Decoding Error Correcting codes: An Optimization-based approach

Rajalaxmi Rajagopalan
rr30@illinois.edu

Adi Pasic
pasic2@illinois.edu

Akshay Pandit
apandit2@illinois.edu

I. Introduction

A. Error Correcting Codes (ECCs)

In a digital communication system, the information from a source is encoded as symbols and transmitted by the transmitter through the communication channel and is received by the receiver. In the case of an ideal channel, the same transmitted symbol is received unaltered but in practice noise is introduced by the channel which corrupts the transmitted symbols as shown in Figure 1. The communication channel can be

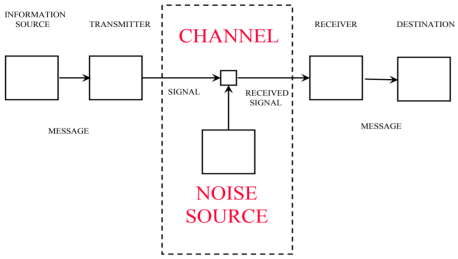

Fig. 1. A noisy communication system

modelled in several ways but one of the common models is the binary symmetric channel (BSC). As shown in Figure 2, the information is encoded as binary digits. A transmitter sends either a 1 or a 0 and the receiver receives either a 1 or 0. The probability that a bit will be flipped as it passed through the channel is modeled by the "crossover probability" p, which implies that a bit passes through unaltered with probability $(1-p)$.

Claude E. Shannon developed the notion of channel capacity

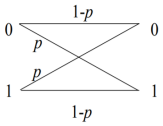

Fig. 2. Binary Symmetric Channel (BSC)

and provided a mathematical model to quantify it. The capacity of a channel is the maximum information rate that can be achieved with negligibly small error probability. Shannon developed a measure for channel capacity using entropy. The entropy – a measure of the amount of information - of a BSC is given as,

$$H_{\mathrm{b}}(p) = x \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p} \tag{1}$$

The capacity of the BSC is,

$$C_{\mathrm{BSC}} = 1 - \mathrm{H_b}(p) \tag{2}$$

Given that a channel is noisy, it motivated the development of error correcting codes (ECC). In Figure 3, a code is a mapping of information bits into codewords which contain mechanisms that enable the receiver to detect and correct errors by introducing redundancies. The code rate is defined as the ratio of the number of information bits to the total number of bits in each codeword,

$$R = k/N \tag{3}$$

According to Shannon, if $R > C_{\mathrm{BSC}}$, then the probability of

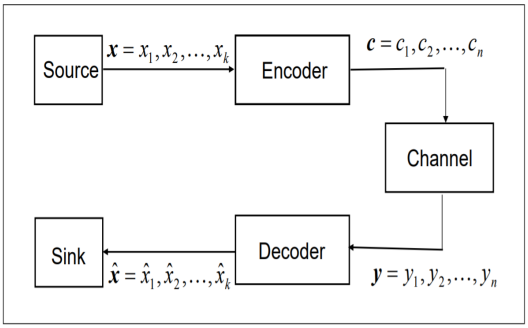

Fig. 3. Channel coding block scheme

Parameter	Description
k	Number of information bits
N	Number of code bits
d_{min}	minimum distance
$\mathcal{N}_c(j) \forall j \in \mathcal{J}$	Number of parity-check equations involving each code bit i
$\mathcal{N}_v(i) \forall i \in \mathcal{I}$	Number of code bits involved in each parity-check equation j

TABLE I
LDPC PARAMETERS

sending any information through the channel and receiving it correctly is 0 (Reliable communication is not possible). However, if $R < C_{\mathrm{BSC}}$, there exists a code of length N that can minimize the error probability while achieving maximum

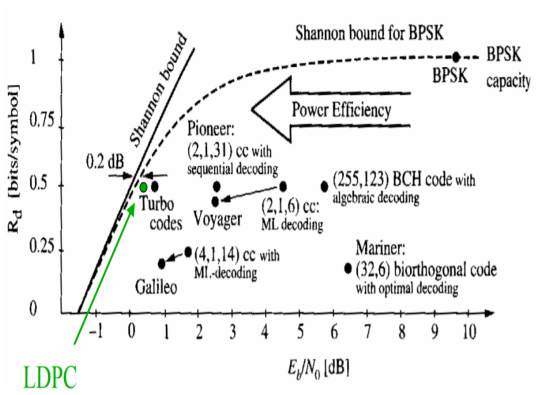

Fig. 4. Shannon limit on power efficiency

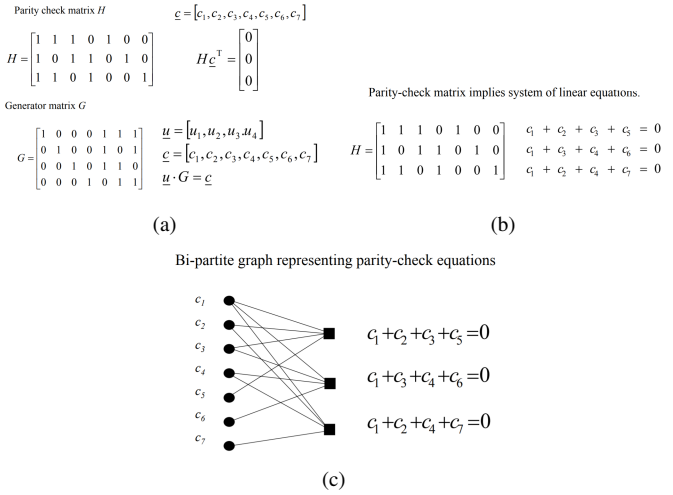

Fig. 5. (a) Parity-check & Generator matrices (b) Linear equations of parity check (c) Tanner Graph of LDPC

allowed rate R as close to C_{BSC}. Shannon also highlighted some interesting properties that these codes must possess: (1) must have randomness (2) must have large block length so that the transmitted codewords can encounter all the channel states for a time varying channel (3) codeword decoding at the receiver must be optimal. Despite highlighting these properties, Shannon never specified a method to arrive at these codes. This led to the development of several ECCs. In this paper, we focus on an optimal decoding method for one such coding technique in BSC: Low-density parity check (LDPC) codes.

B. Low-density parity check code (LDPC)

LDPC belongs to the linear block codes family and is defined by the parameters in Table I. D_{min} is the minimum number of bits by which two codewords in the scheme vary. There are several ways to describe a code (Figure 5): (1) *Codebook*: a list of all possible codewords of the scheme. (2) *Parity-check matrix/ generator matrix*: converts information bits into codewords. (3) *Graphical representation of generation*: Tanner graphs. Parity check codes detect and correct errors by checking whether the decoded codewords satisfy a system of linear equations as shown in Figure 5b. A parity-check matrix describes the linear relations that components of a codeword must satisfy. At the receiver, this matrix is used to decide whether the received/decoded data is a valid codeword. LDPC is one of the most preferred ECCs because (1) its parity-check matrix is sparse, so "low-density", meaning the number of 1's in H grows linearly with the block length (many ECCs have a faster rate of growth) (2) it achieves randomness by the random placement of 1's in H (3) its decoder is relatively simple. Thus, it satisfies the required conditions proposed by Shannon while achieving near Shannon limit code rates (Figure 4).

C. Decoding LDPC

Maximum Likelihood (ML) decoding chooses one codeword from Y (the list of all possible codewords) which maximizes the following probability,

$$\mathbb{P}(x \text{ received} \mid y \text{ sent}) \tag{4}$$

that is, the codeword y that maximizes the probability that x was received, given that y was sent. If all codewords are equally likely to be sent then this scheme is equivalent to ideal observer decoding. In fact, by Bayes Theorem,

$$\begin{aligned}\mathbb{P}(x \text{ recvd} \mid y \text{ sent}) &= \frac{\mathbb{P}(x \text{ recvd}, y \text{ sent})}{\mathbb{P}(y \text{ sent})} \\ &= \mathbb{P}(y \text{ sent} \mid x \text{ recvd}) \cdot \frac{\mathbb{P}(x \text{ recvd})}{\mathbb{P}(y \text{ sent})}\end{aligned} \tag{5}$$

By drawing upon convex optimization techniques, we can solve the optimal decoding problem of LDPC codes. It has been shown by Feldman *et al* [1] that over BSC, a relaxed version of the maximum likelihood (ML) decoding problem can be stated as a linear program (LP). The state-of-the-art decoding algorithm for LDPC is Belief Propagation (BP) [2]. However, it is only understood empirically why BP results in low error probability decoding of LDPC and furthermore when there are loops in the Tanner graphs of LDPC, BP fails [3]. The LP decoder can be used to decode LDPC at bit-error-rates comparable to state-of-the-art BP decoders, but with significantly stronger theoretical guarantees (LP decoder provides a certificate of correctness (ML certificate) [1]-verifying with probability 1 when the decoder has found the ML codeword). However, LP decoding when implemented with standard LP solvers does not easily scale to the large block lengths of modern error correcting codes. Furthermore, unlike BP, standard solvers do not have a distributed nature, limiting their scalability via parallelized (and hardware-compatible) implementation. In this paper, we use decomposition methods from optimization, specifically the Alternating Direction Method of Multipliers (ADMM), to develop efficient distributed algorithms for LP decoding.

II. Problem Statement

A. Mathematical model of LDPC

In this paper, we consider a binary linear LDPC code $\mathcal{C}$ of length N defined by a $M \times N$ parity-check matrix $\mathbf{H}$. Each of the M parity checks: $\mathcal{J} = \{1, 2, \ldots, M\}$, corresponds to a row in the parity check matrix $\mathbf{H}$ (Parity eqns in Figure 5). Codeword symbols are $\mathcal{I} = \{1, 2, \ldots, N\}$. The neighborhood of a check j, denoted by $\mathcal{N}_c(j)$, is the set of indices $i \in \mathcal{I}$ that participate in the j^{th} parity check (i.e., they are the components of the codeword that form each eqn in parity-check), $\mathcal{N}_c(j) = \{i \mid \mathbf{H}_{j,i} = 1\}$. Similarly, for a component $i \in \mathcal{I}$, $\mathcal{N}_v(i) = \{j \mid \mathbf{H}_{j,i} = 1\}$ (No. of parity eqns a particular codeword component appears in). Given a vector $\mathbf{x} \in \{0,1\}^N$, the j^{th} parity-check is said to be satisfied if $\sum_{i \in \mathcal{N}_c(j)} x_j$ is even (0). We say that a length-N binary vector $\mathbf{x}$ is a codeword, $\mathbf{x} \in \mathcal{C}$, if and only if (iff) all parity checks are satisfied ($\forall j \in \mathcal{J}$). For simplicity, we only consider LDPC codes such that for all parity-checks $j \in \mathcal{J}$, $|\mathcal{N}_c(j)| = d$, where d is a fixed constant (i.e., d codeword components are involved in each parity eqn).

To denote compactly the subset of components of $\mathbf{x}$ that participate in the j^{th} check, we use $\mathbf{P}_j$. $\mathbf{P}_j$ is the binary $d \times N$ matrix that selects out the d components of $\mathbf{x}$ that participate in the j^{th} check. For any codeword $\mathbf{x} \in \mathcal{C}$ and for any j, $\mathbf{P}_j\mathbf{x}$ is an even parity vector of dimension d. In other words, we say that $\mathbf{P}_j\mathbf{x} \in \mathbb{P}_d$ for all $j \in \mathcal{J}$ where $\mathbb{P}_d$ is defined as,

$$\mathbb{P}_d = \{\mathbf{e} \in \{0,1\}^d \mid ||\mathbf{e}||_1 \text{is even}\} \tag{6}$$

Thus, $\mathbb{P}_d$ is the codebook of the length-d single parity-check code for each j.

B. LP Formulation

The first step to solving the ML problem is formulating an equivalent LP.

Recall that a memoryless channel has the property

$$\Pr[x \mid y] = \prod_{i=1}^{n} Pr[x_i \mid y_i] \tag{7}$$

where $\Pr[x \mid y]$ is the probability that x was sent, given that y was received. If all codes are equally likely to be sent, then by Bayes' Rule

$$\Pr[x \mid y] = \Pr[y \mid x] \tag{8}$$

To maximize the likelihood of x, the negative log-likelihood is minimized. The negative log-likelihood is written as

$$\sum_{i=1}^{n} \log\left(\frac{\Pr[x_i \mid y_i = 0]}{\Pr[x_i \mid y_i = 1]}\right) y_i \tag{9}$$

Therefore the ML decoding can be formulated as minimizing a linear cost function. Let γ_i be the negative log-likelihood ratio in (9), $\gamma_i = \log\left(\frac{\Pr[x_i|y_i=0]}{\Pr[x_i|y_i=1]}\right)$. Thus, the ML decoding reduces to finding an $\mathbf{y} \in \mathcal{C}$ that minimizes,

$$\gamma^T y = \sum_{i \in \mathcal{I}} \gamma_i y_i, \text{ subject to } \mathbf{P}_j\mathbf{y} \in \mathbb{P}_d \quad \forall j \in \mathcal{J} \tag{10}$$

, which is the LP problem.

C. Relaxation of LP Decoding

It still remains to define the feasible set. Instead of solving over the set of all codewords, the LP is solved over the *codeword polytope*

$$\text{poly}(\mathcal{C}) = \left\{ \sum_{y \in \mathcal{C}} \lambda_y y \mid \lambda_y \geq 0, \sum_{y \in \mathcal{C}} \lambda_y = 1 \right\} \tag{11}$$

which is the convex hull of all possible codewords (Figure 6). The vertices of the codeword polytope correspond to codewords, and the solution to the LP will always be a vertex, so ML decoding is equivalent to solving

$$\sum_{i=1}^{n} \log\left(\frac{\Pr[x_i \mid y_i = 0]}{\Pr[x_i \mid y_i = 1]}\right) y_i \quad y \in \text{poly}(\mathcal{C})) \tag{12}$$

This set does include non-integral points. It is clear to see that if the solution $y*$ to the optimization problem over this set is an integral point, then $y*$ is the optimal codeword. If $y*$ is not an integral point, then further analysis is required to obtain the optimal codeword.

Another issue with this approach is that the dimension of the parity polytope grows with code length n, and the number of hyperplanes required to define the parity polytope increases exponentially in n. The number of constraints grows too rapidly with code length, so a relaxed polytope is constructed to make the problem tractable. The construction proceeds as follows:

(1) For the j^{th} check node, define the set $\mathcal{S}_j$ of all codewords that have even weight when applied to its neighbors. Essentially, this the set of all binary vectors which satisfy the j^{th} check. Denote the set of all $\mathcal{S}_j$ as $\mathcal{E}_j$.
(2) Construct a *local codeword polytope* from the elements of $\mathcal{E}_j$. This is identical to the construction of the codeword polytope in (11).
(3) Take the intersection of all M local codeword polytopes to obtain a global codeword polytope. Since all local codeword polytopes are convex, the global codeword polytope will also be convex.

The above is an intuitive definition of a relaxed codeword polytope; it is more convenient to define a *parity polytope* as single set that every $P_j y$ is contained in. Recall that P_j is a matrix that selects symbols from y to participate in the j^{th} check. The parity polytope is defined as

$$\mathbb{PP}_d = \text{conv}\left(\{e \in \{0,1\}^d : ||e||_1 \text{ is even}\right) \tag{13}$$

Now the constraint on y is that $P_j y \in \mathbb{PP}_d$. This is a relaxation since the previous constraint enforced that every y must belong to the convex hull of *all* codewords.

D. ADMM formulation to the LP problem

In this section we present the ADMM formulation to the LP problem in (10). ADMM is a robust approach that allows for dual ascent and also allows for dual decomposition. The ability to deliver dual decomposition allows undertaking of multiple dual ascents parallelly. This is an improvement over

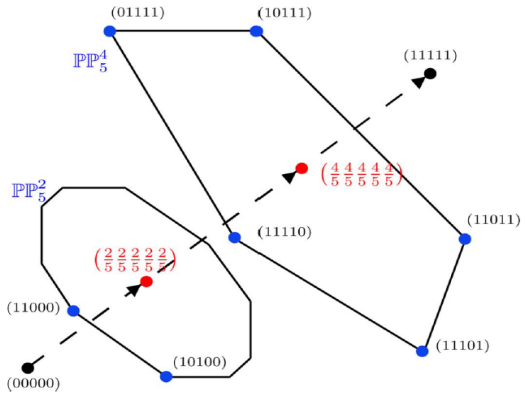

Fig. 6. Parity polytope $\mathbb{PP}_d$, $d = 5$, Each "slice" is a permutahedra containing all r-weight vertices of $\mathbb{PP}_d^r$ and orthogonal to line connecting codewords all 0's and all 1's

the method of multipliers that can not do decomposition due to the dual penalty. Therefore, before we start formulating ADMM for the given LP, it is relevant to provide insights into how ADMM improves upon method of multipliers.

Let's take a convex equality constrained optimization problem,

$$\begin{aligned} &minimize\, f(\mathbf{x}) \\ &subject\, to\, \mathbf{Ax} = \mathbf{b} \end{aligned}$$

Also, suppose f is separable,

$$f(\mathbf{x}) = f(\mathbf{x_1}) + f(\mathbf{x_2}) + ...f(\mathbf{x_N}),\ \mathbf{x} = (\mathbf{x_1}, ...\mathbf{x_N})$$

Now, for method of multipliers, we use **augmented Lagrangian**,

Augmented Lagrangian:
$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^\top(\mathbf{Ax} - \mathbf{b}) + (\rho/2)\|\mathbf{Ax} - \mathbf{b}\|^2$
Dual: $g(\lambda) = \min_{\mathbf{x}} L(\mathbf{x}, \lambda)$
Dual problem: $max\, g(\lambda)$

As we can see, due to the quadratic penalty in the augmented Lagrangian, we can not decompose the Lagrangian into separate sub-problems.
This is remedied by the ADMM approach where the form of the problem is,

$$\begin{aligned} &minimize\, f(\mathbf{x}) + g(\mathbf{z}) \\ &subject\, to\, \mathbf{Ax} + \mathbf{Bz} = \mathbf{c} \end{aligned}$$

Augmented Lagrangian:

$$\begin{aligned} L(\mathbf{x}, \mathbf{z}, \lambda) = f(\mathbf{x}) + g(\mathbf{z}) + \lambda^\top(\mathbf{Ax} + \mathbf{Bz} - \mathbf{c}) \\ + (\rho/2)\|\mathbf{Ax} + \mathbf{Bz} - \mathbf{c}\|^2 \end{aligned} \tag{14}$$

Combining the linear and quadratic term in (14), we get,

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + g(\mathbf{z}) + (\rho/2)\|\mathbf{Ax} + \mathbf{Bz} - \mathbf{c} + \mathbf{u}\|^2$$

where $\mathbf{u}_j = (1/\rho)\lambda_j$ for every j^{th} iteration.
The key to ADMM is that we minimize one variable at a time, keeping the other fixed. This allows for dual decomposition. Now, we implement the ADMM approach to the given LP problem. The LP problem given in (10),

$$\gamma^T x = \sum_{i \in \mathcal{I}} \gamma_i x_i, \text{ subject to } \mathbf{P}_j\mathbf{x} \in \mathbb{P}_d \quad \forall j \in \mathcal{J}, \mathbf{x} \in \mathcal{C}$$

To make the LP fit into the ADMM template, we relax $\mathbf{x}$ to lie in the hypercube $\mathbf{x} \in [0,1]^N$ and add auxiliary "replica" variables $\mathbf{z} \in \mathbf{R}^d$ for all $j \in \mathcal{J}$ We work with a decoupled parameterization of the decoding LP,

$$\begin{aligned} &minimize \quad \gamma^\top \mathbf{x} \\ subject\, to \quad &\mathbf{P_j x} = \mathbf{z_j} \quad \forall j \in J \\ &\mathbf{z}_j \in \mathbb{PP}_d \quad \forall j \in \mathcal{J} \\ &\mathbf{x} \in [0,1]^N \end{aligned}$$

And the augmented lagrangian for this problem is:

$$\begin{aligned} L(\mathbf{x}, \mathbf{z}, \lambda) := \gamma^\top \mathbf{x} + \sum_{j \in \mathcal{J}} \lambda^\top(\mathbf{P_j x} - \mathbf{z_j}) \\ + (\rho/2) \sum_{j \in \mathcal{J}} \|\mathbf{P_j x} - \mathbf{z_j}\|^2 \end{aligned} \tag{15}$$

Here $\lambda_\mathbf{j} \in \mathbb{R}^d$ for $j \in \mathcal{J}$ are the Lagrange multipliers and $\rho > 0$ is a fixed penalty parameter.We use λ and $\mathbf{z}$ to succinctly represent the collection of $\lambda_\mathbf{j}$'s and $\mathbf{z_j}$'s , respectively. Let $\mathcal{X}$ and $\mathcal{Z}$ denote the feasible regions for variables $\mathbf{x}$ and $\mathbf{z}$, respectively: $\mathcal{X} = [0,1]^N$ and we use $\mathbf{z} \in \mathcal{Z}$ to mean that $\mathbf{z_1} \times \mathbf{z_2} \times \ldots \mathbf{z}_{\|\mathcal{J}\|} \in \mathcal{PP}_d \times \mathcal{PP}_d \times \ldots \mathcal{PP}_d$, the $\|\mathcal{J}\|$-fold product of $\mathcal{PP}_d$. Then, we can succinctly write the iterations of ADMM as

$$\begin{aligned} \mathbf{x}^{k+1} &:= argmin_{\mathbf{x} \in \mathcal{X}} L_\rho(\mathbf{x}, \mathbf{z}^k, \lambda^k) \\ \mathbf{z}^{k+1} &:= argmin_{\mathbf{z} \in \mathcal{Z}} L_\rho(\mathbf{x}^k, \mathbf{z}, \lambda^k) \\ \lambda_j^{k+1} &:= \lambda_j^k + \rho(\mathbf{P_j x^{k+1}} - \mathbf{z_j^{k+1}}) \end{aligned}$$

The ADMM update steps involve fixing one variable and minimizing the other. In particular, $\mathbf{x^k}$ and $\mathbf{z^k}$ are the k^{th} iterate and the updates to the $\mathbf{x}$ and $\mathbf{z}$ variable are performed in an alternating fashion.

The $\mathbf{x}$-update corresponds to fixing $\mathbf{z}$ and λ (obtained from the previous iteration or initialization) and minimizing $L_\rho(\mathbf{x}, \mathbf{z}, \lambda)$ subject to $\mathbf{x} \in [0,1]^N$. Taking the gradient of (15), setting the result to zero, and limiting the result to the hypercube $\mathcal{X} = [0,1]^N$, the $\mathbf{x}$-update simplifies to

$$\mathbf{x} = \Pi_{[0,1]^N} \left(\mathbf{P^{-1}} \times \left(\sum_j \mathbf{P_j^\top} \left(\mathbf{z}_j - \frac{1}{\rho} \lambda_j \right) - \frac{1}{\rho} \gamma \right) \right)$$

where $\mathbf{P} = \sum_j \mathbf{P}_j^\top \mathbf{P}_j$ and $\Pi_{[0,1]^N}(\cdot)$ corresponds to projecting onto the hypercube $[0,1]^N$.

Component-wise, the update rule corresponds to taking the average of the corresponding replica values $\mathbf{z}_j$ adjusted by the scaled dual variable λ_j/ρ and taking a step in the negative log-likelihood direction. For any $j \in \mathcal{N}_v(i)$, let $z_j^{(i)}$ denote the component of $\mathbf{z}_j$ that corresponds to the i^{th} component of $\mathbf{x}$, in other words the i^{th} component of $\mathbf{P}_j^\top \mathbf{z_j}$. Similarly, $\lambda_j^{(i)}$ let be the i^{th} component of $\mathbf{P}_j^\top \lambda_\mathbf{j}$. With this notation, the update rule for the i^{th} component of $\mathbf{x}$ is

$$x_i = \Pi_{[0,1]^N} \left(\frac{1}{|\mathcal{N}_v(i)|} \left(\sum_{j \in \mathcal{N}_v(i)} \left(z_j^{(i)} - \frac{1}{\rho} \lambda_j^{(i)} \right) - \frac{1}{\rho} \gamma^{(i)} \right) \right)$$

Each variable update can be done in parallel.

The $\mathbf{z}$-update corresponds to fixing $\mathbf{x}$ and λ minimizing $L_\rho(\mathbf{x}, \mathbf{z}, \lambda)$ subject to $\mathbf{z}_j \in \mathcal{PP}_d$ for all $j \in \mathcal{J}$. The relevant observation here is that the augmented Lagrangian is separable with respect to the $\mathbf{z}_j$'s and hence the minimization step can be decomposed (or "factored") into $\|\mathcal{J}\|$ separate problems, each of which be solved independently. This decouples the overall problem, making the approach scalable.
We start from equation (15) and concentrate on the terms that involve $\mathbf{z}_j$. For each $j \in \mathcal{J}$ the update is to find the $\mathbf{z}_j$ that minimizes

$$\frac{\rho}{2} \|\mathbf{P}_j \mathbf{x} - \mathbf{z}\|^2 - \lambda_j^\top \mathbf{z}_j \quad , subject\, to \quad \mathbf{z}_j \in \mathcal{PP}_d$$

.

Since the values of $\mathbf{x}$ and λ are fixed, so are $\mathbf{P}_j \mathbf{x}$ and λ_j/ρ .Setting $\mathbf{v} = \mathbf{P}_j \mathbf{x} + \lambda_j/\rho$ and completing the square we get that the desired update $\mathbf{z_j^*}$ is

$$\mathbf{z_j^*} = argmin_{\tilde{\mathbf{z}} \in \mathcal{PP}_d} \|\mathbf{v} - \tilde{\mathbf{z}}\|^2$$

The $\mathbf{z}$-update thus corresponds to $\|\mathcal{J}\|$ projections onto the parity polytope.

III. Conclusion

In this paper, we highlighted an application of optimization methods for error correction in the field of communication systems. We discussed the need for error correcting codes for encoding information passing through a communication channel and focused on a particular error correcting code, the low-density parity-check codes (LDPC). We formulated the decoding of LDPC encoded codewords at the receiver as an optimization problem. We reformulated the problem by providing a relaxation to the problem and addition of auxiliary variables such that it can be solved using augmented lagrangian-based method, the alternating directions method of multipliers (ADMM).

References

[1] J. Feldman, M. J. Wainwright and D. R. Karger, "Using linear programming to Decode Binary linear codes," in IEEE Transactions on Information Theory, vol. 51, no. 3, pp. 954-972, March 2005, doi: 10.1109/TIT.2004.842696.

[2] P. O. Vontobel and R. Koetter, "On the relationship between linear programming decoding and min-sum algorithm decoding," presented at the IEEE Int. Symp. Inf. Theory and Appl., Parma, Italy, Oct. 2004.

[3] J. M. Walsh and P. A. Regalia, "Connecting Belief Propagation with Maximum Likelihood Detection," 4th International Symposium on Turbo Codes & Related Topics; 6th International ITG-Conference on Source and Channel Coding, 2006, pp. 1-6.

[4] S. Barman, X. Liu, S. C. Draper and B. Recht, "Decomposition Methods for Large Scale LP Decoding," in IEEE Transactions on Information Theory, vol. 59, no. 12, pp. 7870-7886, Dec. 2013, doi: 10.1109/TIT.2013.2281372.