the channel model and the target of the considered system are different from the point-to-point links. A natural question is how about the performances of the codes optimized for point-to-point links in the proposed system. On the other hand, LDPC codes are typically optimized for long block length. When the length is short the optimization results may be misleading. Motivated by the above observations, we dedicated to the design of finite-length LDPC codes for the joint algorithm in this section. Obviously, the density evolution algorithm cannot be used since finite-length is our focus. Instead, a hybrid optimization algorithm based on simulation and the downhill-simplex algorithm is used. We noticed a similar optimization algorithm in [24]. The difference lies in the fact that we have different target. The basic idea of the proposed optimization algorithm is presented in the following.

In the proposed optimization procedure, d valid node degree distributions are generated at first. For each node degree distribution, we construct an LDPC code with the progressive edge-growth (PEG) algorithm [24] and then evaluate its frame error rate (FER) with the joint algorithm. During the iterations, we generate a new node degree distribution $\underline{\Lambda}^{t}$ in each iteration. The new generated $\underline{\Lambda}^{t}$ is then taken as the input of the PEG algorithm to generate a new LDPC code. If this new code performs better than the second worst code, the new generated distribution $\underline{\Lambda}^{t}$ is taken for consideration and the worst one is discarded. When the difference between the d codes is small, we terminate the iteration and output the code with the best FER as the optimization result. We omit the details for its simplicity.

1) OPTIMIZATION FOR GAUSSIAN INTERFERENCE CHANNELS

In this example, we execute the proposed optimization procedure to optimize finite-length LDPC codes for GIFC. Particularly, we consider the case with strong interference, that is we have $h_1^2 = 0.75$. We select the following initial parameters for our optimization.

- The code length is $n = 4000$.
- The code rate is $r = 0.5$.
- The maximum node degree is $N = 6$.
- The number of candidate codes is $d = 8$.
- The target SNR is 10.25 dB.

After executing the optimization procedure, the optimized result is

$$\Lambda^{(1)}(x) = 0.695x^2 + 0.085x^3 + 0.025x^4 + 0.006x^5 + 0.189x^6.$$

Based on $\Lambda^{(1)}(x)$, we construct two LDPC codes with the PEG algorithm [22]. The first code is a length-4000 code $\mathcal{C}^{(1)}[4000, 2000]$ and the second code is a length-10000 code $\mathcal{C}^{(1)}[10000, 5000]$. Both codes have rate 0.5 and girth six. We present in Fig. 9 the performances of these two codes in GIFC with strong interference. For comparison, the bit error rates of two binary LDPC codes $\mathcal{C}^{(0)}[4000, 2000]$ and $\mathcal{C}^{(0)}[10000, 5000]$ are also presented. These two codes are constructed with the degree distribution $\Lambda^{(0)}(x) = 0.5072x^2 + 0.2506x^3 + 0.0840x^4 + 0.1582x^6$ [25], which is optimized for AWGN channels. It can be observed that about 0.4 dB performance improvement is obtained with our optimization. The bit error rates of a (3,6)-regular LDPC code $\mathcal{C}_{\text{reg}}[10000, 5000]$ is also included in Fig. 9 for comparison. The comparison shows that we can obtain about 2.1 dB performance improvement with the proposed optimization.

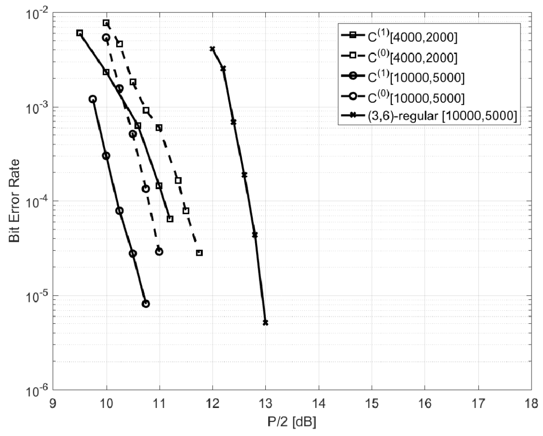


FIGURE 9. Bit error rates of optimized LDPC codes in GIFC with $h_1^2 = 0.75$ under the joint algorithm.

2) OPTIMIZATION FOR MODULO-SUM COMPUTING

We optimize the LDPC node degree distribution for modulo-sum computing. We select the following initial parameters for our optimization.

- The code length is $n = 4500$.
- The code rate is $r = 1/3$.
- The maximum node degree is $N = 6$.
- The number of code is $d = 8$.
- The target SNR is 6.05 dB.

After executing the optimization procedure, the best code obtained is

$$\Lambda^{(2)}(x) = 0.6823x^2 + 0.1932x^3 + 0.0078x^4 \\ +0.0103x^5 + 0.1064x^6.$$

Based on $\Lambda^{(2)}(x)$, we use the PEG algorithm to construct an LDPC code $\mathcal{C}^{(2)}[12288, 4096]$ with girth six. The error performances of this code are shown in Fig. 8. It can be seen that, at BER $= 10^{-5}$, the optimized code $\mathcal{C}^{(2)}[12288, 4096]$ performs about 0.25 dB better than the Kite code $\mathcal{C}_{\text{Kite}}[12288, 4096]$.

VII. CONCLUSION

In this paper, the design of computing algorithms at the sink node of LDPC coded IoT was considered. We focused on the case that all IoT devices are identical. We first presented three multistage computing algorithms based on three normal graphical representations. We then present a compact representation of the considered system and based on which we propose a joint algorithm. Applications of these algorithms in computing different linear functions are considered. The numerical results suggested that the optimality of the computing algorithms depends on channel conditions and the computing functions. We also considered the optimization of finite-length LDPC codes for the joint algorithm.

REFERENCES

[1] L. Da Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[3] M. Y. Metwly, M. S. Abdel-Majeed, A. S. Abdel-Khalik, M. Torki, R. A. Hamdy, M. S. Hamad, and S. Ahmed, "IoT-based supervisory control of an asymmetrical nine-phase integrated on-board EV battery charger," *IEEE Access*, vol. 8, pp. 62619–62631, 2020.

[4] R. Zhou, X. Zhang, X. Du, X. Wang, G. Yang, and M. Guizani, "File-centric multi-key aggregate keyword searchable encryption for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3648–3658, Aug. 2018.

[5] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial Internet of Things (IIoT) systems: A deep reinforcement learning approach," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3559–3570, Jun. 2019.

[6] S. Zhao, J. Wen, S. Mumtaz, S. Garg, and B. J. Choi, "Spatially coupled codes via partial and recursive superposition for industrial IoT with high trustworthiness," *IEEE Trans. Ind. Informat.*, early access, Jan. 13, 2020, doi: 10.1109/TII.2020.2965952.

[7] D. Zhang, L. Zheng, Q. Chen, B. Wei, and X. Ma, "A power allocation-based overlapping transmission scheme in Internet of vehicles," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 50–59, Feb. 2019.

[8] S. Zhang and S.-C. Liew, "Channel coding and decoding in a relay system operated with physical-layer network coding," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 5, pp. 788–796, Jun. 2009.

[9] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: MIT Press, 1962.

[10] S. Zhao and X. Ma, "Extended construction of array-based non-binary LDPC codes," *Electron. Lett.*, vol. 55, no. 4, pp. 196–198, Feb. 2019.

[11] Y. Zhao, Y. Fang, and Z. Yang, "Interleaver design for small-coupling-length spatially coupled protograph LDPC-coded BICM systems over wireless fading channels," *IEEE Access*, vol. 8, pp. 33500–33510, 2020.

[12] Q. Zhang, L. T. Yang, Z. Chen, P. Li, and F. Bu, "An adaptive dropout deep computation model for industrial IoT big data learning with crowdsourcing to cloud computing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2330–2337, Apr. 2019.

[13] L. Duan, B. Rimoldi, and R. Urbanke, "Approaching the AWGN channel capacity without active shaping," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 1997, p. 374.

[14] M. Alhusseini, P. Azmi, and N. Mokari, "Optimal joint subcarrier and power allocation for MISO-NOMA satellite networks," *Phys. Commun.*, vol. 32, pp. 50–61, Feb. 2019.

[15] K. Xu, Z. Lv, Y. Xu, D. Zhang, X. Zhong, and W. Liang, "Joint physical network coding and LDPC decoding for two way wireless relaying," *Phys. Commun.*, vol. 6, pp. 43–47, Mar. 2013.

[16] A. K. Tanc, T. M. Duman, and C. Tepedelenlioglu, "Design of LDPC codes for two-way relay systems with physical-layer network coding," *IEEE Commun. Lett.*, vol. 17, no. 12, pp. 2356–2359, Dec. 2013.

[17] D. Wubben and Y. Lang, "Generalized sum-product algorithm for joint channel decoding and physical-layer network coding in two-way relay systems," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2010, pp. 1–5.

[18] T. Yang, I. Land, T. Huang, J. Yuan, and Z. Chen, "Distance spectrum and performance of channel-coded physical-layer network coding for binary-input Gaussian two-way relay channels," *IEEE Trans. Commun.*, vol. 60, no. 6, pp. 1499–1510, Jun. 2012.

[19] T. ValizadehAslani and A. Falahati, "An analysis and improvement of error control performance of IS-LDPC codes with a large number of subsets," *Phys. Commun.*, vol. 31, pp. 79–86, Dec. 2018.

[20] S. Zhao, X. Ma, and B. Bai, "Decoding algorithms for binary LDPC coded Gaussian interference channels," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2013, pp. 39–43.

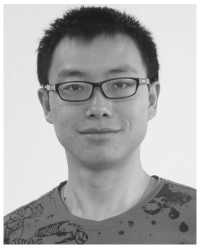
[21] S. Zhao, X. Ma, and B. Bai, "Design of LDPC codes for binary-input Gaussian interference channels," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Oct. 2014, pp. 70–74.

[22] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

[23] K. Zhang, X. Ma, S. Zhao, B. Bai, and X. Zhang, "A new ensemble of rate-compatible LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2012, pp. 2536–2540.

[24] X. Hu, "Low-delay low-complexity error-correcting codes based on sparse graphs," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 2002.

[25] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.



SHANCHENG ZHAO received the bachelor's degree in software engineering and the Ph.D. degree in information and communication engineering from Sun Yat-sen University, in 2009 and 2014, respectively. From 2013 to 2014, he was a Graduate Visiting Student with the University of California at Los Angeles (UCLA), Los Angeles, CA, USA. He is currently an Associate Professor with the College of Information Science and Technology, Jinan University, Guangzhou, China. His current research interests include spatially coupled codes and nonbinary LDPC codes. He was a co-recipient of the Best Paper Award of 2015 IEEE GlobeCom.