A Branch-Cut-and-Price Algorithm for Optimal Decoding in Digital Communication Systems

Banu Kabakulak*[1], Z. Caner Taşkın[2], and Ali Emre Pusane[3]

[1]Department of Industrial Engineering, İstanbul Bilgi University, İstanbul, Turkey

[2]Department of Industrial Engineering, Boğaziçi University, İstanbul, Turkey

[3]Department of Electrical and Electronics Engineering, Boğaziçi University, İstanbul, Turkey

Channel coding aims to minimize the errors that occur during the transmission of digital information from one place to another. Low-density parity-check codes can detect and correct transmission errors if one encodes the original information by adding redundant bits. In practice, heuristic iterative decoding algorithms are used to decode the received vector. However, these algorithms may fail to decode if the received vector contains multiple errors. We consider decoding the received vector with minimum error as an integer programming (IP) problem and propose a branch-and-price method for its solution. We improve the performance of our method by introducing heuristic feasible solutions and adding valid cuts to the mathematical formulation. Our computational experiments reveal that our branch-cut-and-price algorithm significantly improves solvability of the problem compared to a state-of-the-art IP decoder in the literature and has superior error performance than the conventional sum-product algorithm.

Keywords: Telecommunications, LDPC decoding, integer programming, branch-cut-and-price algorithm.

*Corresponding author. Tel.: +90 2123596771; fax: +90 2122651800.
E-mail addresses: banu.kabakulak@bilgi.edu.tr (B. Kabakulak), caner.taskin@boun.edu.tr (Z. C. Taşkın), ali.pusane@boun.edu.tr (A. E. Pusane).

1 Introduction and Literature Review

Low-density parity-check (LDPC) codes are used in the digital communication systems to detect and correct errors that may occur during the data transmission. LDPC codes were first investigated by Gallager (1962) and rediscovered in the 1990s (MacKay, 1999; MacKay and Neal, 1997). LDPC codes are now being used in hard disk drive read channels, wireless communication standards (IEEE 802.11n, IEEE 802.11ac, IEEE 802.16e WiMax), digital television broadcast standard DVB-S2, and more recently in flash solid state drives (Karger et al., 2014).

In a digital communication system, the source encodes the information as the *codeword*, which is recognized by its LDPC code as an error-free information. In the case of an erroneously received information, the sink implements LDPC code-based decoding algorithms to find the locations of the errors and correct them. Maximum likelihood (ML) decoding aims to find the nearest codeword to the received vector among the (possibly) exponential-many codewords of an LDPC code. Alternative integer programming (IP) formulations of the ML decoding problem are given in Feldman et al. (2005) and Yang et al. (2008), which we introduce in Section 3.1 as *integer programming master* (IPM) and *exact model* (EM), respectively. The ML decoding problem is NP-hard and the computational complexity of an ML decoder limits its implemention in the practical applications (Berlekamp et al., 1978). In the literature, there are two main heuristic approaches to obtain a real-time decoder: iterative decoding algorithms and linear programming (LP) based decoders. We summarize different decoding approaches in Table 1.

Gallager A and sum-product (SP) algorithms are commonly used heuristic iterative message-passing decoding algorithms due to their low complexity and low decoding latency (Tanner, 1981; Kschischang et al., 2001; Richardson and Urbanke, 2001). These heuristic methods can give close results to the ML decoding on the sparse LDPC codes (Leiner, 2005). However, they do not ensure the optimality of the decoded vector, and their error correction capability decreases significantly as the error probability or the code density increases. Besides, they may fail to decode if the received vector includes multiple errors or there are small cycles in the bipartite graph representation, namely *Tanner graph* (TG), of the LDPC code. The SP algorithm was further investigated in the literature to develop its variants with improved complexity (Fossorier et al., 1999; Hu et al., 2001; Sarajlić et al., 2014). Vontobel and Koetter (2007) implement an iterative approach similar to the SP algorithm for low complexity LP

decoding, and the technique is improved in Burshtein (2009).

Table 1: Comparison of the LDPC decoding approaches

Decoding Approach	Article (Formulation)	ML Decoding	Output	Effect of the Cycles in TG on the Error Correction
IP Decoding	This work (IPM) Yang et al. (2006) (IPM)	Yes	a codeword(*)	no impact
LP Decoding	Feldman et al. (2005) (Relaxed IPM) Yang et al. (2008) (Relaxed EM) Tanatmis et al. (2010) (Relaxed EM)	Not guaranteed	a codeword or a pseudocodeword(**)	decrease
Iterative Decoding	Gallager A (None) Sum-Product (None)	Not guaranteed	a codeword or an infeasible vector or no solution	decrease

(*) a codeword is a binary feasible solution of the IPM or EM formulation.
(**) a pseudocodeword is a (fractional) feasible solution of the Relaxed IPM or Relaxed EM formulation.

The (fractional) extreme vertices of an LP formulation are known as *pseudocodewords*. Since the codewords are binary vectors, the error correction capability of an LP decoder highly depends on the pseudocodewords. Feldman et al. (2005) focus on *the LP relaxation of the IPM* (LPM) model, which has the codewords as the extreme points if there are no cycles in the TG. However, a cycle-free LDPC code is not possible in practice, which limits the error correction capability of their LP decoder. Then, they aim to trim the fractional pseudocodewords by including their exponential-many valid inequalities to the LPM model without any separation algorithm. The proposed LP decoder can perform better than the SP algorithm only for the small codelengths since the exponential-many variables of the LPM model are executed without any column generation method.

Yang et al. (2006) develop an ML decoder on the IPM formulation with their basic branch-and-bound algorithm which limits the depth of the search tree to reduce the computational complexity. Then, they propose a nonlinear Lagrangean relaxation formulation for the LPM model to design an approximate LP decoder with lower complexity than their ML-decoder. The decoding performance of the LP decoder is better than SP decoder only for special type of regular LDPC codes. Barman et al. (2013) and Zhang and Siegel (2013) also investigate LP decoders which implement Lagrangean relaxation techniques on the LPM formulation.

Chertkov and Stepanov (2008) design an efficient pseudocodeword search heuristic which iteratively updates the weights of the received information in their LP decoder. Yang et al.

(2008) reformulate the ML decoding problem as the EM formulation having fewer constraints than the IPM model proposed in Feldman et al. (2005). In their LP decoder, they consider *the LP relaxation of the EM* (LEM) model and decompose a constraint with many-variables into several subconstraints with fewer-variables by introducing auxiliary variables. They mathematically show that the LEM model with these new constraints is equivalent to the LPM model in Feldman et al. (2005).

The exponential-many valid inequalities of the LPM model introduced in Feldman et al. (2005) are also valid for the LEM formulation. Tanatmis et al. (2010) address the LEM formulation and they propose a separation algorithm to select a subset of these valid inequalities to improve the error correction capability of their LP decoder. Zhang and Siegel (2012) define their LP model only with the box-constraints on the variables and the valid inequalities in Feldman et al. (2005). In their LP decoder, they eliminate the pseudocodewords with a similar separation algorithm to Tanatmis et al. (2010).

In this paper, we focus on the ML decoding problem, which finds the optimal decoding for arbitrary LDPC codes. We utilize the IPM formulation in Feldman et al. (2005) and aim to find the closest codeword to the received vector with low computational complexity. We make the following contributions toward this end:

1. We introduce new theoretical results for the EM and IPM formulations (see Section 3.1).
2. We develop a column generation technique to find the required variables among the exponential-many decision variables of the IPM formulation (see the branch-and-price (BP) method in Section 3.2).
3. We improve the computational complexity of our BP method by searching for a near-ML initial codeword (solution) via our *RandSum* heuristic (see the branch-and-price-random-sum (BPRS) method in Section 3.3.2).
4. We further propose a separation algorithm to generate the required constraints among the exponential-many constraints of the IPM formulation (see the branch-cut-and-price (BCP) method in Section 3.3.3).
5. We compare the performance of our BCP method with two state-of-the-art decoders from the literature: the EM-based decoder (EMD), which is the IP decoder counterpart of the LP decoder by Tanatmis et al. (2010), and the iterative decoder SP (see Section 4).

6. Our BCP decoder solves the ML decoding problem with lower complexity than EMD and has significantly better error correction capability than SP for the practical code lengths (approximately $n = 4000$).

2 Problem Definition

In a digital communication system, information is sent from a source to a sink over a noisy communication channel as shown in Figure 1. The original information is a k-bits long binary sequence $\mathbf{u} = (u_1\ u_2\ ...\ u_k)$ $(u_i \in \{0, 1\})$. The information $\mathbf{u}$ is encoded with a $k \times n$ generator matrix $\mathbf{G}$ through the operation $\mathbf{e} = \mathbf{uG}$ (mod 2) in the encoder. That is, $(n - k)$ redundant parity-check bits are added to $\mathbf{u}$, and n-bits long $(n \geq k)$ encoded vector $\mathbf{e} = (e_1\ e_2\ ...\ e_n)$ $(e_i \in \{0, 1\})$ is obtained.

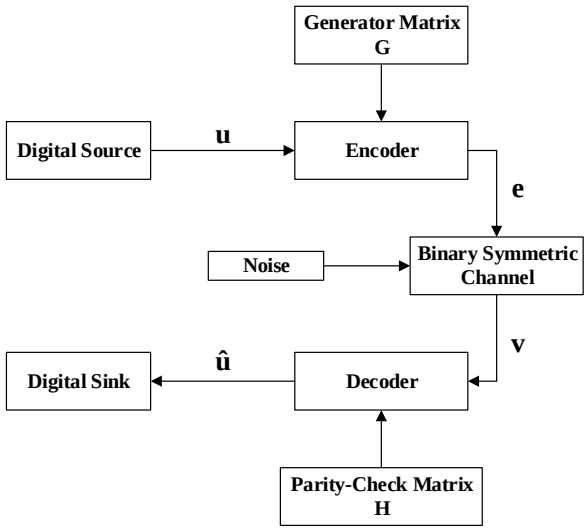

Figure 1: Digital communication system diagram

The encoded vector $\mathbf{e}$ is transmitted over a noisy communication channel to the receiver. Binary symmetric channel (BSC) is generally used in the literature to model a noisy communication channel (MacKay, 2003). In BSC an error occurs on a bit e_i with probability p and its value flips, i.e., a bit 0 is received as 1, and vice versa. The decoder tests the correctness of n-bits long received vector $\mathbf{v}$ with a $(n - k) \times n$ parity-check matrix $\mathbf{H}$. The received vector $\mathbf{v}$ is detected to be erroneous if $\mathbf{vH}^{\mathrm{T}} \neq \mathbf{0}$ (mod 2). In this case, the decoder runs decoding algorithms to fix the errors and estimates the original information as $\hat{\mathbf{u}}$ (Moon, 2005).

One can obtain a generator matrix $\mathbf{G}$, which is not necessarily unique, from the $\mathbf{H}$ matrix

with full row rank by carrying out binary arithmetic (MacKay, 2003). A vector $\mathbf{e}$ is a *codeword* if $\mathbf{e}\mathbf{H}^{\mathrm{T}} = \mathbf{0} \pmod 2$. Since $\mathbf{G}\mathbf{H}^{\mathrm{T}} = \mathbf{0} \pmod 2$ holds for any $(\mathbf{G}, \mathbf{H})$ pair, each row of $\mathbf{G}$ is a codeword. That is, the codewords are in the null space of the $\mathbf{H}$ matrix and $\mathbf{G}$ is a basis for the null space.

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_s^1 & \mathbf{I}_s^2 & \mathbf{I}_s^3 & \mathbf{I}_s^4 & \mathbf{I}_s^5 & \mathbf{I}_s^6 \\ \mathbf{I}_s^7 & \mathbf{I}_s^8 & \mathbf{I}_s^9 & \mathbf{I}_s^{10} & \mathbf{I}_s^{11} & \mathbf{I}_s^{12} \\ \mathbf{I}_s^{13} & \mathbf{I}_s^{14} & \mathbf{I}_s^{15} & \mathbf{I}_s^{16} & \mathbf{I}_s^{17} & \mathbf{I}_s^{18} \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} 1&0&0&1&0&1&1&0&0&1&1&0 \\ 0&1&1&0&1&0&0&1&1&0&0&1 \\ 0&1&1&0&0&1&1&0&0&1&0&1 \\ 1&0&0&1&1&0&0&1&1&0&1&0 \\ 0&1&0&1&1&0&0&1&1&0&0&1 \\ 1&0&1&0&0&1&1&0&0&1&1&0 \end{bmatrix}$$

(a) (b)

Figure 2: A (3, 6)-regular $\mathbf{H}$ matrix

(J, K)-regular LDPC codes are the members of linear block codes that can be represented by a parity-check matrix $\mathbf{H}$ having J-many ones at each column and K-many ones at each row. One can generate a (J, K)-regular $\mathbf{H}$ matrix of dimensions $(Js \times Ks)$ by randomly permuting the columns of an $s \times s$ identity matrix $\mathbf{I}_s$. Regularity of the matrix is provided through augmenting identity matrices K times at each row and J times at each column. The generic structure of a $(3, 6)$-regular $\mathbf{H}$ matrix is given in Figure 2a, where $\mathbf{I}_s^i$ represents the ith randomly permuted identity matrix. We give an example of a $(3, 6)$-regular $\mathbf{H}$ matrix with $s = 2$ and dimensions $n - k = 6$, $n = 12$ in Figure 2b.

$$\mathbf{H}^r = \begin{bmatrix} 1&0&0&1&0&1&1&0&0&1&1&0 \\ 0&1&1&0&1&0&0&1&1&0&0&1 \\ 0&1&1&0&0&1&1&0&0&1&0&1 \\ 1&0&1&0&0&1&1&0&0&1&1&0 \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} 1&1&0&0&0&0&1&1&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&1&0 \\ 1&1&1&1&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&1&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0&1&0&0 \\ 1&1&0&0&1&0&1&0&0&0&0&0 \end{bmatrix}$$

(a) (b)

Figure 3: A full row rank $\mathbf{H}^r$ and a generator $\mathbf{G}$ matrices for the $\mathbf{H}$ matrix in Figure 2b

After eliminating the redundant rows from the $\mathbf{H}$ matrix in Figure 2b, one can obtain a full row ranked $\mathbf{H}^r$ matrix of dimensions $n - k' = 4$, $n = 12$ as given in Figure 3a. Note that the row reduction does not change the null space ($\subseteq \mathbb{B}^k$), i.e., the set of codewords remains the same for both of the $\mathbf{H}$ and $\mathbf{H}^r$ matrices. Figure 3b shows a generator matrix $\mathbf{G}$ for the

null space of $\mathbf{H}^r$. Without loss of generality, we can use the $\mathbf{G}$ matrix of dimensions $k' \times n$ to encode the k-bit ($k \leq k'$) original information $\mathbf{u}$ after augmenting $(k' - k)$-many zero bits. In our example, having $k = 6$ and $k' = 8$, we update a 6-bit original information $\mathbf{u} = (0\ 1\ 1\ 0\ 1\ 1)$ to $\mathbf{u}' = (0\ 1\ 1\ 0\ 1\ 1\ \textit{0}\ \textit{0})$. The encoded vector $\mathbf{e} = \mathbf{u}'\mathbf{G}\ (\text{mod}\ 2) = (0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1)$ is a codeword since $\mathbf{e}\mathbf{H}^{\mathrm{T}} = \mathbf{0}\ (\text{mod}\ 2)$. One can span exponential-many, i.e., $\mathcal{O}(2^k)$, codewords in the null space of $\mathbf{H}$ with the generator matrix $\mathbf{G}$.

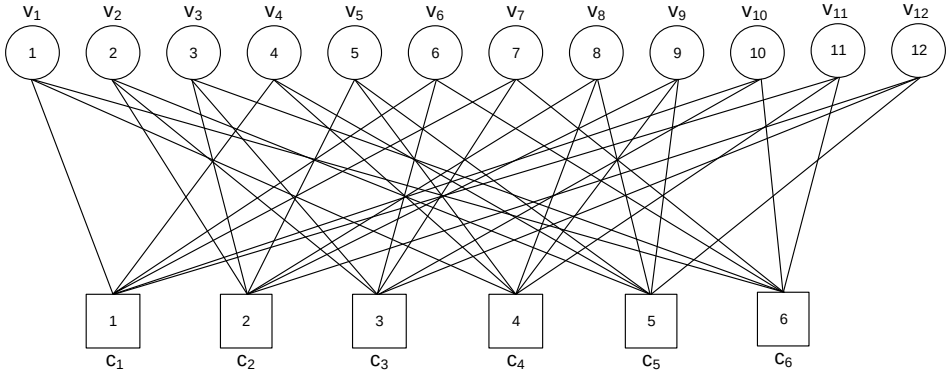

Figure 4: TG representation of the $\mathbf{H}$ matrix in Figure 2b

An LDPC code can alternatively be represented as a TG corresponding to the $\mathbf{H}$ matrix (Tanner, 1981). As shown in Figure 4, on one part of the TG there is a *variable node* i (v_i), $i \in \{1, ..., n\}$, for each bit of the received vector $\mathbf{v}$. Each row of the $\mathbf{H}$ matrix represents a parity-check equation and corresponds to a *check node* j $(c_j), j \in \{1, ..., n - k\}$, in the other part of the TG. A check node c_j is said to be *satisfied* if its parity-check equation is equal to zero (mod 2). As an example, the parity-check equation for c_4 is $c_4 = v_1 + v_4 + v_5 + v_8 + v_9 + v_{11}$ (mod 2) as given in Figure 5. The check node c_4 is satisfied when $v_i = 1$ for the indices $i \in S = \{1, 5, 8, 11\}$, i.e., $c_4 = 1 + 0 + 1 + 1 + 0 + 1 = 0$ (mod 2). The index set of the adjacent check (variable) nodes to a variable node i (check node j) is represented by $N(v_i)(N(c_j))$. Note that we can satisfy the check node c_j by setting $v_i = 1$ for $i \in S$, where S is an even cardinality subset of $N(c_j)$. Moreover, a codeword satisfies all the parity-check equations as in the case of the encoded codeword $\mathbf{e} = (0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1)$ by definition. The *degree* of a v_i (c_j) is the number of the adjacent check nodes (variable nodes) on the TG. That is, degree of a v_i is $d(v_i) = |N(v_i)|$ and c_j is $d(c_j) = |N(c_j)|$.

An *ML decoding* algorithm explores exponential-many codewords of the $\mathbf{H}$ matrix, i.e., an exponential-time algorithm, and decodes the received vector $\mathbf{v}$ to the nearest codeword by utilizing a distance function. As an example, assume that we received the encoded codeword

$$\mathbf{vH}^{\mathrm{T}} = \begin{bmatrix} v_1 + v_4 + v_6 + v_7 + v_{10} + v_{11} \\ v_2 + v_3 + v_5 + v_8 + v_9 + v_{12} \\ v_2 + v_3 + v_6 + v_7 + v_{10} + v_{12} \\ v_1 + v_4 + v_5 + v_8 + v_9 + v_{11} \\ v_2 + v_4 + v_5 + v_8 + v_9 + v_{12} \\ v_1 + v_3 + v_6 + v_7 + v_{10} + v_{11} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} \pmod 2$$

Figure 5: The parity-check equations for the **H** matrix in Figure 2b

$\mathbf{e} = (0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1)$ as the vector $\mathbf{v} = (1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0)$ and Figure 6 shows the codewords of the **H** matrix in Figure 2b. Let the codewords $\mathbf{cw_1}, ..., \mathbf{cw_8}$ be the rows of the **G** matrix in Figure 3b and the *Hamming distance function*, i.e., $Hamming(\mathbf{cw_i}, \mathbf{v})$, count the number of different entries between the ith codeword $\mathbf{cw_i}$ and the received vector **v**.

As illustrated in Figure 6, an ML decoder does not choose the original codeword **e** since there are closer codewords to the received vector **v**, such as $\mathbf{cw_1}$ with $Hamming(\mathbf{cw_1}, \mathbf{v}) = 2$. Assuming that $\mathbf{cw_1} = (1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0)$ is the closest codeword to **v**, the estimate for the original information $\mathbf{u} = (0\ 1\ 1\ 0\ 1\ 1)$ will be $\hat{\mathbf{u}} = (1\ 1\ 0\ 0\ 0\ 0)$. Note that, when there is high error probability in the channel, the original information **u** and its estimate $\hat{\mathbf{u}}$ may not be identical as in this case.

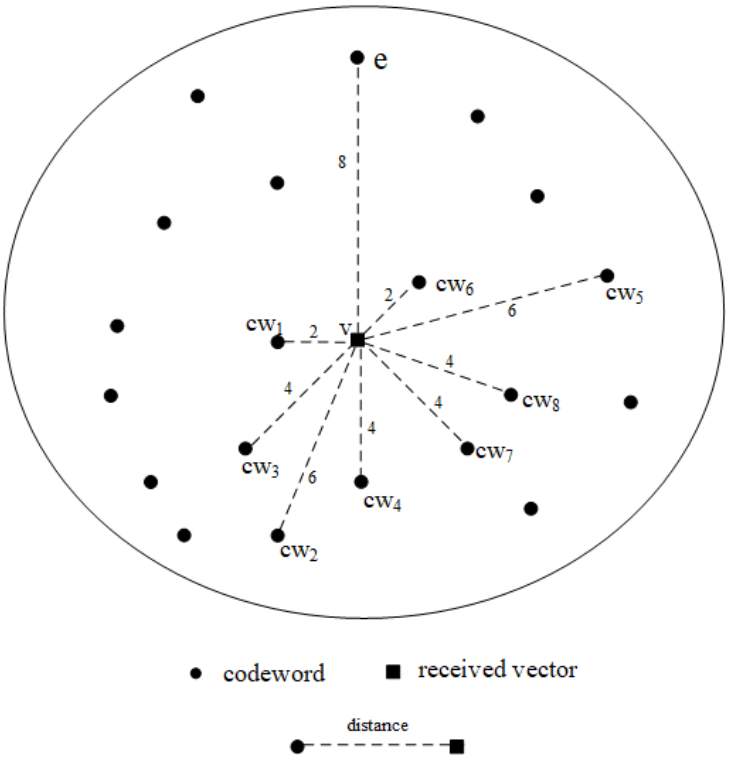

Figure 6: ML decoding of the received vector **v** in the null space of **H**

Bit error rate (BER) is a metric in the telecommunications literature to evaluate the performance of a decoding algorithm. Let $\mathbf{f}$ be the n-bit long decoded codeword of the received vector $\mathbf{v}$. As given in equation (1), BER is the rate of the different decoded bits from the original codeword $\mathbf{e}$ (MacKay, 2003). Note that BER is zero, if and only if $\mathbf{e} = \mathbf{f}$.

$$\text{BER} = \frac{\sum_{i=1}^{n} \mid e_i - f_i \mid}{n} \tag{1}$$

For our ML decoding example, the decoded codeword is $\mathbf{f} = \mathbf{cw_1}$, and BER can be given as

$$\text{BER} = \frac{\sum_{i=1}^{12} \mid e_i - cw_{1i} \mid}{12} = \frac{8}{12} = 0.67.$$

In this study, we focus on developing ML decoding algorithms using optimization techniques. We first propose a branch-and-price (BP) algorithm (in Section 3.2) for the IPM formulation in Feldman et al. (2005), and introduce improvement techniques to BP. In particular, we provide feasible solutions to BP via our *RandSum* heuristic, which gives rise to our branch-and-price-random-sum (BPRS) method given in Section 3.3.2. Furthermore, we tighten the node relaxations with our valid cut separation algorithm, and build our branch-cut-and-price (BCP) method (explained in Section 3.3.3).

3 Solution Methods

In this section, we introduce the exact model (EM) and integer programming master (IPM) formulations for the ML decoding problem in the literature, and give the details of our BP, BPRS, and BCP methods. We summarize the terminology used in this paper in Table 2.

Table 2: List of the symbols

	Parameters		
C	set of the check nodes	k	length of the original information
c_j	check node j	n	length of the encoded codeword, $\|V\|$
V	set of the variable nodes		number of columns in $\mathbf{H}$
v_i	variable node i	m	$n-k$, $\|C\|$, number of rows in $\mathbf{H}$
$d(c_j)(d(v_i))$	degree of $c_j(v_i)$ in TG	p	error probability in BSC
$N(c_j)(N(v_i))$	index set of the variable (check) nodes adjacent to $c_j(v_i)$	$\mathbf{u}$	original information
$\mathbf{G}$	generator matrix	$\mathbf{e}$	encoded codeword
$\mathbf{H}$	parity-check matrix	$\mathbf{v}$	received vector
ε_j	set of the feasible local codewords for c_j	t_{max}	number of trials in *RandSum* heuristic
γ_i	log-likelihood ratio for the bit i		
	Decision Variables		
f_i	ith bit of the decoded codeword	μ_j	dual variable for the constraints (11)
w_{jS}	1 if the local codeword S of c_j is selected, 0 otherwise	τ_{ij}	dual variable for the constraints (12)
ℓ_j	an auxiliary integer variable	ζ_j	optimal objective function value
x_i	1 if $i \in S$ of c_j, 0 otherwise		of Subproblem(j)

3.1 Mathematical Formulations

The EM formulation represents the columns and rows of an $(n-k) \times n$ parity-check matrix $\mathbf{H}$ with the index sets $V = \{1, ..., n\}$ and $C = \{1, ..., n-k\}$, respectively (Yang et al., 2008). In EM, H_{ji} is the (j,i)-entry of $\mathbf{H}$ matrix, f_i is a binary variable denoting the value of the ith code bit, and ℓ_j is an integer variable. Here, $\mathbf{v}$ represents the received vector.

Exact Model (EM):

$$z_{EM} = \min \sum_{i:v_i=1} (1 - f_i) + \sum_{i:v_i=0} f_i \tag{2}$$

$$\text{s.t.} \sum_{i \in V} H_{ji} f_i = 2\ell_j, \qquad \forall j \in C \tag{3}$$

$$f_i \in \{0,1\}, \qquad \forall i \in V, \tag{4}$$

$$\ell_j \geq 0,\ \ell_j \in \mathbb{Z}, \qquad \forall j \in C. \tag{5}$$

The constraints (3) guarantee that the decoded vector $\mathbf{f}$ satisfies the equality $\mathbf{f}\mathbf{H}^{\mathrm{T}} = \mathbf{0} \pmod 2$. The objective (2) minimizes the Hamming distance between the decoded codeword $\mathbf{f}$ and the received vector $\mathbf{v}$, i.e., $z_{EM} = \min_{\mathbf{f}} z_{EM}(\mathbf{f}) = \min_{\mathbf{f}} \mathit{Hamming}(\mathbf{f}, \mathbf{v})$. That is, the aim is to find

the nearest codeword $\mathbf{f}$ to the received vector $\mathbf{v}$. The constraints (4) and (5) set the binary and integrality restrictions on the decision variables $\mathbf{f}$ and $\boldsymbol{\ell}$, respectively.

One can obtain *the LP relaxation of EM* (LEM) with the objective function value z_{LEM} by replacing the constraints (4) and (5) with the following:

$$0 \leq f_i \leq 1,\ \ell_j \geq 0, \qquad \forall i \in V,\ j \in C. \tag{6}$$

Note that the EM formulation has $(n+m)$–variables and $(n+2m)$–many constraints. Tanatmis et al. (2010) propose an LP decoder which utilizes the LEM formulation and the valid inequalities (27) given in Section 3.3.3. In this study, we obtain a benchmark EM decoder (EMD), which is the IP decoder counterpart of the LP decoder of Tanatmis et al. (2010), by implementing EMD via a commercial optimization solver (see Section 4).

An alternative objective function is the *log-likelihood objective*, i.e., $\min_{\mathbf{f}} \mathit{LogLike}(\mathbf{f}, \mathbf{v})$, which can be given as

$$\min \sum_{i \in V} \gamma_i f_i. \tag{7}$$

In the $\mathit{LogLike}(\mathbf{f}, \mathbf{v}) = \sum_{i \in V} \gamma_i f_i$ distance function, $\mathbf{f}$ is the decoded codeword and the coefficient γ_i is the log-likelihood ratio for the received bit v_i, which can be calculated with the equation (8).

$$\gamma_i = log(\tfrac{Pr(v_i|f_i=0)}{Pr(v_i|f_i=1)}) \tag{8}$$

In BSC with error probability p, $\gamma_i = \log[p/(1-p)]$ for the bit $v_i = 1$, and $\gamma_i = \log[(1-p)/p]$ for $v_i = 0$ as given in Feldman et al. (2005). Proposition 1 shows the equivalence of the objectives (2) and (7).

Proposition 1. *The Hamming distance objective (2) and the log-likelihood objective (7) have the same optimal solution set for the decoded codeword* $\mathbf{f}$ *when* $p < 0.5$.

Proof. The log-likelihood objective can be written as

$$\min - \sum_{i:v_i=1} a f_i + \sum_{i:v_i=0} a f_i \tag{9}$$

where $a = \log[(1-p)/p]$. Note that $a \geq 0$ for $0 < p < 0.5$, and it is constant.

Similarly, the Hamming distance objective can be written as

$$\min -\sum_{i:v_i=1} f_i + \sum_{i:v_i=0} f_i + \bar{c} \tag{10}$$

where $\bar{c} = \sum_{i:v_i=1} 1$.

Note that, the distance functions are linearly related as $Hamming(\mathbf{f}, \mathbf{v}) = \frac{1}{a} LogLike(\mathbf{f}, \mathbf{v}) + \bar{c}$. Hence, the corresponding objective functions are equivalent and they have the same optimal solution set. □

The IPM formulation, an alternative formulation for the ML decoding problem, is based on the TG representation of an **H** matrix (Feldman et al., 2005). A *local codeword* can be formed by assigning a value in {0, 1} to each variable node $i \in N(c_j)$. A local codeword is *feasible* if the values of the adjacent variable nodes $i \in N(c_j)$ sum to zero (mod 2). For a node c_j, the set of the feasible local codewords can be given as $\varepsilon_j = \{S \subseteq N(c_j) : |S| \ even\}$. We can satisfy c_j if we set each bit in $S \in \varepsilon_j$ to 1, and all other bits in $N(c_j)$ to 0. One can observe that $\emptyset \in \varepsilon_j$ for all c_j, since $S = \emptyset$ trivially satisfies a check node.

In Figure 7, we give the neighbors of the node c_3 in Figure 4 as an example. The parity-check equation for c_3 can be given as $c_3 = v_2 + v_3 + v_6 + v_7 + v_{10} + v_{12}$ (mod 2). We can obtain $c_3 = 0$ by setting an even number of adjacent variable nodes to value 1, and the remainings to 0. For instance, $S = \{2, 6, 7, 12\}$ is a feasible local codeword, since $c_3 = 1 + 0 + 1 + 1 + 0 + 1 = 0$ (mod 2). A *codeword* is a {0, 1} assignment of v_i values for $i \in V$ that gives $c_j = 0$ for all $j \in C$. One can obtain a codeword by choosing a feasible local codeword for each c_j that conforms with the feasible local codewords of the other check nodes. For example, (0 1 0 0 0 1 1 1 1 0 0 1) is a codeword for the TG in Figure 3.

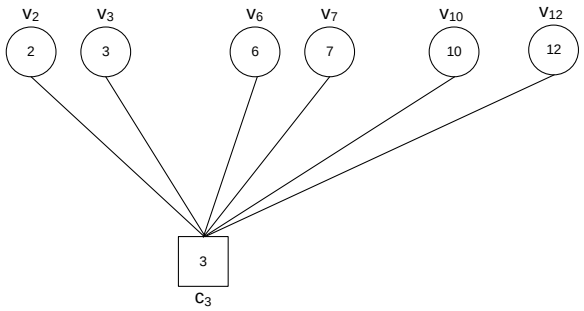

Figure 7: Neighbors of the check node c_3 in Figure 4, $N(c_3)$

Integer Programming Master (IPM):

$$z_{IPM} = \min \sum_{i \in V} \gamma_i f_i \tag{7}$$

$$\text{s.t.} \sum_{S \in \varepsilon_j} w_{jS} = 1, \qquad \forall j \in C \tag{11}$$

$$f_i - \sum_{S \in \varepsilon_j : i \in S} w_{jS} = 0, \qquad \forall \text{ edge } (i,j) \tag{12}$$

$$f_i \geq 0, \qquad \forall i \in V \tag{13}$$

$$w_{jS} \in \{0,1\}, \qquad \forall j \in C, \forall S \in \varepsilon_j \tag{14}$$

In IPM model, the binary decision variable w_{jS} is one if the feasible local codeword $S \in \varepsilon_j$ of the node c_j is selected, and zero otherwise. Hence, the decision variables $\mathbf{w}$ represent a feasible solution of the parity-check equations and f_i variable represents the decoded value of the bit i. We can obtain a trivial solution (an upper bound) of IPM with $w_{j\emptyset} = 1$ for all $j \in C$ and $f_i = 0$ for all $i \in V$. We obtain the LPM model, i.e., *the LP relaxation of IPM*, with the objective function value z_{LPM} by relaxing the constraints (14) as

$$w_{jS} \geq 0, \qquad \forall j \in C, \forall S \in \varepsilon_j. \tag{15}$$

Proposition 2. *In the LPM formulation, the f_i values are binary integral for all $i \in V$ if and only if the w_{jS} values are binary integral for all (j,S) pairs $j \in C$ and $S \in \varepsilon_j$.*

Proof. ($\Leftarrow$) Assume that the w_{jS} values are integral for all (j,S) pairs. The constraints (12) imply that the f_i values are integral for all $i \in V$, since each f_i is the sum of the integer numbers. Besides, we know that $f_i = \sum_{S \in \varepsilon_j : i \in S} w_{jS}$ by the constraints (12). Then, $f_i = \sum_{S \in \varepsilon_j : i \in S} w_{jS} \leq \sum_{S \in \varepsilon_j} w_{jS} = 1$ by the constraints (11), implying that $f_i \leq 1$. Hence, the set $\{0,1\}$ represents the possible values of the f_i variables.

($\Rightarrow$) Assume for the contradiction that the f_i values are integral but there exists a (j,S) pair such that w_{jS} is not integral. By the constraints (11), we know $\sum_{S \in \varepsilon_j} w_{jS} = 1$. Hence, for at least two w_{jS} variables, say $w_{j,S_1} = \alpha$ and $w_{j,S_2} = \beta$ with $\alpha, \beta > 0$ and $\alpha + \beta \leq 1$, we have fractional values. Since $S_1 \neq S_2$, there exists $\hat{\imath} \in S_2 \setminus S_1$ without loss of generality.

For the variable node $\hat{\imath}$ and check node j, we have the constraint $f_{\hat{\imath}} = \sum_{S \in \varepsilon_j : \hat{\imath} \in S} w_{jS}$ for the edge $(\hat{\imath}, j)$. The edge $(\hat{\imath}, j)$ exists, since $\hat{\imath} \in S_2 \in \varepsilon_j$ which implies that $\hat{\imath} \in N(c_j)$. We know that $\hat{\imath} \notin S_1$, meaning that $w_{j,S_1} = \alpha$ will not be in the sum. This means $f_{\hat{\imath}} = \sum_{S \in \varepsilon_j : \hat{\imath} \in S} w_{j,S} \leq 1 -$

$w_{j,S_1} = 1-\alpha < 1$. Moreover, w_{j,S_2} will be in the sum, since $\hat{\imath} \in S_2$. This gives $f_{\hat{\imath}} \geq w_{j,S_2} = \beta > 0$. As a result, $0 < f_{\hat{\imath}} < 1$, i.e., $f_{\hat{\imath}}$ is a fractional value. This contradicts with our assumption that all f_i values are integral. Hence, we conclude that if f_i is integral for all $i \in V$, then the w_{jS} values are also integral for all (j, S) pairs.

Combining these results, we conclude that the f_i values are binary integral for all $i \in V$ if and only if the w_{jS} values are binary integral for all (j, S) pairs $j \in C$ and $S \in \varepsilon_j$. □

One can observe that there are exponential number, i.e., $(n + \sum_{j \in C} 2^{d(c_j)-1})$–many, of variables in the IPM formulation. Hence, it cannot be solved optimally in an acceptable amount of time for the real-sized (approximately $n = 4000$) LDPC codes (see Section 4). Another observation is that although the IPM and EM formulations have different objective functions, they share the same optimal solution set according to Proposition 1.

3.1.1 On the Strength of the LEM and LPM Formulations

In this section, we prove some theoretical results related with the LP relaxations of the EM and IPM formulations for the ML decoding problem, namely the LEM and LPM formulations, respectively.

Proposition 3. *The optimal objective function value of LEM is zero, i.e., $z_{LEM} = 0$, for all $(\mathbf{v}, \mathbf{H})$ instances.*

Proof. Let $(\mathbf{v}, \mathbf{H})$ be an instance of LEM. The objective of LEM is a distance function which means the optimal objective value is nonnegative, i.e., $z_{LEM} \geq 0$.

Assume that we have $\mathbf{f} = \mathbf{v}$ and $\ell_j = \frac{\sum_{i \in V} H_{ij} f_i}{2}$ for all $j \in C$. Then, the vector $(\mathbf{f}, \boldsymbol{\ell})$ is feasible for LEM since $0 \leq f_i \leq 1$ for all $i \in V$ and $\ell_j \geq 0$ for all $j \in C$ with the objective value $z_{LEM}(\mathbf{f}) = \mathit{Hamming}(\mathbf{f}, \mathbf{v}) = 0$. This solution is optimal since $0 \leq z_{LEM} = z_{LEM}(\mathbf{f}) = 0$. Then, the optimal objective function value $z_{LEM} = 0$ for all $(\mathbf{v}, \mathbf{H})$ instances. □

Proposition 4. *The Hamming distance objective (2) of LPM is zero if and only if $\mathbf{v}$ is a codeword.*

Proof. As we have explanied in Section 2, the received vector $\mathbf{v}$ is binary by definition.

($\Rightarrow$) Assume $z_{LPM} = \mathit{Hamming}(\mathbf{f}, \mathbf{v}) = 0$. Then, $\mathbf{f} = \mathbf{v}$ and $\mathbf{f}$ is a binary vector. The binary $\mathbf{f}$ vector indicates binary w_{jS} values for all (j, S) pairs by Proposition 2. Then, $(\mathbf{f}, \mathbf{w})$ is an integral solution of LPM and a feasible solution of IPM. Any feasible $\mathbf{f}$ vector of IPM is a codeword means $\mathbf{v}$ is a codeword as well.

($\Leftarrow$) Assume that the received vector $\mathbf{v}$ is a codeword. Then, $\mathbf{f} = \mathbf{v}$ is a feasible solution of LPM with the objective value $z_{LPM} = \mathit{Hamming}(\mathbf{f}, \mathbf{v}) = 0$. This solution is optimal since $z_{LPM} \geq 0$. $\square$

Example: Let us consider the instance $\mathbf{v} = (0\ 1\ 1)$ and $\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ where $n = 3$ and $k = 1$. Note that the received vector $\mathbf{v}$ is not a codeword since $\mathbf{v}\mathbf{H}^{\mathrm{T}} \neq \mathbf{0} \pmod 2$.

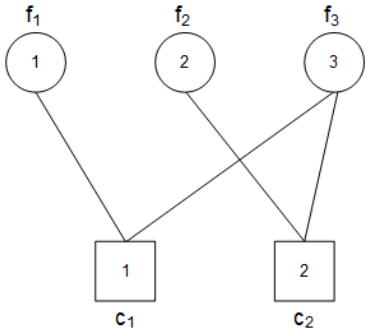

Figure 8: Tanner graph of the example instance $(\mathbf{v}, \mathbf{H})$

The vectors $\mathbf{cw_1} = (0\ 0\ 0)$ and $\mathbf{cw_2} = (1\ 1\ 1)$ are the codewords of $\mathbf{H}$ among $2^{n=3} = 8$ binary vector alternatives. We observe that $\mathit{Hamming}(\mathbf{cw_1}, \mathbf{v}) = 2$ and $\mathit{Hamming}(\mathbf{cw_2}, \mathbf{v}) = 1$. Then, the ML decoding, which can be found by the EM or IPM formulations, of the received vector $\mathbf{v}$ is $\mathbf{cw_2}$.

LEM:

$$
\begin{aligned}
z_{LEM} = \min \quad & f_1 + (1 - f_2) + (1 - f_3) \\
\text{s.t.} \quad & f_1 + f_3 = 2\ell_1 \\
& f_2 + f_3 = 2\ell_2 \\
& f_1, f_2, f_3 \in [0, 1] \\
& \ell_1, \ell_2 \geq 0.
\end{aligned}
$$

LPM:

$$
\begin{aligned}
z_{LPM} = \min \quad & f_1 + (1 - f_2) + (1 - f_3) \\
\text{s.t.} \quad & w_{1,S_1} + w_{1,S_2} = 1 \\
& w_{2,S_1} + w_{2,S_2} = 1 \\
& f_1 = w_{1,S_2},\ f_2 = w_{2,S_2} \\
& f_3 = w_{1,S_2},\ f_3 = w_{2,S_2} \\
& f_1, f_2, f_3 \geq 0 \\
& w_{1,S_1}, w_{1,S_2}, w_{2,S_1}, w_{2,S_2} \geq 0.
\end{aligned}
$$

For the LEM formulation, $\mathbf{f} = (0\ 1\ 1)$ and $\ell = (0.5\ 1)$ is an optimal solution with objective value $z_{LEM} = 0$. For the LPM formulation, the set of feasible local codewords for the check node c_1 is $\varepsilon_1 = \{S_1 = \emptyset, S_2 = \{1, 3\}\}$ and for the check node c_2 is $\varepsilon_2 = \{S_1 = \emptyset, S_2 = \{2, 3\}\}$ as given in Figure 8. Note that, $\mathbf{v} = (0\ 1\ 1)$ is not a feasible solution of LPM, and the optimal solution of LPM is $\mathbf{f} = (1\ 1\ 1)$ with $z_{LPM} = 1$. In this example, LPM finds the ML decoding which is not necessarily true in general. $\square$

As Propositions 3 and 4 claim and the above example illustrates, the LEM formulation is a decision problem, i.e., $z_{LEM} = 0$, and the LPM formulation provides strictly positive lower bound when there are errors in the received vector, i.e., $z_{LPM} > 0$. In the following sections, we discuss the details of our BP, BPRS, and BCP algorithms, which can effectively solve an IPM instance by gradually generating columns as needed.

3.2 Branch-and-Price Algorithm

In our BP algorithm for the IPM formulation, we aim to find the nearest codeword to the received vector $\mathbf{v}$. We start with a restricted LPM (RLPM) which contains a subset of w_{jS} variables. As mentioned before, $w_{j\emptyset} = 1$ for all $j \in C$ constitutes a feasible solution for LPM. Hence, $w_{j\emptyset}$ variables for all $j \in C$ are the initial columns in RLPM. We solve the node relaxations in the B&B tree by solving a column generation subproblem.

We obtain dual LPM (DLPM) model by defining the dual variables μ_j for the constraints (11), and τ_{ij} for the constraints (12). The column generation algorithm adds the w_{jS} variables with positive reduced cost ($\mu_j - \sum_{i \in S} \tau_{ij} > 0$) to RLPM. Such w_{jS} columns correspond to the violated constraints (17) in DLPM.

Dual LPM (DLPM):

$$\max \sum_{j \in C} \mu_j \tag{16}$$

$$\text{s.t.} \sum_{i \in S} \tau_{ij} \geq \mu_j, \qquad \forall j \in C,\ S \in \varepsilon_j \tag{17}$$

$$\sum_{j \in N(v_i)} \tau_{ij} \leq \gamma_i, \qquad \forall i \in V \tag{18}$$

$$\mu_j \text{ free}, \qquad \forall j \in C, \tag{19}$$

$$\tau_{ij} \text{ free}, \qquad \forall \text{ edges } (i,j). \tag{20}$$

If $\zeta_j = \max\{\mu_j - \sum_{i \in S} \tau_{ij} : S \in \varepsilon_j\} > 0$ for some $j \in C$, then we add the column $\begin{bmatrix} 0 \\ e_j \\ A_l \end{bmatrix}$ for the variable w_{jS}. Here, e_j is an m-column vector, which has a 1 at jth row and 0 otherwise, and A_l is a $(\sum_{i=1}^{n} d(v_i))$-column vector, which has -1 at lth row if lth edge is the edge (i,j) with $i \in S$. Thus, we seek a local codeword S for the node c_j by solving the following subproblem for each $j \in C$:

jth Subproblem (SP_j):

$$\min \sum_{i \in N(c_j)} \tau_{ij} x_i - \mu_j \tag{21}$$

$$\text{s.t.} \sum_{i \in N(c_j)} x_i = 2\ell, \tag{22}$$

$$x_i \in \{0,1\}, \ell \in \mathbb{Z}^+. \tag{23}$$

The binary variable x_i indicates whether $i \in S$ or not. We generate the new columns by solving SP_j to optimality with *ColGen* (see Algorithm 1), which runs in $\mathcal{O}(n \log n)$ time due to the sorting step. We solve LPM to optimality by introducing the columns to RLPM until we have $\zeta_j = 0$ for all $j \in C$.

Algorithm 1: (*ColGen*)

Input τ_{ij} values

1. Sort the τ_{ij} values in nondecreasing order.
   Let τ_{ij}^t be the tth smallest τ_{ij} value.
2. Set $x_i = 0 \ \ \forall i \in N(c_j)$, set $t = 1$.
3. **If** $\tau_{i_1,j}^t + \tau_{i_2,j}^{t+1} < 0$, **Then** set $x_{i_1} = x_{i_2} = 1$, **Else** STOP.
4. $t \leftarrow t + 2$, go to Step 3.

Output SP_j is solved.

3.2.1 Branching in the BP Algorithm

In the case of a fractional optimal solution of LPM, we have either f_i or w_{jS} variables fractional. According to Proposition 2, integral f_i (or integral w_{jS}) values ensure that all the decision variables are integral. That is, branching on either the f_i or w_{jS} variables is sufficient to obtain an integral solution of LPM.

Branching on the f_i variables: We consider the case that the B&B tree branches on the fractional f_i variables in the node solutions of LPM. Since f_i is a binary variable in IPM, we fix $f_i = 0$ in one branch and $f_i = 1$ in the other branch for a fractional f_i. In a branch of the B&B tree, let $f_i = 0$ for $i \in N_0$ and $f_i = 1$ for $i \in N_1$, where $N_0 \cup N_1 \subseteq V$ and $N_0 \cap N_1 = \emptyset$. For such a branch, we have the following subproblem for each $j \in C$:

jth Branch Subproblem (BSP_j):

$$\min \sum_{i \in N(c_j)} \tau_{ij} x_i - \mu_j \tag{21}$$

$$\text{s.t.} \sum_{i \in N(c_j)} x_i = 2\ell, \tag{22}$$

$$x_i = 0, \qquad \text{for } i \in N(c_j) \cap N_0, \tag{24}$$

$$x_i = 1, \qquad \text{for } i \in N(c_j) \cap N_1, \tag{25}$$

$$x_i \in \{0,1\}, \ell \in \mathbb{Z}^+. \tag{23}$$

In order to solve the BSP_j, we eliminate the x_i variables for $i \in N(c_j) \cap N_0$ and we plug in $x_i = 1$ values for $i \in N(c_j) \cap N_1$ to obtain a constant term in the objective (21). We solve the remaining problem to optimality by applying *ColGenBranch* (see Algorithm 2), which is a modified version of *ColGen*. *ColGenBranch* is a polynomial algorithm and runs in $\mathcal{O}(n \log n)$ time.

Algorithm 2: (*ColGenBranch*)

Input: Sets N_0 and N_1, where $f_i = 0$ for $i \in N_0$ and $f_i = 1$ for $i \in N_1$.

0. Set $x_i = 0$, if $i \in N(c_j) \cap N_0$, and $x_i = 1$, if $i \in N(c_j) \cap N_1$.
   Let $\mathcal{I}_j = N(c_j) \setminus (N_0 \cup N_1)$.
1. Sort the τ_{ij} values in nondecreasing order for $i \in \mathcal{I}_j$.
   Let τ_{ij}^t be the tth smallest τ_{ij} value.
2. Set $x_i = 0 \ \ \forall i \in \mathcal{I}_j$, set $t = 1$.
3. **If** $|N(c_j) \cap N_1|$ is even
4. **Then** set $x_{i_1} = x_{i_2} = 1$ if $\tau_{i_1,j}^t + \tau_{i_2,j}^{t+1} < 0$, otherwise STOP.
5. $t \leftarrow t + 2$, go to Step 4.
6. **Else** set $x_i = 1$ for τ_{ij}^t
7. **If** $\tau_{ij}^t < 0$, **Then** $t \leftarrow t + 1$ and go to Step 4, **Else** STOP.
8. **End If**

Output: A local codeword S with objective value
$\zeta_j = \sum_{i \in \mathcal{I}_j} \tau_{ij} x_i + \sum_{i \in N(c_j) \cap N_1} \tau_{ij} - \mu_j$.

We observe that branching on the f_i variables does not change the structure of the subproblems and ensures the integrality of the w_{jS} variables due to Proposition 2. On the other hand, the subproblems cannot be solved in polynomial time when one branches on the w_{jS} variables. Hence, we branch on the closest f_i variable to the fractional value 0.5 in our computational experiments.

3.2.2 Repairing Infeasibility in the Node Relaxations

In the application of our BP algorithm, we observe that a branch can be pruned although there exists a feasible solution on that branch. This may happen if the currently generated columns are not sufficient to construct a feasible solution on the branch. As an example, consider the $f_2 = 1$ and $f_4 = 1$ branch of the TG in Figure 9.

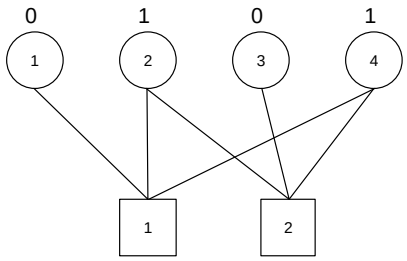

Figure 9: An example TG

The set of all feasible local codewords for the node c_1 is $\varepsilon_1 = \{\emptyset, \{1,2\}, \{1,4\}, \{2,4\}\}$ and for the node c_2 is $\varepsilon_2 = \{\emptyset, \{2,3\}, \{2,4\}, \{3,4\}\}$. On the $f_2 = 1$ and $f_4 = 1$ branch, one can see that (0 1 0 1) is a codeword with local codewords $\{2,4\}$ of c_1 and $\{2,4\}$ of c_2. However, we cannot find this feasible solution on the branch if we have only generated the local codewords $\emptyset, \{1,2\}$ and $\{1,4\}$ for c_1, and the local codeword $\emptyset$ for c_2. Moreover, we cannot find any other feasible solution on this branch with these limited number of local codewords.

In such a case, the $f_2 = 1$ and $f_4 = 1$ branch is pruned by infeasibility although there is a feasible solution for IPM on the branch. In order to overcome this situation, we developed a column generation method based on the dual formulation. Let P be the primal problem representing RLPM, and D be the dual of RLPM.

Proposition 5. *P is infeasible if and only if D is unbounded.*

Proof. From the duality theory, we know that infeasible P implies D is unbounded or infeasible. We know that LPM is bounded since the variables f_i and $w_{jS} \in [0,1]$ and it is feasible since the **0**-codeword is a trivial solution. Then the dual of LPM is also feasible.

D (the dual of RLPM) is feasible since it contains the feasible region defined by LPM dual. This means P is infeasible $\implies$ D is unbounded. Moreover, unbounded D implies P is infeasible from the duality theory. As a result, we conclude that P is infeasible $\iff$ D is unbounded. □

At an infeasible branch, either the current P is permanently infeasible or it temporarily appears to be infeasible since we have not yet generated the columns that are necessary to construct a feasible solution. Farkas' lemma states either $\begin{bmatrix}\mathbf{f} & \mathbf{w}\end{bmatrix}\mathbf{A} = \mathbf{c}$ and $\mathbf{f} \geq \mathbf{0}, \mathbf{w} \geq \mathbf{0}$ is feasible or there is a ray $\mathbf{d}$ with $\mathbf{Ad} \leq \mathbf{0}$ and $\mathbf{cd} > 0$, where $\mathbf{A}$ is the coefficient matrix for the constraints and $\mathbf{c} = \begin{bmatrix}\mathbf{1} & \mathbf{0}\end{bmatrix}$ is the right-hand-side vector of P. In case P is infeasible, we aim to add a variable w_{jS} to P and the corresponding coefficient column $\mathbf{a}^T$ satisfying $\mathbf{ad} > 0$ to $\mathbf{A}$ matrix to fulfill feasibility. This corresponds to finding a dual constraint (dashed line) that can bound the unbounded objective function value of D as shown in Figure 10. An LP solver can provide a dual ray $\mathbf{d}$ in which the dual objective $\mathbf{cd}$ is unbounded when P is infeasible.

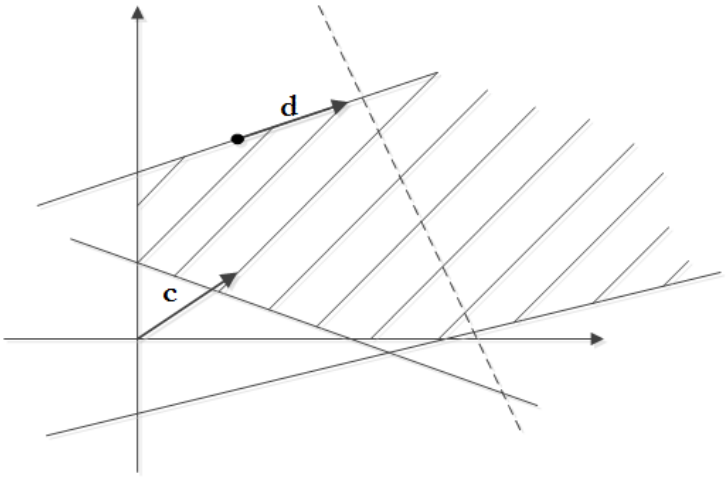

Figure 10: Dual constraint generation

Note that there can be multiple dual constraints (dashed line in Figure 10) that can bound the dual objective. Hence, we search for the vector $\mathbf{a}$ which has the largest positive $\mathbf{ad}$ value. Here, $\mathbf{a}$ is the coefficient vector of the desired dual constraint $\mu_j - \sum_{i \in S} \tau_{ij} \leq 0$ for some $j \in C$ and $S \in \varepsilon_j$ among the constraints (17). Adding the columns to P using the dual ray obtained by solving the dual Farkas system is known as Farkas pricing in the literature (Lübbecke and Desrosiers, 2005; Lübbecke, 2010).

The vector $\mathbf{a}$ has $(m + \eta)$-many entries where m is the number of check nodes and η is the number of edges in TG. The first m entries of $\mathbf{a}$ vector are the coefficients of $\boldsymbol{\mu}$ variables, and jth entry is the only nonzero one. The following η entries are the coefficients of $\boldsymbol{\tau}$ variables, and they are all zero except for the -1 entries corresponding to the check nodes $j \in C$ and $i \in S$. We can decompose the ray $\mathbf{d}$ as $\begin{bmatrix}\mathbf{d}^{\boldsymbol{\mu}} \\ \mathbf{d}^{\boldsymbol{\tau}}\end{bmatrix}$, where $\mathbf{d}^{\boldsymbol{\mu}}$ and $\mathbf{d}^{\boldsymbol{\tau}}$ are the entries of $\mathbf{d}$ corresponding to

the indices of the variables $\boldsymbol{\mu}$ and $\boldsymbol{\tau}$, respectively. We have $\mathbf{ad} = \mathbf{d}^{\boldsymbol{\mu}} - \sum_{i \in S} \mathbf{d}^{\boldsymbol{\tau}}$, and maximizing $\mathbf{ad}$ is equivalent to maximizing $d_j^{\boldsymbol{\mu}} - \sum_{i \in S} d_{ij}^{\boldsymbol{\tau}}$ for each check node c_j. Hence, we have to solve the following direction subproblem for each $j \in C$:

jth Direction Subproblem (DSP_j):

$$\min \sum_{i \in N(c_j)} d_{ij}^{\boldsymbol{\tau}} x_i - d_j^{\boldsymbol{\mu}} \tag{26}$$

$$\text{s.t.} \sum_{i \in N(c_j)} x_i = 2\ell, \tag{22}$$

$$x_i \in \{0,1\}, \ell \in \mathbb{Z}^+. \tag{23}$$

We observe that DSP_j and SP_j are the same problem except the objective function coefficients. Hence, we can solve DSP_j in $\mathcal{O}(n \log n)$ time with *ColGenBranch* after replacing τ_{ij} and μ_j with $d_{ij}^{\boldsymbol{\tau}}$ and $d_j^{\boldsymbol{\mu}}$, respectively. We summarize our dual constraint (primal column) generation method with *DualConsGen* (see Algorithm 3).

Algorithm 3: (*DualConsGen*)

Input: An infeasible restricted primal problem, P

0. $isFeasible \leftarrow true$
1. Solve the dual Farkas system and obtain a dual ray $\mathbf{d}$ that D is unbounded.
2. Solve DSP_j for each check node j. Add generated local codewords, i.e., columns, to P.
3. **If** no columns generated, **Then** conclude P is infeasible. $isFeasible \leftarrow false$ and STOP.
4. Solve problem P.
5. **If** P is feasible, **Then** STOP.
6. **Else** go to Step 1.
7. **End If**

Output: $isFeasible$

3.3 Improvements to the BP Algorithm

SolveIPM (see Algorithm 4) shows the main steps of our BP algorithm. We utilize a new pruning rule (see Section 3.3.1) to improve the performance of BP in terms of the solution quality and computation time. (RS) and (C) steps are our improvements to BP. We name the improved version of BP with (RS) step as the BPRS method, and with the (RS) and (C) steps as the BCP method. In the (RS) step, we generate an initial feasible solution with the *RandSum* heuristic (see Algorithm 5 in Section 3.3.2). In the (C) step, the *FracSep* algorithm adds valid cuts to separate the fractional solutions of RLPM (see Algorithm 6 in Section 3.3.3).

3.3.1 A Pruning Strategy

There are three rules to prune a branch of a B&B tree: optimality, infeasibility, and value dominance. In this section, we introduce a new pruning rule which is based on the difference between the objective function values of two feasible integral solutions.

Proposition 6. *Let* $\mathbf{f}$ *be a feasible integral solution of LPM with the objective function value* z*. Then, there is no feasible integral solution of LPM with objective function value in the range* $(z-a, z)$ *with the log-likelihood objective (9), where* $a = \log[(1-p)/p]$*.*

Proof. Since $\mathbf{f}$ is integral, the value z of the objective (9) is an integral multiple of a, i.e., $z = \delta a$ and $\delta \in \mathbb{Z}$. Let $\mathbf{f}'$ be another integral feasible solution of LPM. Then, its objective value z' is also an integral multiple of a, say $z' = \delta' a$ and $\delta' \in \mathbb{Z}$. The difference between the objectives is $z - z' = (\delta - \delta')a$. From here, we conclude that the nearest objective function value to z can be either $z' = z + a$ or $z' = z - a$. Hence, there is no feasible integral solution of LPM with objective function value in the range $(z-a, z)$. □

In other words, the minimum difference between two feasible integral solutions is a with the log-likelihood objective (9) and 1 with the Hamming distance objective (2).

Proposition 7. *Let* z *be the optimum objective value of RLPM at a branch. The branch can be pruned if* $z > z_{UB} - a$*, where* z_{UB} *is the best upper bound on IPM and* a *is the minimum difference between two feasible integral solutions.*

Proof. A branch can be pruned by value dominance if $z > z_{UB}$. Besides, as shown in Proposition 6, there cannot be an integral feasible solution in the range $(z_{UB} - a, z_{UB})$. Hence, we can prune the branch if $z > z_{UB} - a$. □

Algorithm 4: (*SolveIPM*)

Input: A set of feasible local codewords that constitutes $RLPM$ ($\emptyset \in \varepsilon_j,\ \forall j$).

0. Set $LIST = \{RLPM\}$, let $\bar{z} = \infty$ and $\underline{z} = -\infty$.
(RS). Apply RandSum to generate an initial feasible solution.
1. **While** $LIST \neq \emptyset$ **Do**
2. Select the last problem in $LIST$, say problem P.
3. Solve P to optimality by adding the columns with *ColGenBranch*.
Obtain the optimal primal $(\mathbf{f}^*, \mathbf{w}^*)$ and the dual $(\boldsymbol{\mu}^*, \boldsymbol{\tau}^*)$ solutions with value $\underline{z}^i$.
Pruning / delete P from the LIST */*
4. **If** P is infeasible, **Then** prune by infeasibility **if** *DualConsGen* returns $false$.
Go to Step 1.
5. **End If**
6. **If** $\underline{z}^i \geq \bar{z}$, **Then** prune by bound and go to Step 1.
7. **If** P has an integer optimal solution, **Then** $\bar{z} = \underline{z}^i$,
prune by optimality and go to Step 1.
8. **End If**
Branching / add P to the LIST */*
9. **If** P has a fractional optimal solution,
Then choose the closest fractional f_i to 0.5
Left Branch
10. Let $RLPM_0 = P \cap \{(\mathbf{f}, \mathbf{w}) : f_i = 0\}$, add $x_i = 0$ to subproblem j, if $i \in N(c_j)$.
Add $RLPM_0$ to $LIST$.
Right Branch
11. Let $RLPM_1 = P \cap \{(\mathbf{f}, \mathbf{w}) : f_i = 1\}$, add $x_i = 1$ to subproblem j, if $i \in N(c_j)$.
Add $RLPM_1$ to $LIST$.
12. Go to Step 1.
(C). Apply FracSep for adding cuts (27) to RLPM.
13. **End If**
14. **End While**

Output: An integral solution $(\mathbf{f}^*, \mathbf{w}^*)$ to IPM with objective value $\bar{z}$.

3.3.2 Random Sum Heuristic

As we explained in Section 2, each row of $\mathbf{G}$ is a codeword (feasible solution). We can rewrite a parity-check $\mathbf{H}$ matrix as $\mathbf{H} = [\mathbf{A}|\mathbf{I_{n-k}}]$ by carrying out elementary row operations under binary arithmetic. Here, $\mathbf{A}$ is a $(n-k) \times k$ binary matrix, and $\mathbf{I_{n-k}}$ is the $(n-k) \times (n-k)$ identity matrix. Then a $k \times n$ generator matrix $\mathbf{G} = [\mathbf{I_k}|\mathbf{A^T}]$ can be obtained using this $\mathbf{A}$ matrix. Note that, one can obtain different $\mathbf{A}$ matrices and the generator matrix $\mathbf{G}$ is not unique.

Since $\mathbf{G}$ is a basis for the solution space of EM, any feasible solution can be written as a binary combination $\mathbf{u'}$ of the rows of $\mathbf{G}$. There are 2^k different $\mathbf{u'}$ combinations, where k is the number of the rows of $\mathbf{G}$. We try t_{max} random $\mathbf{u'}$ combinations and update the upper bound with the best solution found as given in the Random Sum (*RandSum*) heuristic (see Algorithm 5).

Algorithm 5: (*RandSum*)

Input: A generator matrix $\mathbf{G}$

0. Initialize $\mathbf{z}^* = \infty, \mathbf{e}^*, t_{max}$.
1. **While** $t < t_{max}$
2. Randomly set u'_i from {0, 1} for $i = 1, ..., k$.
3. Obtain a feasible solution by $\mathbf{e} = \mathbf{u}'\mathbf{G}$.
4. Calculate the objective function value $\mathbf{z_e}$ of solution $\mathbf{e}$.
5. **If** $\mathbf{z_e} < \mathbf{z}^*$, **Then**
6. $\mathbf{z}^* = \mathbf{z_e}$, $\mathbf{e}^* = \mathbf{e}$
7. **End If**
8. t = t+1
9. **End While**

Output: A codeword $\mathbf{e}^*$ with objective value $\mathbf{z}^*$.

The BPRS method implements the *RandSum* heuristic in (*RS*) step of *SolveIPM* in order to tighten the upper bound. Moreover, we add the columns corresponding to the best known solution to RLPM.

3.3.3 Fractional Solution Separation

As given in Feldman et al. (2005), for a check node c_j and all $S \subseteq N(c_j)$ with $|S|$ odd, the following inequalities are valid for IPM:

$$\sum_{i \in N(c_j) \setminus S} f_i + \sum_{i \in S} (1 - f_i) \geq 1. \tag{27}$$

Since there are m check nodes in a TG and $2^{d(c_j)-1}$ many odd cardinality $S \subseteq N(c_j)$, there are exponential-many valid cuts (27), i.e., $m2^{d(c_j)-1}$. We can trim a fractional solution of RLPM, which is found by BP, if it violates the inequality (27) for some check node c_j and odd cardinality $S \subseteq N(c_j)$. We propose the *FracSep* algorithm (see Algorithm 6) that searches for such violated inequalities to separate a given fractional solution. Notice that the valid cuts (27) do not include w_{jS} variables. Hence, the formulations of SP_j, BSP_j, and DSP_j are not affected by the valid cuts (27), which allows us to use the same solution methods for the subproblems. *FracSep* is an exact separation algorithm, since it generates all violated inequalities (27) by a given fractional solution. *FracSep* runs in $\mathcal{O}(n \log n)$ time due to the sorting step. The BCP method introduces the valid cuts (27) in (*C*) step of *SolveIPM*.

Algorithm 6: (*FracSep*)

Input: A fractional solution $\mathbf{f}$ of $RLPM$

1. Sort $\mathbf{f}$ values in nonincreasing order. Let $\mathcal{I}_s$ be sorted indices.
2. **For Each** check node c_j and odd cardinality $\omega \leq d(c_j)$
3. Construct $S \subseteq N(c_j)$ using first ω neighbors of c_j in $\mathcal{I}_s$.
4. **If** inequality (27) is violated by $\mathbf{f}$ with S,
5. **Then** add cut (27) to $RLPM$.
6. **End If**
7. **End For Each**

Output: Cuts added to $RLPM$.

4 Computational Results

The computations have been carried out on a computer with 2.6 GHz Intel Core i7-9750H processor and 16 GB of RAM working under Windows 10 Professional. We compare the performances of the methods summarized in Table 3. We implemented all methods in C# programming language. In the BP method, we apply our branch-and-price algorithm for IPM without any improvements (see Section 3.2). The BPRS method is an extension of the BP method with the *RandSum* heuristic, and implements the (RS) step in *SolveIPM*. The BCP method is our branch-cut-and-price algorithm, which utilizes the (RS) step and adds the valid cuts to RLPM with the (C) step in *SolveIPM*. We solve the RLPM models at the nodes of the B&B tree with CPLEX 12.10. We compare the performance of our BP, BPRS, and BCP decoders with two state-of-the-art decoders from the literature: an iterative heuristic decoder SP and the benchmark ML decoder EMD, which is the IP decoder counterpart of the LP decoder of Tanatmis et al. (2010) (see Section 3.1). In EMD, we solve the the EM formulation with CPLEX 12.10 by adding the valid cuts (27) with the separation algorithm of Tanatmis et al. (2010).

Table 3: Summary of the methods

Method	Formulation	(RS)	(C)
BP	IPM	–	–
BPRS	IPM	√	–
BCP	IPM	√	√
EMD	EM	–	√
SP	–	–	–

A summary of the parameters used in the computational experiments is given in Table 4. We try eight different code lengths from $n = 300$ to $n = 8400$ for three error probability p levels. We randomly construct (5, 10)-regular $\mathbf{H}$ permutation code for each n (see Section 2). We ensure that the length of the smallest cycle in the TGs of the generated codes is at least 6. We test the

quality of the upper bounds obtained by the *RandSum* heuristic with two different t_{max} values. In order to speed up the row sums and the objective function calculation, we utilize *BitArray* data structure in *RandSum*. We set a time limit of 600 seconds for all methods in Table 3.

Table 4: List of the computational parameters

Parameters	
n	300, 600, 1200, 1800, 2400, 3000, 6000, 8400
p	0.05, 0.07, 0.10
t_{max}	1000, 10000
Time Limit	600 secs

In our first experiment, we try $t_{max} = 1000$ and 10000 to observe the quality of the upper bound obtained by the *RandSum* heuristic (see Section 3.3.2). In Table 5, we report the average results for 30 trials for each n. We generate an original codeword by randomly combining the rows of $\mathbf{G}$. According to the results, we can obtain closer codewords to the original codeword, i.e., the BER values are smaller, when t_{max} gets higher. Hence, we prefer to take $t_{max} = 10000$ in our BPRS and BCP methods.

Table 5: Performance of the *RandSum* heuristic

t_{max}	1000			10000		
n	z	BER ($\times 10^{-2}$)	CPU (secs)	z	BER ($\times 10^{-2}$)	CPU (secs)
300	115.8	39.4	0.01	78.7	22.8	0.08
600	251.6	43.3	0.02	161.7	24.4	0.17
1200	522.0	44.8	0.05	332.3	24.7	0.46
1800	805.6	45.6	0.08	502.3	24.5	0.80
2400	1093.2	46.5	0.13	667.0	24.5	1.27
3000	1374.3	46.3	0.19	846.4	25.0	1.84
6000	2819.4	47.4	0.66	1715.0	25.2	6.04
8400	4004.0	48.2	1.22	2411.6	25.2	11.79

We experiment to compare the error correction capabilities of the BP, BPRS, BCP and EMD methods. For each code length n, we run the decoders on 10 random received vectors (in total 240 instances for three p levels) and report the average values. We give the computational results of the BP and EMD methods in Table 6 and the BPRS and BCP methods in Table 7. The column "z_l" gives the best lower bound found by the method, and the column "z" is the objective value of the best known solution. Although the objective functions of EM and IPM are equivalent (see Proposition 1), their values are different. Hence, we report the Hamming distance objective value for the solutions found by BP, BPRS, and BCP. We report the percentage difference among z_l and z in "Gap." The number of instances that are solved to

optimality (i.e., $z_l = z$) is given in the column "#Opt" and the number of nodes evaluated in the B&B tree is reported in "#Nodes."

EMD adds the valid inequalities (27) by implementing the separation algorithm of Tanatmis et al. (2010). The "#Cuts" column gives the number of cuts generated in EMD. BP decoder can use the trivial solution of $f_i = 0$ for all $i \in V$ as an initial upper bound (see Section 3.1). The column "z_u^i" shows the objective value of the initial solution.

We observe that the BP method uses fewer nodes than EMD. As the code length n increases, the number of nodes that both methods can evaluate decreases due to time limitation. When $p = 0.05$, BP is worse than EMD in terms of gap, BER and CPU. For all p values, BP cannot complete the evaluation of the root node for $n = 6000$ and 8400 within the time limit to provide a lower bound (i.e., $z_l = 0$). EMD solves 90 instances to optimality whereas BP finds the optimal solution of 68 among 240 instances.

Table 6: Performances of EMD and BP

		EMD								BP							
				Gap	BER	CPU	#						Gap	BER	CPU	#	
p	n	z_l	z	(%)	($\times 10^{-2}$)	(secs)	Opt	Nodes	Cuts	z_l	z	z_u^i	(%)	($\times 10^{-2}$)	(secs)	Opt	Nodes
0.05	300	14.0	14.0	0	0	0.54	10	0.7	131.4	14.0	14.0	153.4	0	0	3.42	10	0
	600	28.7	28.7	0	0	79.33	10	16.1	953.9	28.2	28.7	295.7	1.4	0	27.56	8	4.4
	1200	57.3	57.3	0	0	4.58	10	2.2	565.1	57.2	57.3	599.9	0.1	0	122.04	9	0.4
	1800	87.0	87.0	0	0	1.40	10	0	747.7	77.9	166.6	907.2	10.0	5.0	153.44	9	0
	2400	117.7	117.7	0	0	2.18	10	0	1053.5	105.5	224.9	1198.7	10.0	5.0	177.95	9	0
	3000	147.3	147.3	0	0	2.32	10	0	1221.6	103.5	540.1	1452.7	30.0	14.5	275.13	7	0
	6000	301.7	301.7	0	0	66.52	10	0	3108.6	0	2970.9	2972.5	100	50.4	*time*	0	0
	8400	425.9	425.9	0	0	12.74	10	0	3966.9	0	4199.7	4206.5	100	49.9	*time*	0	0
0.07	300	18.9	68.4	40.8	19.6	350.94	5	146329.3	2372.1	17.0	58.2	153.2	30.6	0	312.45	4	838.1
	600	38.1	135.0	41.3	24.3	383.58	5	76456.9	3044.8	34.8	191.5	296.3	54.4	4.7	428.36	2	92.3
	1200	77.1	346.3	77.7	39.4	*time*	0	41630.2	4180.4	70.8	290.4	604.2	40.6	5.0	542.42	1	16.6
	1800	114.7	534.2	78.5	41.0	*time*	0	26690.6	4431.9	107.4	275.4	906.6	25.9	5.0	255.07	0	1.3
	2400	156.7	712.8	78.0	41.9	*time*	0	20069.0	3678.1	156.6	368.0	1199.4	20.1	10.0	452.45	4	0
	3000	197.7	893.0	77.9	42.3	*time*	0	12049.4	4603.9	141.2	581.9	1456.5	30.9	14.5	411.76	4	0
	6000	397.0	1766.7	77.5	49.5	*time*	0	245.8	7177.2	0	2970.9	2970.9	100	49.5	*time*	0	0
	8400	556.0	2507.4	77.8	49.9	*time*	0	9.7	8595.4	0	4200.6	4211.8	100	49.9	*time*	0	0
0.10	300	23.6	85.4	72.3	39.3	*time*	0	344196.6	3636.0	19.6	152.0	153.6	87.1	48.7	*time*	0	1405.6
	600	44.5	174.6	74.5	39.2	*time*	0	129500.2	3057.1	38.9	294.5	294.5	86.8	49.0	*time*	0	111.8
	1200	86.4	351.6	75.4	39.8	*time*	0	48311.8	2941.5	77.4	552.6	601.4	81.8	22.7	*time*	0	16.1
	1800	130.1	528.9	75.4	39.7	*time*	0	25975.4	3865.5	124.3	682.0	901.8	69.7	22.6	*time*	0	1.0
	2400	172.7	713.2	75.8	40.0	*time*	0	19352.6	3988.0	185.7	621.6	1197.0	48.3	15.0	*time*	0	0
	3000	215.7	881.8	75.5	39.3	*time*	0	9691.8	4424.6	161.8	758.9	1457.3	44.7	24.1	597.50	1	0
	6000	430.9	1778.5	75.8	39.4	*time*	0	195.9	6505.4	0	2961.5	2961.5	100	49.5	*time*	0	0
	8400	602.8	2500.0	75.9	39.7	*time*	0	6.0	7884.2	0	4190.6	4202.4	100	49.9	*time*	0	0

The BPRS and BCP methods utilize an initial solution with the *RandSum* heuristic (see Table 7). We report the number of valid cuts (27) used by the BCP method in the column "#Cuts." BCP can find the original codeword (i.e., BER = 0) for all instances by improving the solution of *RandSum* with the BP algorithm. The number of cases out of 240 instances solved to optimality for BPRS and BCP are 110 and 171, respectively. As we improve our BP algorithm to BPRS and BCP, we observe better gap, BER, #Opt values and fewer nodes. Moreover, BCP

gives better figures for these performance metrics compared to EMD while generating fewer cuts (27) than EMD.

Table 7 shows that our algorithms can solve more instances to optimality as the code length n increases for $p = 0.07$ and 0.10, which is somewhat counter intuitive. For example, BCP can solve 10 instances to optimality when $n \geq 6000$ for these p values. This is due to the properties of **H** codes and the LPM formulation. In particular, randomly constructed permutation codes have fewer cycles in their TG representations as the dimension of the code gets larger (Lau et al., 2017). When a TG is cycle-free, any optimal solution of LPM is integral as noted in Feldman et al. (2005). Hence, as the code length n increases, we have a TG with fewer cycles, which results in a better LP lower bound at the root node. This is not apparent for BP since we cannot complete the root node evaluation due to time limit for $n \geq 6000$.

Table 7: Performances of BPRS and BCP

		BPRS								BCP								
					Gap	BER	CPU	#					Gap	BER	CPU	#		
p	n	z_l	z	z_u^i	(%)	($\times 10^{-2}$)	(secs)	Opt	Nodes	z_l	z	z_u^i	(%)	($\times 10^{-2}$)	(secs)	Opt	Nodes	Cuts
0.05	300	14.0	14.0	76.0	0	0	1.57	10	0	14.0	14.0	76.0	0	0	0.74	10	0	63.2
	600	28.2	28.7	151.9	1.4	0	64.74	8	71.6	28.7	28.7	151.9	0	0	64.90	10	10.2	7.6
	1200	57.2	57.3	326.7	0.1	0	35.65	9	1.2	57.3	57.3	326.7	0	0	32.81	10	1.7	11.9
	1800	87.0	87.0	481.2	0	0	107.05	10	0	87.0	87.0	481.2	0	0	135.45	10	0	14.6
	2400	117.7	117.7	643.9	0	0	303.86	10	0	117.7	117.7	643.9	0	0	340.03	10	0	36.2
	3000	147.3	212.9	813.3	8.1	0	502.74	9	0	147.3	147.3	813.3	0	0	517.13	10	0	53.5
	6000	149.2	970.7	1638.1	40.0	12.4	*time*	6	0	301.7	301.7	1638.1	0	0	596.00	10	0	56.7
	8400	296.6	994.1	2316.7	30.0	7.5	*time*	7	0	425.9	425.9	2316.7	0	0	567.00	10	0	13.0
0.07	300	17.2	19.4	78.4	9.6	0	360.86	5	1759.8	18.9	19.4	78.4	2.1	0	303.79	6	25.1	7.6
	600	34.8	84.9	157.7	34.2	9.6	426.39	2	915.1	38.7	39.3	157.7	1.4	0	507.20	6	11.3	193.0
	1200	71.1	156.6	335.8	29.7	0	545.00	2	124.2	78.0	79.0	335.8	1.2	0	533.30	5	9.7	715.9
	1800	110.2	196.2	497.8	22.7	2.4	565.00	2	32.8	117.7	120.8	497.8	2.5	0	559.00	5	9.7	813.2
	2400	156.1	212.0	666.2	11.0	2.5	507.40	4	9.3	161.2	162.2	666.2	0.6	0	538.00	6	7.6	453.0
	3000	198.0	265.5	839.5	9.9	2.5	510.00	5	1.9	203.7	204.1	839.5	0.2	0	527.10	9	7.6	422.6
	6000	207.2	1060.1	1698.3	50.0	12.4	582.00	5	0	420.1	420.1	1698.3	0	0	573.90	10	4.9	62.3
	8400	291.4	1492.6	2398.2	40.0	12.7	599.80	6	0	591.2	591.2	2398.2	0	0	586.30	10	4.6	2167.0
0.10	300	19.6	60.8	82.2	58.1	14.9	*time*	0	2836.1	29.0	30.6	82.2	4.9	0	582.90	4	9.1	58.9
	600	38.9	122.9	168.3	58.6	14.2	*time*	0	1361.2	57.2	59.3	168.3	3.5	0	574.70	3	8.8	92.2
	1200	77.4	258.6	351.8	60.3	15.5	*time*	0	110.9	110.4	117.6	351.8	6.0	0	594.10	1	7.0	171.5
	1800	124.3	352.0	523.4	51.7	12.5	*time*	0	30.6	171.6	177.2	523.4	3.1	0	584.80	2	7.0	135.7
	2400	187.9	281.8	699.2	24.2	14.8	*time*	0	8.1	226.1	234.8	699.2	3.7	0	*time*	0	7.0	138.2
	3000	243.6	469.1	879.3	31.2	7.5	598.00	1	1.4	287.0	294.9	879.3	2.7	0	595.90	4	8.1	309.8
	6000	417.3	956.7	1784.9	30.5	7.5	578.60	5	0	597.0	597.0	1784.9	0	0	594.80	10	6.3	61.4
	8400	425.2	1853.0	2521.2	56.8	15.2	550.20	4	0	842.0	842.0	2521.2	0	0	599.30	10	6.0	81.9

Figure 11 illustrates the decoding latency and the optimality gap values of the BP, BPRS, BCP and EMD methods in our experiments. In particular, Figure 11a shows that the average decoding time per bit is almost the same for all methods at each error probability level. According to Figure 11b, the average optimality gap increases significantly as the probability p gets higher for EMD. We observe that, the average optimality gap gradually improved from BP to BPRS and BCP achieves the best optimality gap.

SP is a commonly used iterative message-passing algorithm to decode the received vector

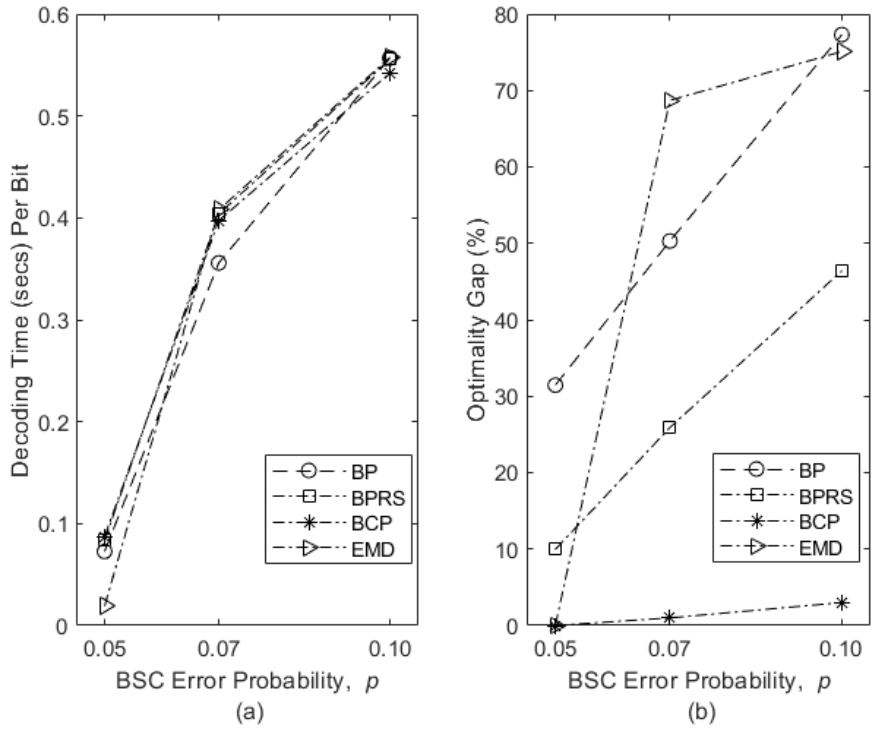

Figure 11: Decoding time and optimality gap comparison of the methods

in practical applications. At each iteration of SP, each variable node in the TG sends the probability of being 0 or 1 as a message to the adjacent check nodes. A variable node decides on the value of the bit after receiving the messages from its neighboring check nodes. It is well known in the literature that the iterative decoding algorithms (such as SP) may fail to decode if the TG includes cycles (Kschischang et al., 2001). On the other hand, the optimization-based decoders BP, BPRS, BCP, and the benchmark EMD can complete decoding independent from the existence of cycles. Note that, as there are fewer cycles in the TG, the decoding latency of an optimization-based decoder improves, since the LP relaxation gets tighter.

4.1 Simulation Results

We further compare the performance of our BCP decoder to the state-of-the-art decoders EMD and SP via BER simulations. Figure 12 shows the BER values of the decoders for five different channel error probabilities, i.e., $p = 10^{-1}, 10^{-1.2}, 10^{-1.4}, 10^{-1.6}$, and $10^{-1.8}$. For each p value, we randomly generate 10 different $(5, 10)$-regular LDPC codes with $n = 300$, and plot the average BER of these 10 trials. For each LDPC code, we continue with the decoding of randomly generated received vectors until we observe 50 decoded vectors different from the original codeword. For example, the average BER of 10 different LDPC codes is around 10^{-5} when $p = 10^{-1.2}$ for

our BCP decoder in Figure 12.

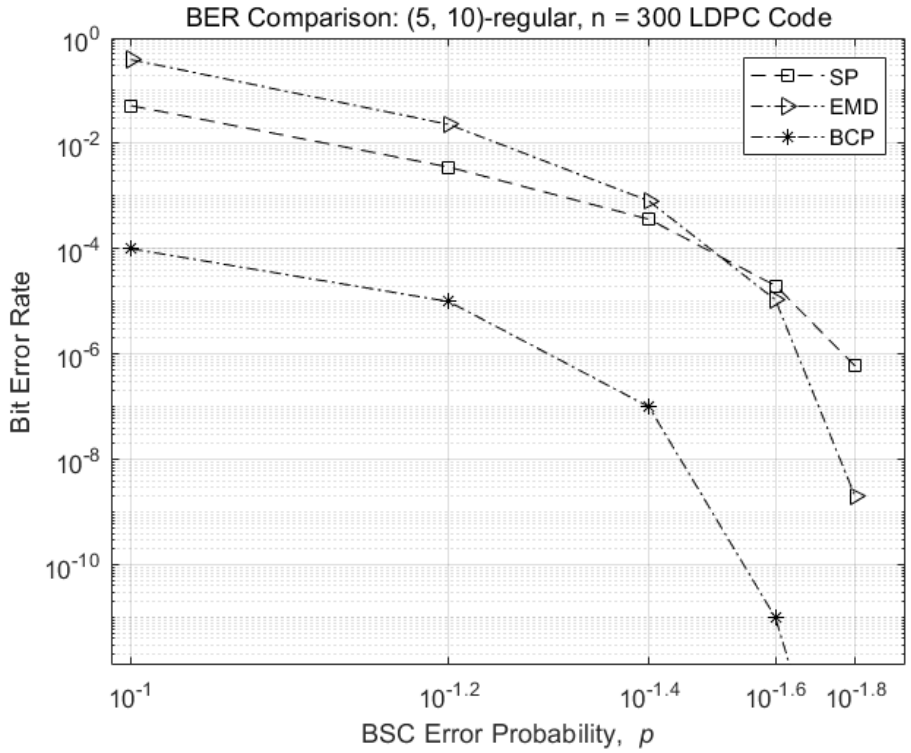

Figure 12: BER simulations of SP, EMD, and BCP

Figure 12 reveals that EMD has smaller BER figures than SP when $p < 10^{-1.6}$, whereas our BCP decoder has the best BER figures for all p values. As we mentioned earlier, EMD and our decoders (BP, BPRS and BCP) are optimization-based decoders, and their BER achievements are independent of the cycles. On the other hand, being an iterative message-passing algorithm, SP can be adversely affected by the existence of cycles. In order to compare the performances of SP and the optimization-based decoders, we facilitate TGs having the length of the smallest cycle at least 6. Figure 12 shows that the BER performance of the BCP decoder is significantly superior compared to the SP algorithm.

The polar codes, which is a dense code family, of Arıkan are currently in the 5G communication standards (Arikan, 2008). In this study, we experiment on the dense $(5, 10)$-regular codes, and observe that our BCP method can contribute to the new telecommunication standards with its impressive BER performance.

Furthermore, BCP is implementable for the cases such as deep space communications, that we cannot reobtain the information from the digital source. For such applications, high quality decoding is important instead of decoding speed. Our BCP method is a candidate decoder thanks to its high decoding quality for such communication systems.

5 Conclusions

In this study, we focus on the decoding algorithms that correct the errors in the received vector using LDPC codes for digital communication systems. We consider a mathematical formulation from the literature and propose a branch-and-price (BP) algorithm for its solution. We improve the error correction capability of our BP algorithm by providing tight upper bounds with the *RandSum* heuristic and introducing valid cuts to the mathematical formulation. These enhancements give rise to our branch-and-price-random-sum (BPRS) and branch-cut-and-price (BCP) methods, respectively.

Our computational experiments show that our BCP method outperforms exact model decoder (EMD), which makes use of the commercial solver CPLEX 12.10, in terms of gap, BER and the number of instances solved to optimality. According to the BER simulations of BCP, and the state-of-the-art EMD and SP, we observe that BCP has the highest error correction capability with acceptable decoding latency.

Our BCP decoder can contribute to the construction of reliable digital communication systems with its high BER achievement. In particular, BCP can be used for the critical applications, such as NASA's Mission Cassini, in which we receive the information only once. In such settings, solution quality is crucial instead of decoding latency. Considering decoding is an online problem, faster decoders are desired. Hence, improving the solution time of the BCP method can be a future track of research.

Acknowledgments

This research has been supported by the Turkish Scientific and Technological Research Council with grant no 113M499.

References

Arikan, E. (2008). A performance comparison of polar codes and reed-muller codes. *IEEE Communications Letters*, 12(6):447–449.

Barman, S., Liu, X., Draper, S. C., and Recht, B. (2013). Decomposition methods for large scale LP decoding. *IEEE Transactions on Information Theory*, 59(12):7870–7886.

Berlekamp, E., McEliece, R., and Van Tilborg, H. (1978). On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386.

Burshtein, D. (2009). Iterative approximate linear programming decoding of LDPC codes with linear complexity. *IEEE Transactions on Information Theory*, 55(11):4835–4859.

Chertkov, M. and Stepanov, M. G. (2008). An efficient pseudocodeword search algorithm for linear programming decoding of LDPC codes. *IEEE Transactions on Information Theory*, 54(4):1514–1520.

Feldman, J., Wainwright, M. J., and Karger, D. R. (2005). Using linear programming to decode binary linear codes. *IEEE Transactions on Information Theory*, 51(3):954–972.

Fossorier, M. P., Mihaljevic, M., and Imai, H. (1999). Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE Transactions on Communications*, 47(5):673–680.

Gallager, R. (1962). Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28.

Hu, X.-Y., Eleftheriou, E., Arnold, D.-M., and Dholakia, A. (2001). Efficient implementations of the sum-product algorithm for decoding LDPC codes. In *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270)*, volume 2, pages 1036–1036E. IEEE.

Karger, D. R., Oh, S., and Shah, D. (2014). Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24.

Kschischang, F. R., Frey, B. J., Loeliger, H.-A., et al. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

Lau, F. C., Mo, F., Tarn, W. M., and Sham, C.-W. (2017). Random-permutation-matrix-based cyclically-coupled LDPC codes. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 497–500. IEEE.

Leiner, B. M. (2005). LDPC codes–a brief tutorial. *Apr*, 8:1–9.

Lübbecke, M. E. (2010). Column generation. In *Wiley Encyclopedia of Operations Research and Management Science*. Wiley.

Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, 53(6):1007–1023.

MacKay, D. J. (1999). Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431.

MacKay, D. J. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

MacKay, D. J. and Neal, R. M. (1997). Near shannon limit performance of low density parity check codes. *Electronics Letters*, 33(6):457–458.

Moon, T. K. (2005). *Error Correction Coding: Mathematical Methods and Algorithms*. John Wiley & Sons Ltd.

Richardson, T. J. and Urbanke, R. L. (2001). The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618.

Sarajlić, M., Liu, L., and Edfors, O. (2014). Reducing the complexity of LDPC decoding algorithms: An optimization-oriented approach. In *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, pages 861–866. IEEE.

Tanatmis, A., Ruzika, S., Hamacher, H. W., Punekar, M., Kienle, F., and Wehn, N. (2010). A separation algorithm for improved LP-decoding of linear block codes. *IEEE Transactions on Information Theory*, 56(7):3277–3289.

Tanner, R. (1981). A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547.

Vontobel, P. O. and Koetter, R. (2007). On low-complexity linear-programming decoding of LDPC codes. *European Transactions on Telecommunications*, 18(5):509–517.

Yang, K., Feldman, J., and Wang, X. (2006). Nonlinear programming approaches to decoding low-density parity-check codes. *IEEE Journal on Selected Areas in Communications*, 24(8):1603–1613.

Yang, K., Wang, X., and Feldman, J. (2008). A new linear programming approach to decoding linear block codes. *IEEE Transactions on Information Theory*, 54(3):1061–1072.

Zhang, X. and Siegel, P. H. (2012). Adaptive cut generation algorithm for improved linear programming decoding of binary linear codes. *IEEE Transactions on Information Theory*, 58(10):6581–6594.

Zhang, X. and Siegel, P. H. (2013). Efficient iterative LP decoding of LDPC codes with alternating direction method of multipliers. In *2013 IEEE International Symposium on Information Theory*, pages 1501–1505. IEEE.