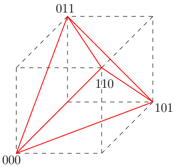
Efficient projection onto the parity polytope and its application to linear programming decoding

Stark Draper
joint work with Siddharth Barman, Xishuo Liu and Ben Recht



Communications & Signal Processing Seminar
University of Michigan
17 October 2013

Setup: consider a length-d single parity-check code

A length-d binary vector $\mathbf{x}$ is a codeword,

$$\mathbf{x} \in \mathcal{C} \quad \text{if} \quad \underbrace{[1\,1\ldots1]}_{d\ \text{ones}}\mathbf{x} = 0$$

or, equivalently, if

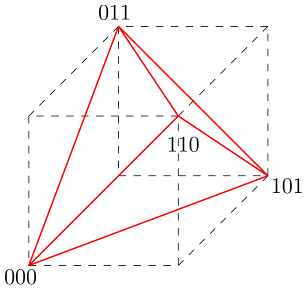
$$\mathbf{x} \in \mathbb{P}_d$$

where $\mathbb{P}_d = \{\text{all length-}d \text{ binary vectors of even weight}\}$

In other words: even-weight vertices of the d-dimension hypercube

Goal: efficient projection onto $\mathrm{conv}(\mathbb{P}_d)$, "parity polytope"

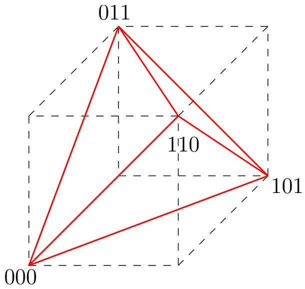
The parity polytope $\mathbb{PP}_d = \mathrm{conv}(\mathbb{P}_d)$, the convex hull of $\mathbb{P}_d$



- Number of vertices of $\mathbb{PP}_d$ is 2^{d-1}; if $d = 31$ about 1 billion

Goal: efficient projection onto $\mathrm{conv}(\mathbb{P}_d)$, "parity polytope"

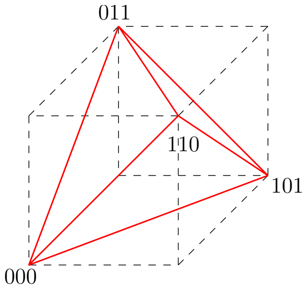
The parity polytope $\mathbb{PP}_d = \mathrm{conv}(\mathbb{P}_d)$, the convex hull of $\mathbb{P}_d$



- Number of vertices of $\mathbb{PP}_d$ is 2^{d-1}; if $d = 31$ about 1 billion
- The algorithm we develop can project any vector $\mathbf{v} \in \mathbb{R}^d$ onto $\mathbb{PP}_d$ in log-linear time, $O(d \log d)$, complexity of `sort`

Goal: efficient projection onto $\mathrm{conv}(\mathbb{P}_d)$, "parity polytope"

The parity polytope $\mathbb{PP}_d = \mathrm{conv}(\mathbb{P}_d)$, the convex hull of $\mathbb{P}_d$



- Number of vertices of $\mathbb{PP}_d$ is 2^{d-1}; if $d = 31$ about 1 billion
- The algorithm we develop can project any vector $\mathbf{v} \in \mathbb{R}^d$ onto $\mathbb{PP}_d$ in log-linear time, $O(d \log d)$, complexity of `sort`
- We use the projection to develop a new LP decoding technique via the Alternating Directions Method of Multipliers (ADMM)

Agenda

Background and Problem Setup

- LP decoding formulation: a relaxation of ML

Optimization Framework

- The alternating direction method of multipliers (ADMM)

Technical Core

- Characterizing the parity polytope
- Projecting onto the parity polytope

Experimental results

- Various codes & parameter settings
- Penalized decoder

Maximum likelihood (ML) decoding: memoryless channels

- Given codebook $\mathcal{C}$ and received sequence $\mathbf{y}$
- ML decoding picks a codeword $\mathbf{x} \in \mathcal{C}$ to:

$$\text{maximize} \quad \Pr(\text{received } \mathbf{y} \mid \text{sent } \mathbf{x})$$

$$\Updownarrow$$

$$\text{maximize} \quad \prod_i p_{Y|X}(y_i \mid x_i) \quad \text{subject to } \mathbf{x} \in \mathcal{C}$$

$$\Updownarrow$$

$$\text{maximize} \quad \sum_i \log p_{Y|X}(y_i \mid x_i) \quad \text{subject to } \mathbf{x} \in \mathcal{C}$$

Maximum likelihood (ML) decoding: binary inputs

- Objective for binary input channel:

$$\sum_i \log p_{Y|X}(y_i \mid x_i)$$
$$= \sum_i \left[\log \frac{p_{Y|X}(y_i|x_i = 1)}{p_{Y|X}(y_i|x_i = 0)} x_i + \log p_{Y|X}(y_i|x_i = 0) \right]$$

- γ_i is negative log-likelihood ratio of ith symbol, e.g., if BSC-p:

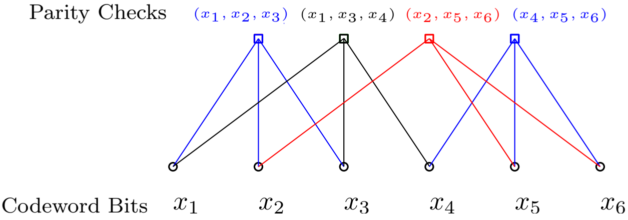
$$\gamma_i = \begin{cases} \log \frac{p}{1-p} & \text{if } y_i = 1 \\ \log \frac{1-p}{p} & \text{if } y_i = 0 \end{cases}$$

- ML decoding: linear objective, integer constraints

$$\boxed{\text{minimize} \quad \sum_i \gamma_i x_i \quad \text{s.t.} \quad \mathbf{x} \in \mathcal{C}}$$

Specialize to binary linear codes

$\mathbf{x} \in \mathcal{C}$ iff all parity checks have even parity. Factor graph:

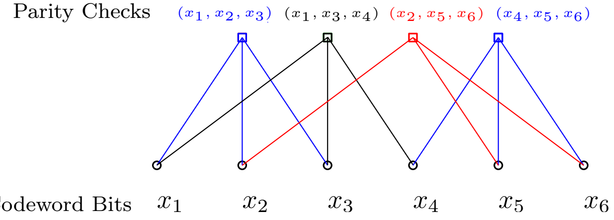


- Let $d \times n$ matrix P_j select variables neighboring jth parity check
- Examples: $P_1 \mathbf{x} = (x_1\, x_2\, x_3)$, $P_3 \mathbf{x} = (x_2\, x_5\, x_6)$
- Example:

$$P_3 \mathbf{x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} x_2 \\ x_5 \\ x_6 \end{bmatrix}$$

For simplicity: consider graphs of check degree d

Example: $d = 3$



- Let $d \times n$ matrix P_j select variables neighboring jth parity check
- Examples: $P_1\mathbf{x} = (x_1\ x_2\ x_3)$, $P_3\mathbf{x} = (x_2\ x_5\ x_6)$
- $\mathbb{P}_d = \{\text{all length-}d \text{ binary vectors of even weight}\}$

Binary linear codes

$\mathbf{x} \in \mathcal{C}$ if and only if $P_j\mathbf{x} \in \mathbb{P}_d$ for all j.

Relax $\mathbb{P}_d$ to $\mathbb{PP}_d$ to get a Linear Program (LP)

ML Decoding: an integer program with a linear objective

$$\begin{aligned} \text{minimize} \quad & \sum_i \gamma_i x_i \\ \text{subject to} \quad & P_j \mathbf{x} \in \mathbb{P}_d \quad \forall\, j \\ \text{(and} \quad & \mathbf{x} \in \{0,1\}^n) \end{aligned}$$

LP Decoding: relax $\mathbb{P}_d$ to $\mathbb{PP}_d = \mathrm{conv}(\mathbb{P}_\mathrm{d})$ for all j

$$\begin{aligned} \text{minimize} \quad & \sum_i \gamma_i x_i \\ \text{subject to} \quad & P_j \mathbf{x} \in \mathbb{PP}_d \quad \forall\, j \\ \text{and} \quad & \mathbf{x} \in [0,1]^n \end{aligned}$$

Relaxation due to Feldman, Wainwright, Karger 2005

Why care about LP decoding?

LP decoding vs. **Belief Propagation (BP)** decoding:

- **BP** empirically successful,
  inherently distributed,
  takes full advantage of spare code structure

  but, no convergence guarantees & BP suffers from error-floor

- **LP** well understood theoretically,
  has convergence guarantees,
  not observed to suffer from error-floor,
  ML certificate property,
  able to tighten relaxation to approach ML performance

  but, generic LP solvers don't efficiently exploit code sparsity

Why care about projecting onto $\mathbb{PP}_d$?

Projecting onto $\mathbb{PP}_d$: crucial step in solving the LP using the *Alternating Direction Method of Multipliers* (ADMM)

- a classic algorithm (mid-70s), efficient, scalable, distributed, convergence guarantees, numerically robust
- decomposes global problem into local subproblems, recombine iteratively (simple scheduling) to find global solution
- simple form today as objective and constraints all linear
- cf. Boyd et al. review in *FnT in Machine Learning*, 2010.

Prior work on low-complexity LP decoding:

- earliest low-complexity LP decoding results (Vontobel & Koetter '06, '08) coordinate ascent on "softened" dual
- computational complexity linear in blocklength given good choice of scheduling (Burshtein '08, '09)

Agenda

Background and Problem Setup

- LP decoding formulation: a relaxation of ML

Optimization Framework

- The alternating direction method of multipliers (ADMM)

Technical Core

- Characterizing the parity polytope
- Projecting onto the parity polytope

Experimental results

- Various codes & parameter settings
- Penalized decoder

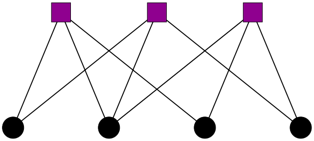
Fitting LP Decoding into ADMM template

LP Decoding:

$$\begin{aligned} \text{minimize} \quad & \sum_i \gamma_i x_i \\ \text{subject to} \quad & P_j \mathbf{x} \in \mathbb{PP}_d \quad \forall j \\ & \mathbf{x} \in [0,1]^n \end{aligned}$$

To formulate as an ADMM associate "replicas" $\mathbf{z}_j$s with each edge:

$$\begin{aligned} \text{minimize} \quad & \sum_i \gamma_i x_i \\ \text{subject to} \quad & \mathbf{z}_j = P_j \mathbf{x} \quad \forall j \\ & \mathbf{z}_j \in \mathbb{PP}_d \quad \forall j \\ & \mathbf{x} \in [0,1]^n \end{aligned}$$



- Replicas allow us to decompose into small subproblems

Lagrangian formulation

$$\text{minimize} \quad \sum_i \gamma_i x_i \qquad \text{subject to} \quad \begin{array}{l} \mathbf{z}_j = P_j \mathbf{x} \quad \forall j \\ \mathbf{z}_j \in \mathbb{PP}_d \quad \forall j \\ \mathbf{x} \in [0,1]^n \end{array}$$

Start with regular Lagrangian with multipliers $\lambda = \{\lambda_1, \lambda_2, \ldots\}$

$$\gamma^T \mathbf{x} + \sum_j \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j),$$

Lagrangian formulation

$$\text{minimize} \quad \textstyle\sum_i \gamma_i x_i \qquad \text{subject to} \quad \begin{array}{l} \mathbf{z}_j = P_j \mathbf{x} \;\; \forall j \\ \mathbf{z}_j \in \mathbb{PP}_d \;\; \forall j \\ \mathbf{x} \in [0,1]^n \end{array}$$

Start with regular Lagrangian with multipliers $\lambda = \{\lambda_1, \lambda_2, \ldots\}$

$$\gamma^T \mathbf{x} + \sum_j \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j),$$

ADMM works with an augmented Lagrangian:

$$L_\mu(\mathbf{x}, \mathbf{z}, \lambda) := \gamma^T \mathbf{x} + \sum_j \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2} \sum_j \|P_j \mathbf{x} - \mathbf{z}_j\|_2^2$$

Effect is to smooth the dual problem, accelerating convergence

Alternating Direction Method of Multipliers

Round-robin update of $\mathbf{x}$ then $\mathbf{z}$ then λ until converge:

$$L_\mu(\mathbf{x}, \mathbf{z}, \lambda) := \gamma^T \mathbf{x} + \sum_j \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2} \sum_j \| P_j \mathbf{x} - \mathbf{z}_j \|_2^2$$

ADMM Update Steps:

$$\begin{aligned} \mathbf{x}^{k+1} &:= \mathrm{argmin}_{\mathbf{x} \in \mathcal{X}} L_\mu(\mathbf{x}, \mathbf{z}^k, \lambda^k) \\ \mathbf{z}^{k+1} &:= \mathrm{argmin}_{\mathbf{z} \in \mathcal{Z}} L_\mu(\mathbf{x}^{k+1}, \mathbf{z}, \lambda^k) \\ \lambda_j^{k+1} &:= \lambda_j^k + \mu \left(P_j \mathbf{x}^{k+1} - \mathbf{z}_j^{k+1} \right) \end{aligned}$$

where

$$\begin{aligned} \mathcal{X} &= [0,1]^n \\ \mathcal{Z} &= \underbrace{\mathbb{PP}_d \times \ldots \times \mathbb{PP}_d}_{\text{number of checks}} \end{aligned}$$

- Updates: msg-passing on a "Forney-style" factor graph

ADMM x-Update: turns out to be (almost) averaging

With $\mathbf{z}$ and λ fixed the $\mathbf{x}$-updates are:

$$\text{minimize} \quad L_\mu(\mathbf{x}, \mathbf{z}^k, \lambda^k) \quad \text{subject to} \quad \mathbf{x} \in [0,1]^n \quad \text{where}$$

$$L_\mu(\mathbf{x}, \mathbf{z}, \lambda) := \gamma^T \mathbf{x} + \sum_j \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2} \sum_j \|P_j \mathbf{x} - \mathbf{z}_j\|_2^2$$

ADMM x-Update: turns out to be (almost) averaging

With $\mathbf{z}$ and λ fixed the $\mathbf{x}$-updates are:

$$\text{minimize} \quad L_\mu(\mathbf{x}, \mathbf{z}^k, \lambda^k) \quad \text{subject to} \quad \mathbf{x} \in [0,1]^n \quad \text{where}$$

$$L_\mu(\mathbf{x}, \mathbf{z}, \lambda) := \gamma^T \mathbf{x} + \sum_j \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2} \sum_j \|P_j \mathbf{x} - \mathbf{z}_j\|_2^2$$

Partial derivatives of a quadratic form (and apply box constraints)

$$\frac{\partial}{\partial x_i} L_\mu(\mathbf{x}, \mathbf{z}^k, \lambda^k) = 0$$

ADMM $\mathbf{x}$-Update: turns out to be (almost) averaging

With $\mathbf{z}$ and λ fixed the $\mathbf{x}$-updates are:

$$\text{minimize} \quad L_\mu(\mathbf{x}, \mathbf{z}^k, \lambda^k) \quad \text{subject to} \quad \mathbf{x} \in [0,1]^n \quad \text{where}$$

$$L_\mu(\mathbf{x}, \mathbf{z}, \lambda) := \gamma^T \mathbf{x} + \sum_j \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2} \sum_j \|P_j \mathbf{x} - \mathbf{z}_j\|_2^2$$

Partial derivatives of a quadratic form (and apply box constraints)

$$\frac{\partial}{\partial x_i} L_\mu(\mathbf{x}, \mathbf{z}^k, \lambda^k) = 0$$

Get component-wise (averaging) updates:

$$x_i = \Pi_{[0,1]} \left(\frac{1}{|\mathcal{N}_v(i)|} \left(\sum_{j \in \mathcal{N}_v(i)} \left(\mathbf{z}_j^{(i)} - \frac{1}{\mu} \lambda_j^{(i)} \right) - \frac{1}{\mu} \gamma_i \right) \right)$$

$\mathcal{N}_v(i)$: set of parity checks neighboring variable i.
$\mathbf{z}_j^{(i)}$: component of the jth replica associated with x_i.

ADMM z-Update

Recall:

$$L_\mu(\mathbf{x}, \mathbf{z}, \lambda) := \gamma^T \mathbf{x} + \sum_j \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2} \sum_j \|P_j \mathbf{x} - \mathbf{z}_j\|_2^2$$

z-update: with **x** and λ fixed we want to solve

$$\begin{aligned} \text{minimize} \quad & \sum_j \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2} \sum_j \|P_j \mathbf{x} - \mathbf{z}_j\|_2^2 \\ \text{subject to} \quad & \mathbf{z}_j \in \mathbb{PP}_d \quad \forall j \end{aligned}$$

The minimization is separable in j: for each j we need to solve

$$\begin{aligned} \text{minimize} \quad & \lambda_j^T (P_j \mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2} \|P_j \mathbf{x} - \mathbf{z}_j\|_2^2 \\ \text{subject to} \quad & \mathbf{z}_j \in \mathbb{PP}_d \end{aligned}$$

ADMM $\mathbf{z}_j$-Update: project onto parity polytope

$\mathbf{z}_j$-update:

$$\begin{aligned} \text{minimize} \quad & \lambda_j^T(P_j\mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2}\|P_j\mathbf{x} - \mathbf{z}_j\|_2^2 \\ \text{subject to} \quad & \mathbf{z}_j \in \mathbb{PP}_d \end{aligned}$$

Setting $\mathbf{v} = P_j\mathbf{x} + \lambda_j/\mu$ (completing the square) the problem is equivalent to:

$$\boxed{\begin{aligned} \text{minimize} \quad & \|\mathbf{v} - \tilde{\mathbf{z}}\|_2^2 \\ \text{subject to} \quad & \tilde{\mathbf{z}} \in \mathbb{PP}_d \end{aligned}}$$

The primary challenge in ADMM

The $\mathbf{z}$-update requires *projecting onto the parity polytope*.

Agenda

Background and Problem Setup

- LP decoding formulation: a relaxation of ML

Optimization Framework

- The alternating direction method of multipliers (ADMM)

Technical Core

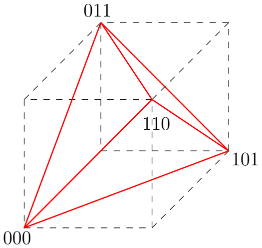
- Characterizing the parity polytope
- Projecting onto the parity polytope

Experimental results

- Various codes & parameter settings
- Penalized decoder

Prior characterizations of parity polytope

- Jeroslow (1975)
- Yannakakis (1991) has a quadratic d^2 characterization
- Feldman et al. (2005) use Yannakakis
- "Standard Polytope" in Feldman uses 2^{d-1} linear constraints per parity-check, many not active as exploited in "Adaptive LP Decoding" Taghavi and Siegel (2008)



Most points in $\mathbb{PP}_d$ have multiple representations

By definition:

- $\mathbf{y} \in \mathbb{PP}_d$ iff $\mathbf{y} = \sum_i \alpha_i \mathbf{e}_i$
- $\sum_i \alpha_i = 1$, $\alpha_i \geq 0$
- $\mathbf{e}_i$ are even-hamming-weight binary vectors of dimension d
- Most $\mathbf{y} \in \mathbb{PP}_d$ have multiple representations

Example A ($d = 6$):

$$\begin{pmatrix} 1 \\ 1 \\ 1/2 \\ 1/2 \\ 1/4 \\ 1/4 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{4} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{4} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Most points in $\mathbb{PP}_d$ have multiple representations

By definition:

- $\mathbf{y} \in \mathbb{PP}_d$ iff $\mathbf{y} = \sum_i \alpha_i \mathbf{e}_i$
- $\sum_i \alpha_i = 1,\ \alpha_i \geq 0$
- $\mathbf{e}_i$ are even-hamming-weight binary vectors of dimension d
- Most $\mathbf{y} \in \mathbb{PP}_d$ have multiple representations

Example B ($d = 6$):

$$\begin{pmatrix} 1 \\ 1 \\ 1/2 \\ 1/2 \\ 1/4 \\ 1/4 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

There always exists a "two-slice" representation

Two-Slice Lemma:

For any $\mathbf{y} \in \mathbb{PP}_d$ there exists a representation $\mathbf{y} = \sum_i \alpha_i \mathbf{e}_i$ where

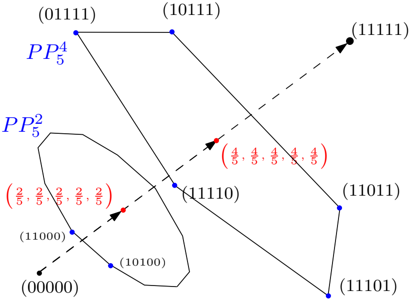
- $\sum_i \alpha_i = 1$, $\alpha_i \geq 0$
- $\mathbf{e}_i$ are of only two weights: r or $r+2$ for all i
- r is the even integer $r = \lfloor \|\mathbf{y}\|_1 \rfloor_{\text{even}}$

Example B is one such representation with $d = 6$ and $r = 2$:

$$\begin{pmatrix} 1 \\ 1 \\ 1/2 \\ 1/2 \\ 1/4 \\ 1/4 \end{pmatrix} = \frac{1}{4} \underbrace{\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\text{wt}=2} + \frac{1}{2} \underbrace{\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}}_{\text{wt}=4} + \frac{1}{4} \underbrace{\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}}_{\text{wt}=4}$$

Visualizing properties of $\mathbb{PP}_d$: always between two slices

Example: d = 5



- Let $\mathbb{PP}_d^r = \mathrm{conv}\{\mathbf{e}_i \mid \|\mathbf{e}_i\|_1 = r\}$, a "permutohedron"
  $\Rightarrow$ easy to characterize using majorization
- **Two-slice restated:** Any $\mathbf{y} \in \mathbb{PP}_d$ is sandwiched between two permutohedrons $\mathbb{PP}_d^r$ and $\mathbb{PP}_d^{r+2}$ where $r = \lfloor \|\mathbf{y}\|_1 \rfloor_{\text{even}}$

Majorization: definition & application to $\mathbb{PP}_d^r$

Definition: Let $\mathbf{u}$ and $\mathbf{w}$ be d-vectors *sorted* in decreasing order. The vector $\mathbf{w}$ is said to majorize $\mathbf{u}$ if

$$\sum_{k=1}^{d} u_k = \sum_{k=1}^{d} w_k$$

$$\sum_{k=1}^{q} u_k \leq \sum_{k=1}^{q} w_k \quad \forall \quad q,\ 1 \leq q < d$$

Specialize to $\mathbb{PP}_d^r$ where $\mathbf{w} = [\underbrace{1\ 1 \ldots 1}_{r}\ \underbrace{0\ 0 \ldots 0}_{d-r}]$

$$\sum_{k=1}^{d} u_k = r$$

$$\sum_{k=1}^{q} u_k \leq \min(q, r) \quad \forall \quad q,\ 1 \leq q < d$$

Majorization & permutohedrons

Theorem: $\mathbf{u}$ is in the convex hull of all permutations of $\mathbf{w}$ (the permutohedron defined by $\mathbf{w}$) if and only if $\mathbf{w}$ majorizes $\mathbf{u}$.

Majorization & permutohedrons

Theorem: $\mathbf{u}$ is in the convex hull of all permutations of $\mathbf{w}$ (the permutohedron defined by $\mathbf{w}$) if and only if $\mathbf{w}$ majorizes $\mathbf{u}$.

$$\mathbf{u} = \sum_i \beta_i \mathbf{\Sigma_i} \mathbf{w}$$

where $\mathbf{\Sigma_i}$ are permutation matrices and β_i are weightings

Majorization & permutohedrons

Theorem: $\mathbf{u}$ is in the convex hull of all permutations of $\mathbf{w}$ (the permutohedron defined by $\mathbf{w}$) if and only if $\mathbf{w}$ majorizes $\mathbf{u}$.

$$\mathbf{u} = \sum_i \beta_i \mathbf{\Sigma_i} \mathbf{w}$$

where $\mathbf{\Sigma_i}$ are permutation matrices and β_i are weightings

Proving two-slice lemma:

- Use above to characterize each $\mathbb{PP}_d^r$, r even, $r \leq d$.
- Express $\mathbf{y}$ as a weighted combination of points in $\mathbb{PP}_d^r$, $1 \leq r \leq d$.
- Show you can set all weightings to zeros except those on $r = \lfloor \|\mathbf{y}\|_1 \rfloor_{\mathrm{even}}$ and $r = \lfloor \|\mathbf{y}\|_1 \rfloor_{\mathrm{even}} + 2$.
- Note that finding r is trivial.

Next: use two-slice lemma to develop projection operation

Projecting onto the parity polytope

Desired projection:

$$\begin{aligned} &\min \; \|\mathbf{v} - \mathbf{y}\|_2^2 \\ &\text{s.t. } \mathbf{y} \in \mathbb{PP}_d \end{aligned}$$

Projecting onto the parity polytope

Desired projection:

$$\min \|\mathbf{v} - \mathbf{y}\|_2^2$$
$$\text{s.t. } \mathbf{y} \in \mathbb{PP}_d$$

Use two-slice lemma to reformulate as:

$$\begin{aligned} \min\ & \|\mathbf{v} - \alpha \mathbf{s} - (1-\alpha)\mathbf{t}\|_2^2 \\ \text{s.t.}\ & 0 \le \alpha \le 1,\ \mathbf{s} \in \mathbb{PP}_d^r,\ \mathbf{t} \in \mathbb{PP}_d^{r+2} \end{aligned}$$

Projecting onto the parity polytope

Desired projection:

$$\min \ \|\mathbf{v} - \mathbf{y}\|_2^2 \quad \text{s.t. } \mathbf{y} \in \mathbb{PP}_d$$

Use two-slice lemma to reformulate as:

$$\min \ \|\mathbf{v} - \alpha\mathbf{s} - (1-\alpha)\mathbf{t}\|_2^2 \quad \text{s.t. } 0 \le \alpha \le 1,\ \mathbf{s} \in \mathbb{PP}_d^r,\ \mathbf{t} \in \mathbb{PP}_d^{r+2}$$

We also show (where $\Pi(\cdot)$ is shorthand for projection):

$$\underbrace{\left\lfloor \|\Pi_{[0,1]^d}(\mathbf{v})\|_1 \right\rfloor_{\text{even}}}_{\mathbf{r}} \le \|\Pi_{\mathbb{PP}_d}(\mathbf{v})\|_1 \le \underbrace{\left\lfloor \|\Pi_{[0,1]^d}(\mathbf{v})\|_1 \right\rfloor_{\text{even}} + 2}_{\mathbf{r+2}}$$

in other words, it is *trivial* to identify the two slices

Use majorization to simplify problem further

Assume w.l.o.g that $\mathbf{v}$ is sorted and let

$$\mathbf{z} = \Pi_{\mathbb{PP}_d}(\mathbf{v}) = \arg\min \ \|\mathbf{v} - \alpha\mathbf{s} - (1-\alpha)\mathbf{t}\|_2^2$$
$$\text{s.t.} \quad 0 \le \alpha \le 1, \ \mathbf{s} \in \mathbb{PP}_d^r, \ \mathbf{t} \in \mathbb{PP}_d^{r+2}$$

Constraint set can be restated as

(i) $\quad 0 \le \alpha \le 1$

(ii) $\quad \displaystyle\sum_{k=1}^{d} z_k = \alpha r + (1-\alpha)(r+2)$

(iii) $\quad \displaystyle\sum_{k=1}^{q} z_k \le \alpha \min(q, r) + (1-\alpha)\min(q, r+2) \ \ \forall \ \ q, \ 1 \le q < d$

(iv) $\quad z_1 \ge z_2 \ge \ldots \ge z_d$

Combine knowledge of r with first two constraints

From (ii) we have

$$\sum_{k=1}^{d} z_k = \alpha r + (1-\alpha)(r+2) \qquad (*)$$

Now we apply the bound from (i) on α, $0 \leq \alpha \leq 1$ to get

$$r \leq \sum_{k=1}^{d} z_k \leq r+2$$

Deal with third constraint

Consider the partial sums of the sorted vectors

$$\sum_{k=1}^{q} z_k \leq \alpha \min(q, r) + (1-\alpha)\min(q, r+2) \;\; \forall \;\; q, 1 \leq q < d$$

- For $q \leq r$ ineq. satisfied by box constraints: $0 \leq z_k \leq 1 \; \forall k$
- For $q \geq r+2$ inequalities also satisfied since

$$\sum_{k=1}^{q} z_k \leq \sum_{k=1}^{d} z_k = \alpha r + (1-\alpha)(r+2) \qquad (*)$$

Hence only need to deal with $q = r+1$, which specializes as

$$\sum_{k=1}^{r+1} z_k \leq \alpha r + (1-\alpha)(r+1) = r + (1-\alpha) \qquad (**)$$

Third constraint (continued...)

Solve $(*)$ for α to find

$$\alpha = 1 + \frac{r - \sum_{k=1}^{d} z_k}{2}.$$

Finally, substitute into $(**)$ to get

$$\begin{aligned}\sum_{k=1}^{r+1} z_k &\le r + (1-\alpha) \\ &= r - \frac{r - \sum_{k=1}^{d} z_k}{2}\end{aligned}$$

Which becomes

$$\sum_{k=1}^{r+1} z_k - \sum_{k=r+2}^{d} z_k \le r$$

Reformulated projection as a quadratic program (QP)

$$
\begin{aligned}
\min \quad & \|\mathbf{v} - \alpha \mathbf{s} - (1-\alpha)\mathbf{t}\|_2^2 \\
\text{s.t.} \quad & 0 \leq \alpha \leq 1 \\
& \mathbf{s} \in \mathbb{PP}_d^r, \\
& \mathbf{t} \in \mathbb{PP}_d^{r+2}
\end{aligned}
$$

$$
\begin{aligned}
\min \quad & \|\mathbf{v} - \mathbf{z}\|_2^2 \\
\text{s.t.} \quad & 1 \geq z_k \geq 0 \ \forall \ k \\
& z_1 \geq z_2 \geq \ldots \geq z_d \\
& r + 2 \geq \sum_k z_k \geq r \\
& r \geq \sum_{k=1}^{r+1} z_k - \sum_{k=r+2}^{d} z_k
\end{aligned}
$$

Reformulated projection as a quadratic program (QP)

$$\begin{aligned} \min \quad & \|\mathbf{v} - \alpha \mathbf{s} - (1-\alpha)\mathbf{t}\|_2^2 \\ \text{s.t.} \quad & 0 \le \alpha \le 1 \\ & \mathbf{s} \in \mathbb{PP}_d^r, \\ & \mathbf{t} \in \mathbb{PP}_d^{r+2} \end{aligned}$$

$$\begin{aligned} \min \quad & \|\mathbf{v} - \mathbf{z}\|_2^2 \\ \text{s.t.} \quad & 1 \ge z_k \ge 0 \ \forall \ k \\ & z_1 \ge z_2 \ge \ldots \ge z_d \\ & r+2 \ge \sum_k z_k \ge r \\ & r \ge \sum_{k=1}^{r+1} z_k - \sum_{k=r+2}^{d} z_k \end{aligned}$$

- for the QP the KKT conditions are necessary and sufficient
- we develop a linear-time water-filling type algorithm that determines a solution satisfying the KKT conditions

$$\mathbf{z}^* = \Pi_{[0,1]^d}\left(\mathbf{v} - \beta[\underbrace{1 \ldots 1}_{r+1} \ \underbrace{-1 \ldots -1}_{d-r-1}]\right) \quad \text{some} \quad \beta_{\text{opt}} \in [0, \beta_{\max}]$$

Agenda

Background and Problem Setup

- LP decoding formulation: a relaxation of ML

Optimization Framework

- The alternating direction method of multipliers (ADMM)

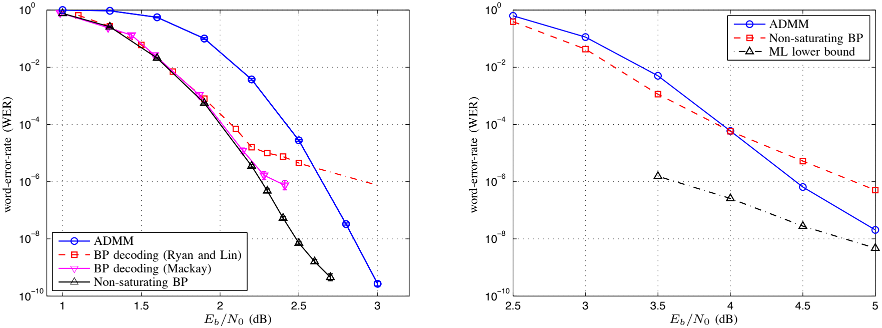
Technical Core

- Characterizing the parity polytope
- Projecting onto the parity polytope

Experimental results

- Various codes & parameter settings
- Penalized decoder

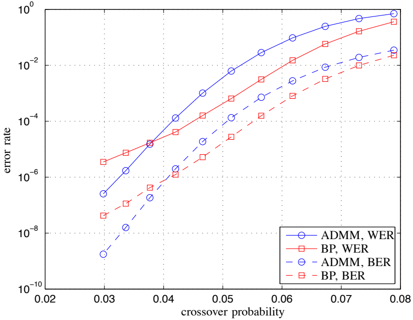
Performance results: two LDPC codes over AWGN



- length-2640, rate-0.5
- $(3, 6)$-regular LDPC
- *non-saturating* BP per Butler & Siegel (Allerton '11)

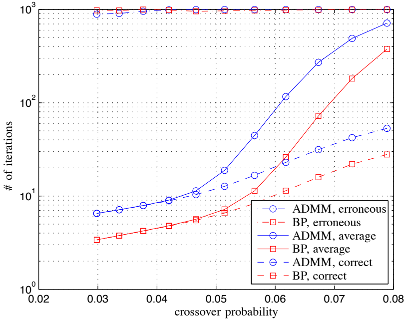
- length-1057, rate-0.77
- $(3, 13)$-regular LDPC
- observable error floor

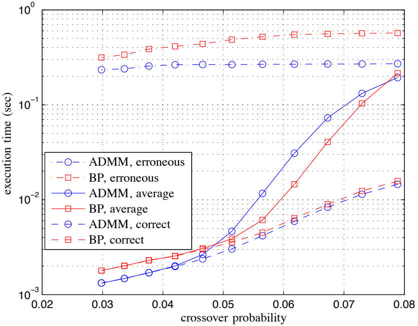
Performance results: random LDPC ensemble over BSC



- results averaged over ensemble of 100 codes
- each a randomly generated length-1002 $(3, 6)$-regular LDPC
- all codes had girth at least 4

Random ensemble: iteration count & execution time





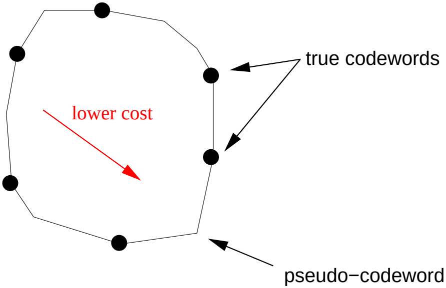
- iteration count
- ADMM & BP for:
  (i) errors, (ii) avg, (iii) correct

- execution time
- ADMM & BP for
  (i) errors, (ii) avg, (iii) correct

Understanding LP decoding failures

LP decoding fails to a "pseudocodeword", a non-integer vertex of the fundamental polytope introduced when we relaxed each of the various integer constraints $\mathbb{P}_d$ to $\mathbb{PP}_d$ in

$$\min \gamma^T \mathbf{x} \quad \text{s.t.} \quad P_j \mathbf{x} \in \mathbb{PP}_d \ \forall\, j, \quad \mathbf{x} \in [0,1]^n$$



ℓ_2-penalized ADMM

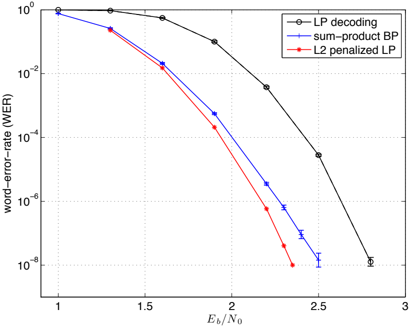
In order to eliminate pseudocodewords, introduce an ℓ_2-penalty to push the solution towards an integral solution, now solve:

$$\min \ \gamma^T \mathbf{x} - c\|\mathbf{x} - 0.5\|_2 \quad \text{s.t.} \quad \mathrm{P_j}\mathbf{x} \in \mathbb{PP}_\mathrm{d} \ \ \forall\, \mathrm{j}, \ \ \mathbf{x} \in [0,1]^\mathrm{n}$$

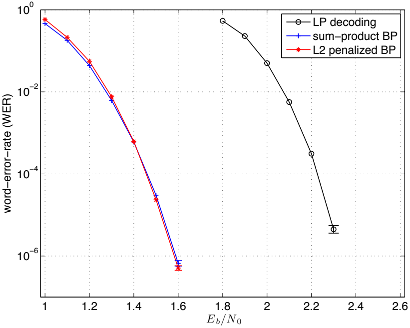
ℓ_2-penalized ADMM

In order to eliminate pseudocodewords, introduce an ℓ_2-penalty to push the solution towards an integral solution, now solve:

$$\min \; \gamma^T \mathbf{x} - c\|\mathbf{x} - 0.5\|_2 \quad \text{s.t.} \quad P_j\mathbf{x} \in \mathbb{PP}_d \;\; \forall\, j, \;\; \mathbf{x} \in [0,1]^n$$



[2640,1320] “Margulis” LDPC



[13298, 3296] rate-0.25 LDPC

Recap & wrap-up

Recap:

- LP decoding via ADMM
- main hurdle: efficient projection onto the parity polytope, complexity of `sort`
- simple scheduling and complexity linear in the block-length
- roughly same execution time as BP
- further improvements via ℓ_2-penalty (alternately ℓ_1-penalty)
- **Try it yourself!** Documented code available at

  `https://sites.google.com/site/xishuoliu/codes`

Things to do:

- error floor analysis (LP & penalized)
- effects of finite precision
- how to implement in hardware
- understand BP/LP low-SNR gap (without penalty)
- other codes: non-binary codes, permutation-based codes

2014 IEEE North American School on Information Theory

To be held at the Fields Institute at the University of Toronto
18-21 June 2014