Decomposition Methods for Large Scale LP Decoding

Siddharth Barman

Joint work with Xishuo Liu, Stark Draper, and Ben Recht

Outline

- Background and Problem Setup
  LP Decoding Formulation
- Optimization Framework
  ADMM
- Technical Core
  Projecting onto the Parity Polytope
- Numerical results
  Graphs

Efficient and Reliable Digital Transmission

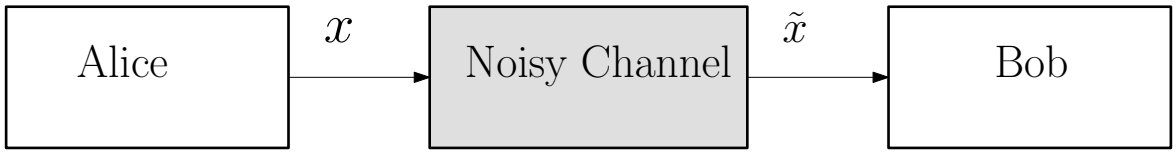
Error Correcting Codes

Constructs that enable reliable delivery of digital data over unreliable (noisy) communication channels.

Claude Shannon

Richard Hamming

Model for Communication Channel



Binary Symmetric Channel (BSC)

We will focus on transmission of bit strings ($x, \tilde{x} \in \{0,1\}^n$) and the Binary Symmetric Channel.

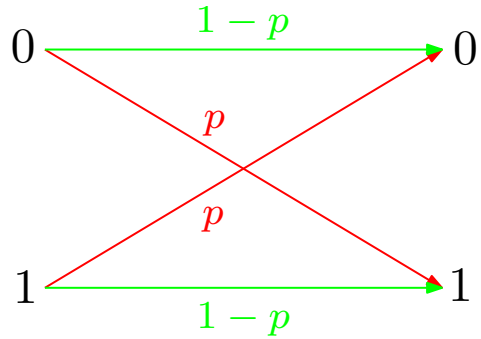
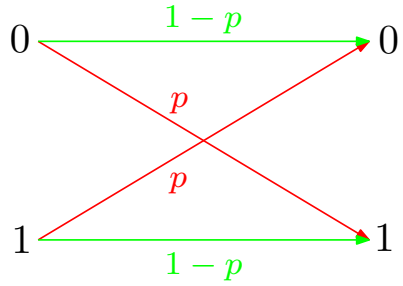


Figure : BSC with crossover probability p

Probabilistic implications of BSC

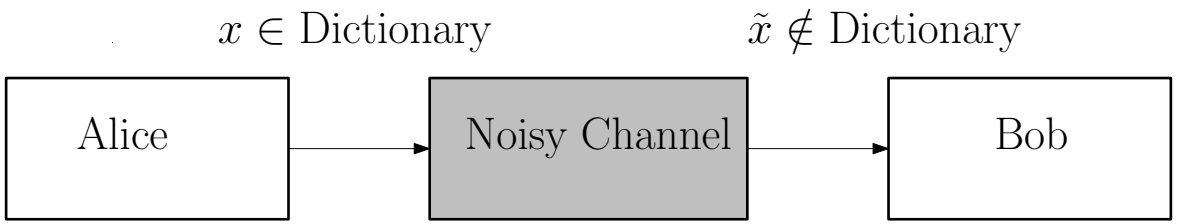


- Say x is the transmitted bit string and $\tilde{x}$ is the received bit string.
- BSC is memoryless hence

$$\Pr(\tilde{x} \mid x) = \prod_i \Pr(\tilde{x}_i \mid x_i)$$

Example: $\Pr(\tilde{x} = 001 \mid x = 000) = (1-p) \times (1-p) \times p$.

Executive Summary of Error Correcting Codes



- Codewords $\mathcal{C}$: A "Dictionary" (structured set).
- Error Detection: Spell Checking.
- Error Correction: Spell Correction.

Phonetic Alphabet

A - alpha	N - november
B - bravo	O - oscar
C - charlie	P - papa
D - delta	Q - quebec
E - echo	R - romeo
F - foxtrot	S - sierra
G - golf	T - tango
H - hotel	U - uniform
I - india	V - victor
J - juliet	W - wiskey
K - kilo	X - x-ray
L - lima	Y - yankee
M - mike	Z - zulu

Maximum Likelihood (ML) Decoding

- With codeword $\mathcal{C}$ and received bit string $\tilde{x}$, ML decoding picks a codeword $x \in \mathcal{C}$ that maximizes the probability that $\tilde{x}$ was received given that x was sent: $\Pr(\text{received } \tilde{x} \mid \text{sent } x)$

$$\text{maximize} \quad \Pr(\tilde{x} \mid x) \quad \text{subject to } x \in \mathcal{C}$$

$$\Updownarrow$$

$$\text{maximize} \quad \prod_i \Pr(\tilde{x}_i \mid x_i) \quad \text{subject to } x \in \mathcal{C}$$

Maximum Likelihood (ML) Decoding

- With codeword $\mathcal{C}$ and received bit string $\tilde{x}$, ML decoding picks a codeword $x \in \mathcal{C}$ that maximizes the probability that $\tilde{x}$ was received given that x was sent: $\Pr(\text{received } \tilde{x} \mid \text{sent } x)$

$$\text{maximize} \quad \Pr(\tilde{x} \mid x) \quad \text{subject to } x \in \mathcal{C}$$

$$\Updownarrow$$

$$\text{maximize} \quad \prod_i \Pr(\tilde{x}_i \mid x_i) \quad \text{subject to } x \in \mathcal{C}$$

$$\Updownarrow$$

$$\text{maximize} \quad \sum_i \log \Pr(\tilde{x}_i \mid x_i) \quad \text{subject to } x \in \mathcal{C}$$

Maximum Likelihood (ML) Decoding

- ML decoding:

$$\text{maximize} \quad \sum_i \log \Pr(\tilde{x}_i \mid x_i) \quad \text{s.t.} \quad x \in \mathcal{C}$$

- Negative log-likelihood ratios for all i:

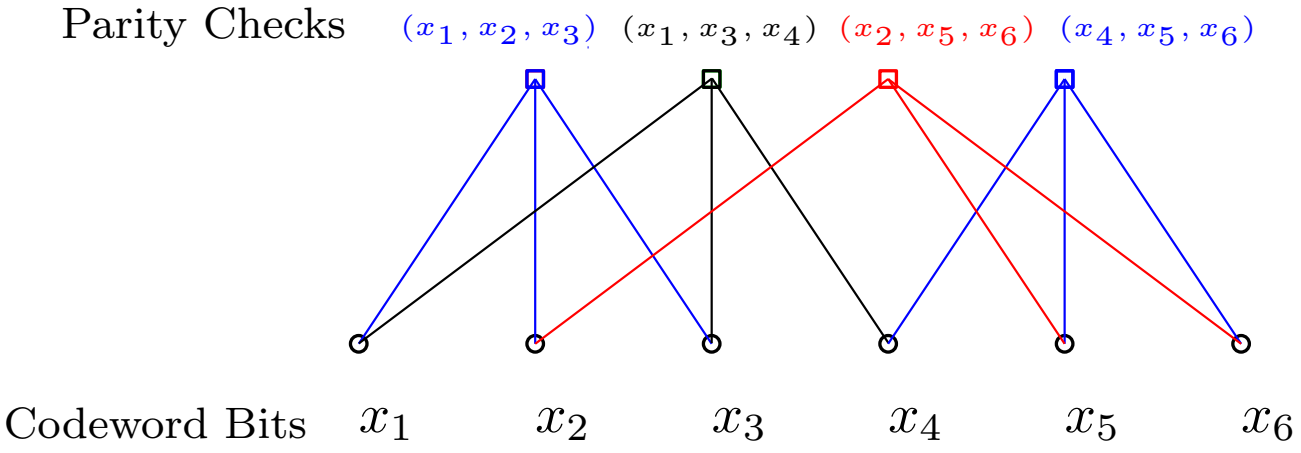
$$\gamma_i = \begin{cases} \log \frac{p}{1-p} & \text{if } \tilde{x}_i = 1 \\ \log \frac{1-p}{p} & \text{if } \tilde{x}_i = 0 \end{cases}$$

- ML decoding:

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

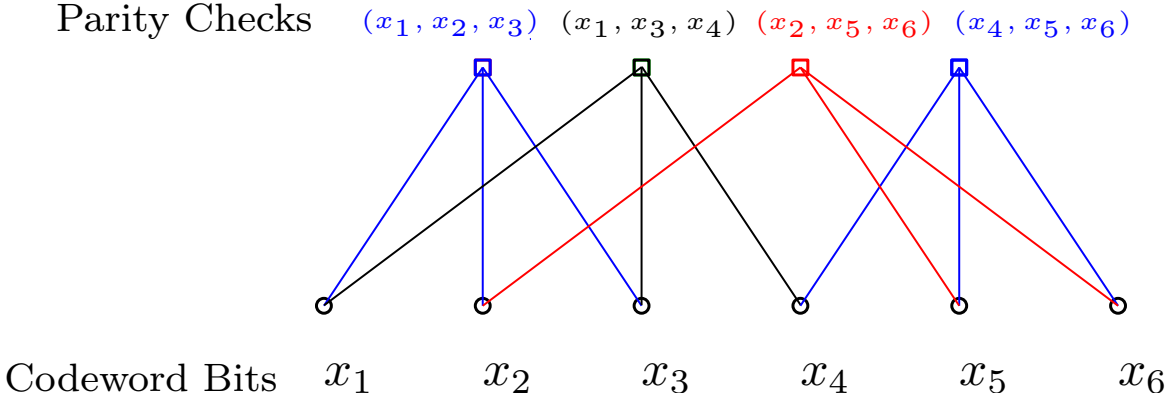
Low Density Parity Check (LDPC) Codes

- LDPC: $x \in \mathcal{C}$ iff all the parity checks are satisfied.



- Example: $x = (1\ 0\ 1\ 0\ 1\ 1\ )$
  - Parity Check 1: (1 0 1)
  - Parity Check 2: (1 1 0)
  - Parity Check 3: (0 1 1)
  - Parity Check 4: (0 1 1)

Decoding Low Density Parity Check (LDPC) Codes



- Let $P_j x$ be the sub-vector participating in the jth parity check.
- $P_1 x = (x_1\ x_2\ x_3)$, $P_4 x = (x_4\ x_5\ x_6)$ and $d = 3$.
- $\mathbb{P}_d = \{\text{all even parity bit-vectors of length } d\}$

LDPC

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

ML Decoding for LDPC Codes

- γ: Negative log-likelihood ratios.
- $P_j x$: The sub-vector participating in the jth parity check.
- $\mathbb{P}_d = \{\text{all even parity bit-vectors of length } d\}$

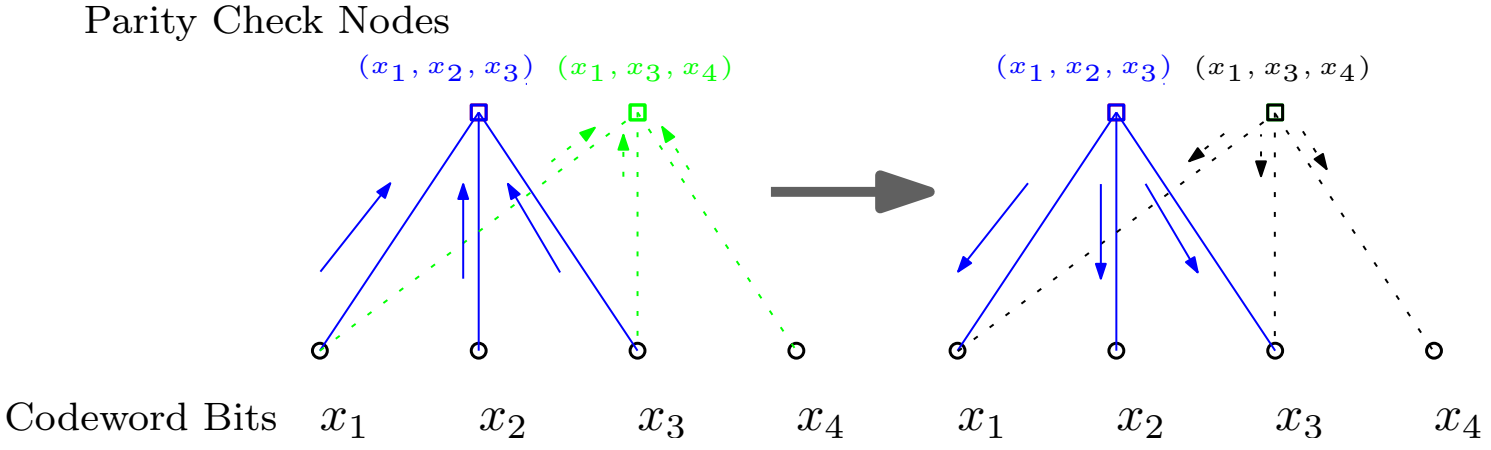
ML decoding:

$$\begin{aligned} \text{minimize} \quad & \sum_i \gamma_i x_i \\ \text{subject to} \quad & x \in \mathcal{C} \end{aligned}$$

ML Decoding for LDPC Codes:

$$\begin{aligned} \text{minimize} \quad & \sum_i \gamma_i x_i \\ \text{subject to} \quad & P_j x \in \mathbb{P}_d \quad \forall j \end{aligned}$$

Decoding by Belief Propagation (BP)



- BP has been empirically successful in decoding LDPC codes but possess are no convergence guarantees.
- Inherently distributed and takes full advantage of locality (low density of parity checks).
- A distributed decoding algorithm is desirable as it directly implies scalability.

Decoding Linear Program

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

Parity Polytope, $\mathbb{PP}_d$, is the convex hull of all even parity bit-vectors of length d, $\mathbb{PP}_d = \text{conv}(\mathbb{P}_d)$.

Feldman et al. (2005) proposed the following relaxation:

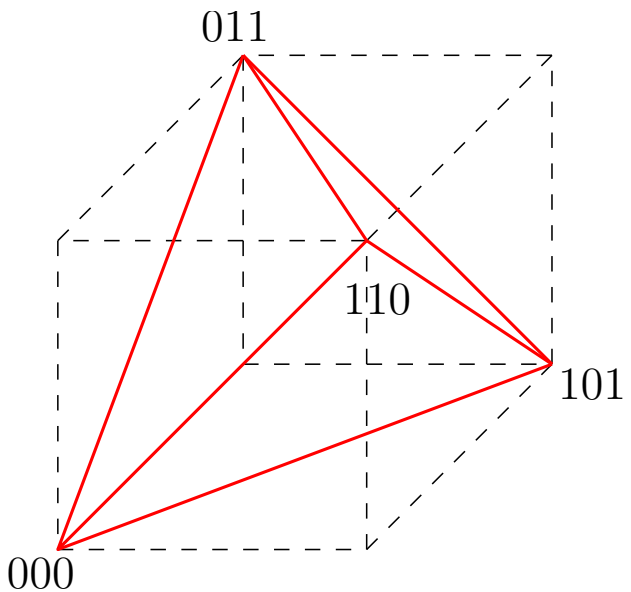
$$\begin{aligned} \text{minimize} \quad & \sum_i \gamma_i x_i \\ \text{subject to} \quad & P_j x \in \mathbb{PP}_d \quad \forall j \text{ and } x \in [0,1]^n \end{aligned}$$

Decoding LP and the Parity Polytope

- $\mathbb{P}_d = \{\text{all even parity bit-vectors of length } d\}$
- $\mathbb{PP}_d = \text{conv}(\mathbb{P}_d)$

$$\text{minimize} \quad \sum_i \gamma_i x_i$$

$$\text{subject to} \quad P_j x \in \mathbb{PP}_d \quad \forall j \text{ and } x \in [0,1]^n$$



Outline

- Background and Problem Setup [6 mins.]
  LP Decoding Formulation

Outline

- Background and Problem Setup [6 mins.]
  LP Decoding Formulation
- Optimization Framework [4 mins.]
  Alternating Direction Method of Multipliers (ADMM)

Decoding LP with parity polytope $\mathbb{PP}_d$:

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

Add Replicas z_js:

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

Augmented Lagrangian

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

Augmented Lagrangian with Lagrange multiplier λ and penalty parameter μ:

$$L_\mu(x, z, \lambda) := \gamma^T x + \sum_j \lambda_j^T (P_j x - z_j)$$

Augmented Lagrangian

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

Augmented Lagrangian with Lagrange multiplier λ and penalty parameter μ:

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

Alternating Direction Method of Multipliers

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

ADMM Update Steps: Lather, Rinse, Repeat.

$$x^{k+1} := \text{argmin}_{x \in \mathcal{X}} L_\mu(x, z^k, \lambda^k)$$
$$z^{k+1} := \text{argmin}_{z \in \mathcal{Z}} L_\mu(x^{k+1}, z, \lambda^k)$$
$$\lambda_j^{k+1} := \lambda_j^k + \mu \left(P_j x^{k+1} - z_j^{k+1} \right)$$

ADMM x-Update

- Decoding LP:

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

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

ADMM x-Update

- Decoding LP:

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

- Augmented Lagrangian:
  $L_\mu(x,z,\lambda) := \gamma^T x + \sum_j \lambda_j^T (P_j x - z_j) + \frac{\mu}{2} \sum_j \|P_j x - z_j\|_2^2$
- With z and λ fixed the x-updates are simple:

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

ADMM x-Update

In the x-update step replicas (z) and dual variables (λ) are fixed.

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

Component wise update:

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

$\mathcal{N}_v(i)$: set of parity checks containing component i.
$z_j^{(i)}$: component of the jth replica associated with x_i.

ADMM z-Update

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

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

- The minimization is completely separable in j, hence for each z_j we need to solve:

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

- z-updates:

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

- With $v = P_j x + \lambda_j / \mu$ (completing the square) the problem is equivalent to:

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

The primary challenge in ADMM

The z-update requires *projecting onto the parity polytope*.

Outline

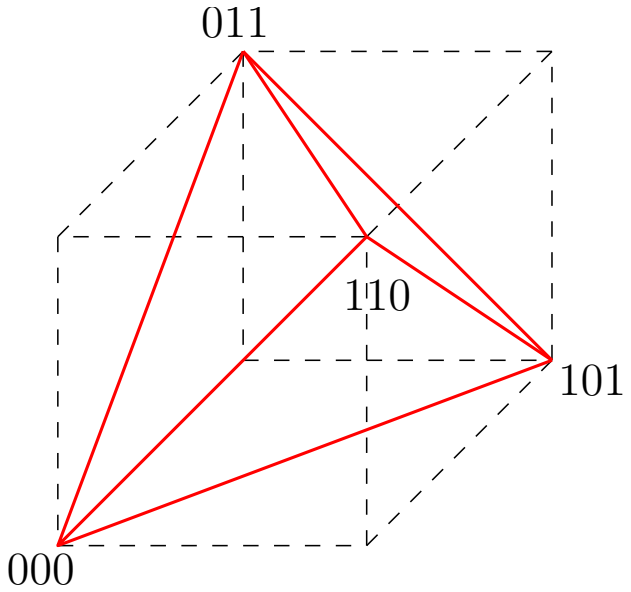
- Background and Problem Setup [6 mins.]
  LP Decoding Formulation
- Optimization Framework [4 mins.]
  Alternating Direction Method of Multipliers (ADMM)

Outline

- Background and Problem Setup [6 mins.]
  LP Decoding Formulation
- Optimization Framework [4 mins.]
  Alternating Direction Method of Multipliers (ADMM)
- Technical Core [$5 + \epsilon$ mins.]
  Projecting onto the Parity Polytope

$$\begin{array}{ll} \text{minimize} & \|v - \tilde{z}\|_2^2 \\ \text{subject to} & \tilde{z} \in \mathbb{PP}_d \end{array}$$

Recall that the *parity polytope*, $\mathbb{PP}_d$, is the convex hull of all binary vectors of length d and even hamming weight.



Parity Polytope

- $y \in \mathbb{PP}_d$ iff $y = \sum_i \alpha_i e_i$
- Here e_i are even-hamming-weight vectors of dimension d, $\sum_i \alpha_i = 1$ and $\alpha_i \geq 0$.
- Example ($d = 6$):

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

Parity Polytope

- $y \in \mathbb{PP}_d$ iff $y = \sum_i \alpha_i e_i$
- Here e_i are even-hamming-weight vectors of dimension d, $\sum_i \alpha_i = 1$ and $\alpha_i \geq 0$.
- Example ($d = 6$):

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

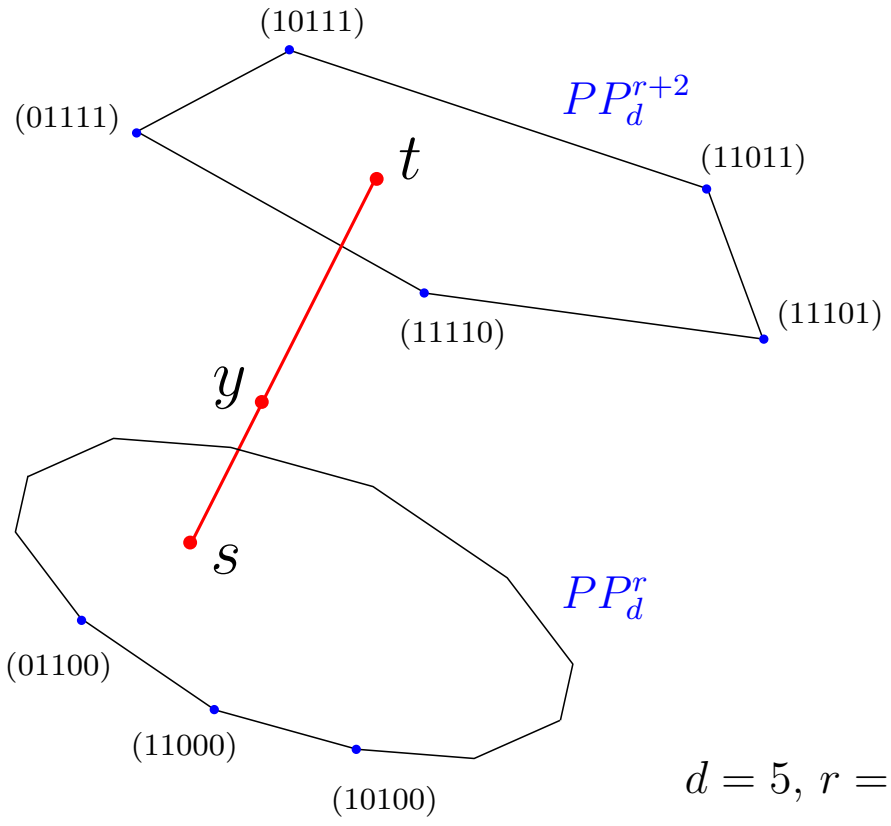
Characterizing the Parity Polytope

Two-Slice Lemma: For any $y \in \mathbb{PP}_d$ there exists a representation: $y = \sum_i \alpha_i e_i$ such that the hamming weight of all e_i is either r or $r+2$, for some even integer r.
Example with $d = 6$ and $r = 2$.

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

Characterizing the Parity Polytope

Lemma: For any $y \in \mathbb{PP}_d$ there exists a representation: $y = \sum_i \alpha_i e_i$ such that the hamming weight of all e_i is either r or $r+2$, for some even integer r.

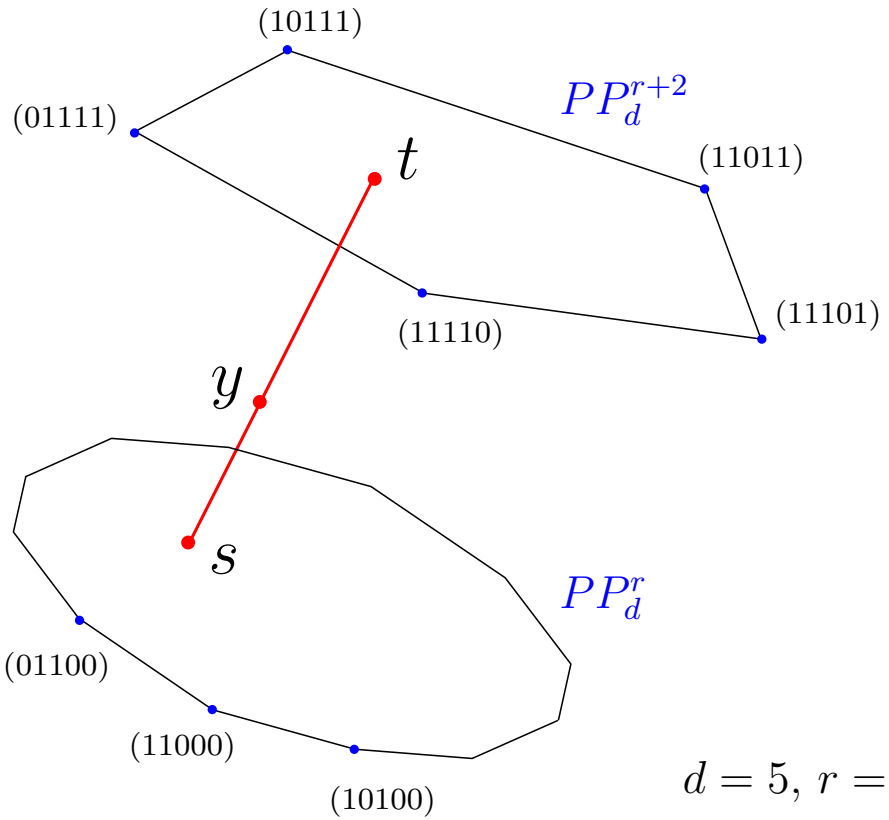


$d = 5,\ r = 2$

Characterizing the Parity Polytope

Let $\mathbb{PP}_d^r$ be the convex hull of all binary vectors of hamming weight r.

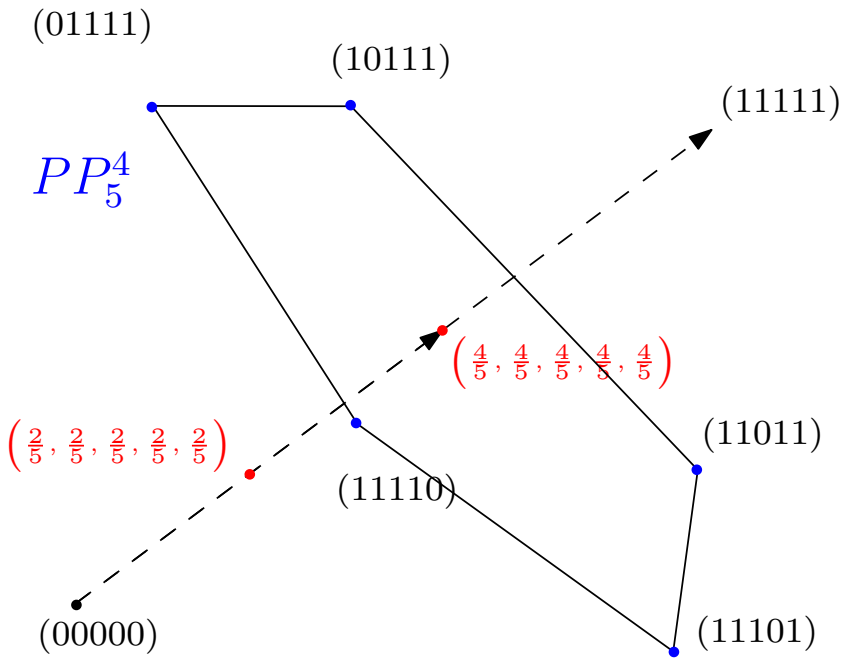
Two-Slice Lemma: $y \in \mathbb{PP}_d$ iff $y = \alpha s + (1-\alpha)t$, where $s \in \mathbb{PP}_d^r$ and $t \in \mathbb{PP}_d^{r+2}$.



$d = 5,\ r = 2$

Structure of the Parity Polytope

- $\mathbb{PP}_d^r$ = (Hyperplane containing vectors of weight r) $\bigcap$ Hypercube.



- Any $u \in \mathbb{PP}_d$ sandwiched between slices of weight r and $r+2$ where $r \leq \|u\|_1 \leq r+2$.

Projecting onto the parity polytope

Two-Slice Lemma: $y \in \mathbb{PP}_d$ iff $y = \alpha s + (1-\alpha)t$, where $s \in \mathbb{PP}_d^r$ and $t \in \mathbb{PP}_d^{r+2}$.

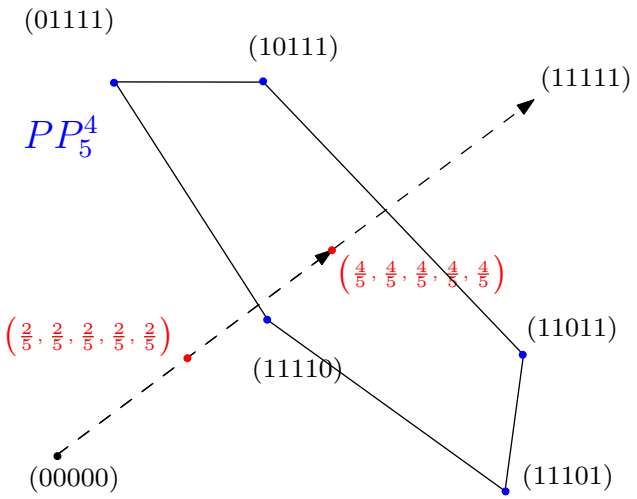
$$\begin{aligned} \text{Polytope Projection:} \quad & \min \|v - y\|_2^2 \\ & \text{s.t. } y \in \mathbb{PP}_d \\ \text{From the two-slice lemma:} \quad & \min \|v - (\alpha s + (1-\alpha)t)\|_2^2 \\ & \text{s.t. } s \in \mathbb{PP}_d^r, t \in \mathbb{PP}_d^{r+2} \\ & \quad \alpha \in [0,1] \end{aligned}$$

Majorization

Let u and w be d-vectors sorted in decreasing order. The vector w is said to majorize u if $\|w\|_1 = \|u\|_1$ and

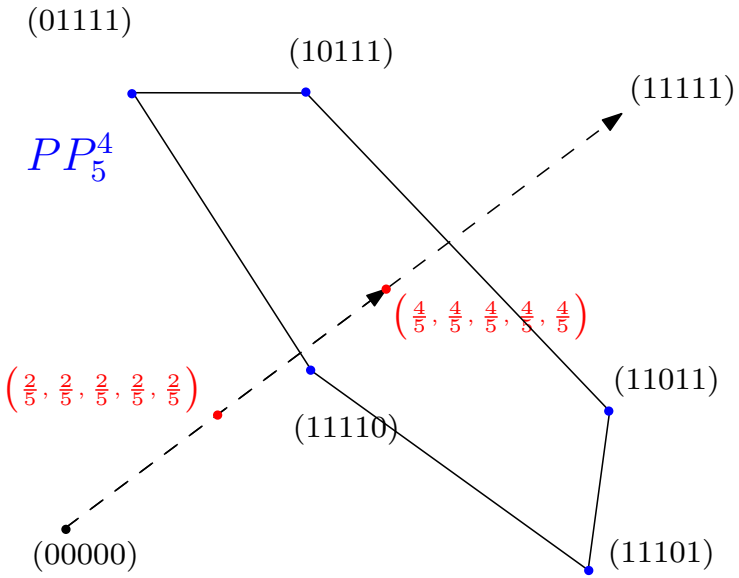
$$\sum_{k=1}^{q} u_k \leq \sum_{k=1}^{q} w_k \quad \forall q$$

Vector $(1,1,1,1,0)$ majorizes $\left(\frac{4}{5}, \frac{4}{5}, \frac{4}{5}, \frac{4}{5}, \frac{4}{5}\right)$.



Majorization

Theorem: u is in the convex hull of all permutations of w if and only if w majorizes u.
Vector $(1, 1, 1, 1, 0)$ majorizes $\left(\frac{4}{5}, \frac{4}{5}, \frac{4}{5}, \frac{4}{5}, \frac{4}{5}\right)$.



Projecting onto the parity polytope

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

From the two-slice lemma:
$$\min \|v - (\alpha s + (1-\alpha)t)\|_2^2$$
$$\text{s.t. } s \in \mathbb{PP}_d^r, t \in \mathbb{PP}_d^{r+2}$$
$$\alpha \in [0,1]$$

Using Majorization:
$$\min \|v - y\|_2^2$$
$$\text{s.t. } \sum_i y_i = \alpha r + (1-\alpha)(r+2)$$
$$\sum_{k=1}^{r+1} y_{(k)} \leq r + (1-\alpha)$$
$$0 \leq \alpha \leq 1$$

Quadratic program for the projection problem:

$$\begin{aligned} \min \ & \|v - y\|_2^2 \\ \text{s.t.} \ & \sum_i y_i = \alpha r + (1-\alpha)(r+2) \\ & \sum_{k=1}^{r+1} y_{(k)} \leq r + (1-\alpha) \\ & 0 \leq \alpha \leq 1 \end{aligned}$$

- For the quadratic program the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient.
- We develop a water-filling type algorithm which determines a solution satisfying the KKT conditions.

Overall Result

We can project onto the parity polytope in $O(d \log d)$ time.

Final Projection Algorithm

- The KKT conditions imply that either the projection of v onto the hypercube, $[0,1]^d$, is in the parity polytope, or
- There exists a $\beta^* \in \mathbb{R}_+$ such that the projection y^* satisfies:

$$y^* = v - \beta^* w$$

where $w = (\underbrace{1,\ldots,1}_{r+1},\underbrace{-1,\ldots-1}_{d-r-1})^T$.

- Using this characterization we develop a water-filling type algorithm that determines y^*, the projection onto the parity polytope.

Overall Result

We can project onto the parity polytope in $O(d \log d)$ time.

Outline

- Background and Problem Setup [6 mins.]
  LP Decoding Formulation
- Optimization Framework [4 mins.]
  ADMM
- Technical Core [$5 + \epsilon$ mins.]
  Projecting onto the Parity Polytope

Outline

- Background and Problem Setup [6 mins.]
  LP Decoding Formulation
- Optimization Framework [4 mins.]
  ADMM
- Technical Core [$5 + \epsilon$ mins.]
  Projecting onto the Parity Polytope
- Numerical results [5 mins.]
  Graphs

Implementation of ADMM decoder

Recap ADMM update steps:

$$x^{k+1} := \text{argmin}_{x \in \mathcal{X}} L_\mu(x, z^k, \lambda^k)$$
$$z^{k+1} := \text{argmin}_{z \in \mathcal{Z}} L_\mu(x^{k+1}, z, \lambda^k)$$
$$\lambda_j^{k+1} := \lambda_j^k + \mu(P_j x^{k+1} - z_j^{k+1})$$

Implementation of ADMM decoder

Recap ADMM update steps:

$$x^{k+1} := \text{argmin}_{x \in \mathcal{X}} L_\mu(x, z^k, \lambda^k)$$
$$z^{k+1} := \text{argmin}_{z \in \mathcal{Z}} L_\mu(x^{k+1}, z, \lambda^k)$$
$$\lambda_j^{k+1} := \lambda_j^k + \mu(P_j x^{k+1} - z_j^{k+1})$$

- Question: How to choose penalty parameter μ?

Implementation of ADMM decoder

Recap ADMM update steps:

$$x^{k+1} := \text{argmin}_{x \in \mathcal{X}} L_\mu(x, z^k, \lambda^k)$$
$$z^{k+1} := \text{argmin}_{z \in \mathcal{Z}} L_\mu(x^{k+1}, z, \lambda^k)$$
$$\lambda_j^{k+1} := \lambda_j^k + \mu(P_j x^{k+1} - z_j^{k+1})$$

- Question: How to choose penalty parameter μ?
- Another question: When do we terminate the iteration?

Implementation of ADMM decoder

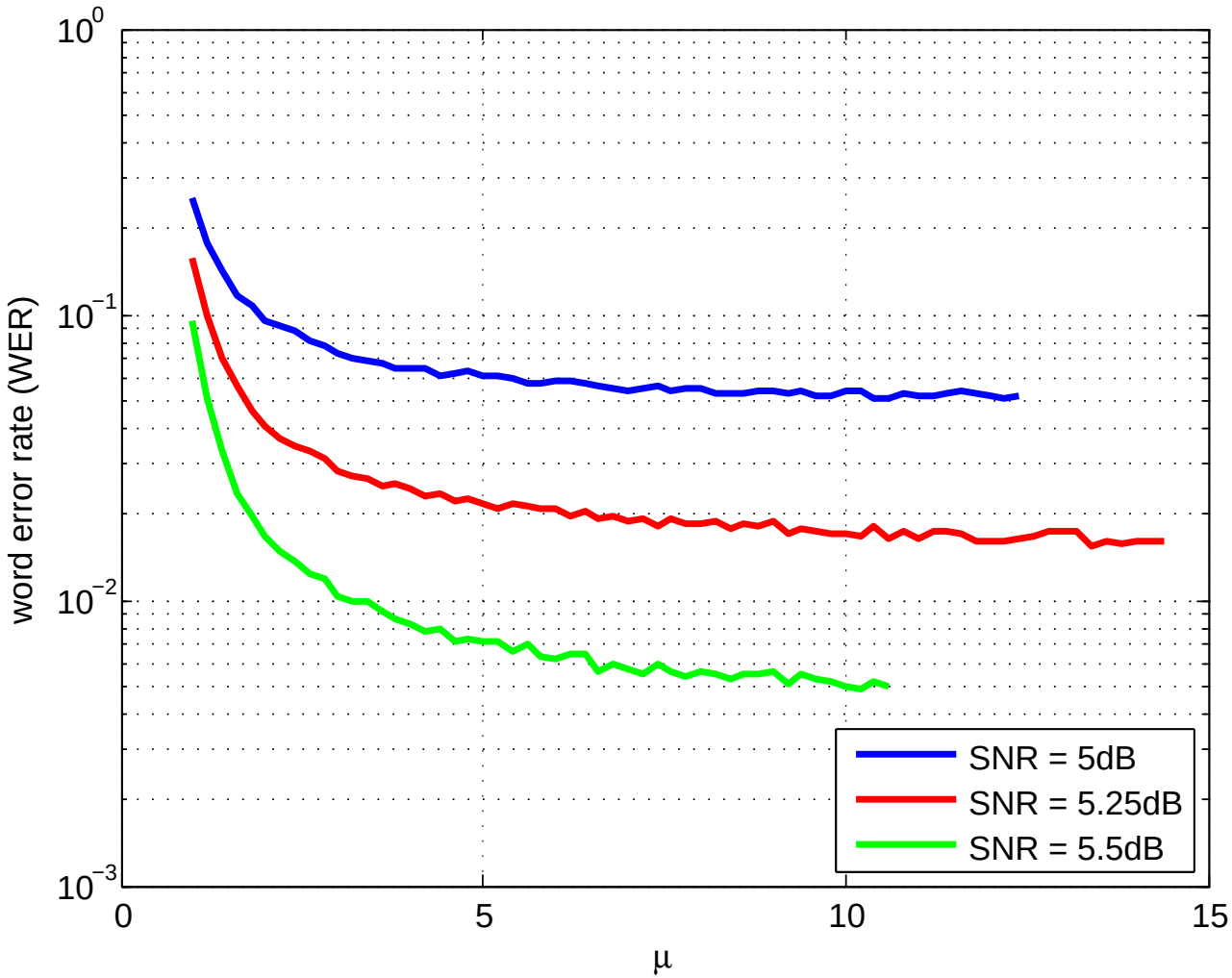
Recap ADMM update steps:

$$x^{k+1} := \text{argmin}_{x \in \mathcal{X}} L_\mu(x, z^k, \lambda^k)$$
$$z^{k+1} := \text{argmin}_{z \in \mathcal{Z}} L_\mu(x^{k+1}, z, \lambda^k)$$
$$\lambda_j^{k+1} := \lambda_j^k + \mu(P_j x^{k+1} - z_j^{k+1})$$

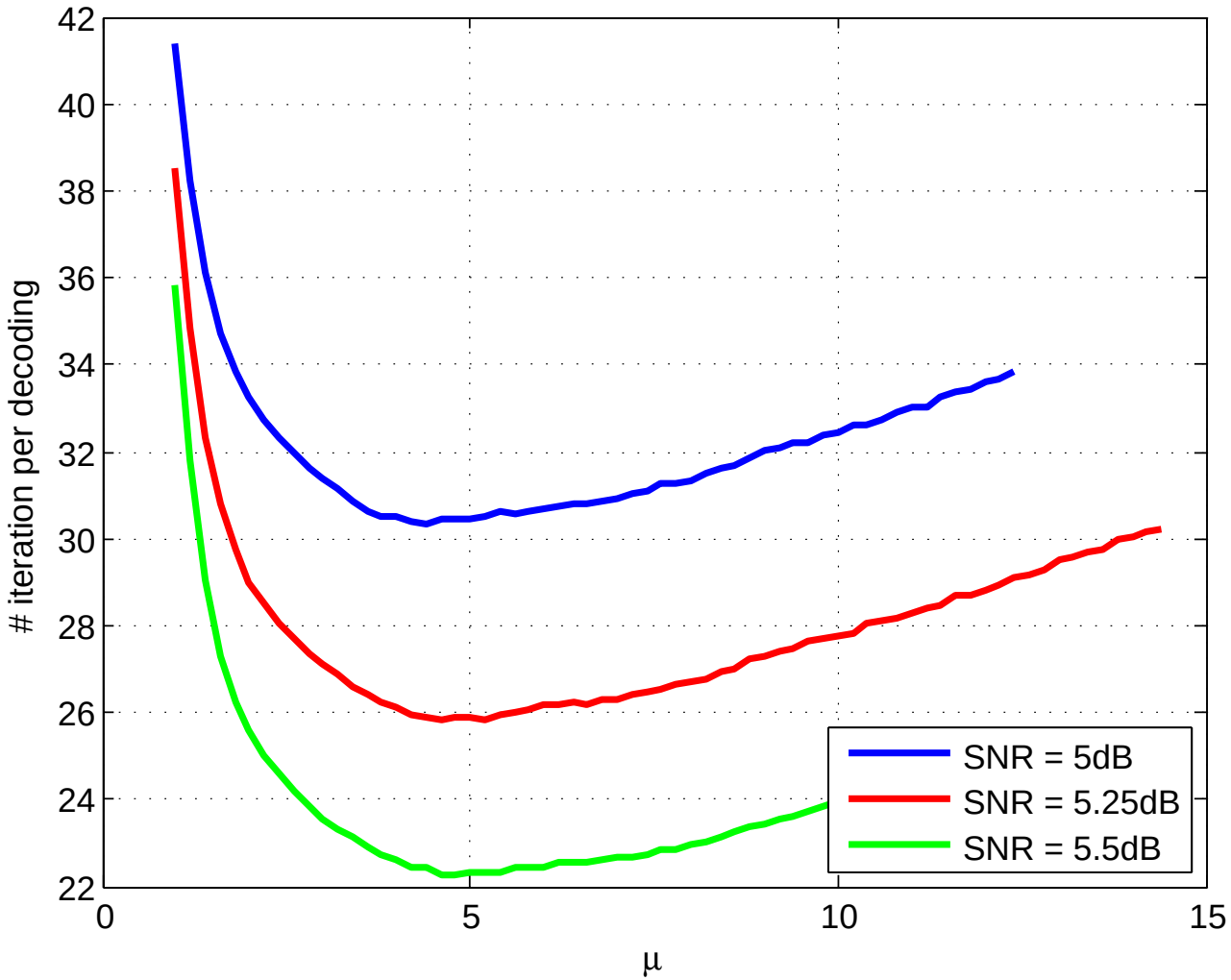
- Question: How to choose penalty parameter μ?
- Another question: When do we terminate the iteration?

Need to determine μ, maximum number of iteration $T_{\max}$ and error tolerance ϵ.

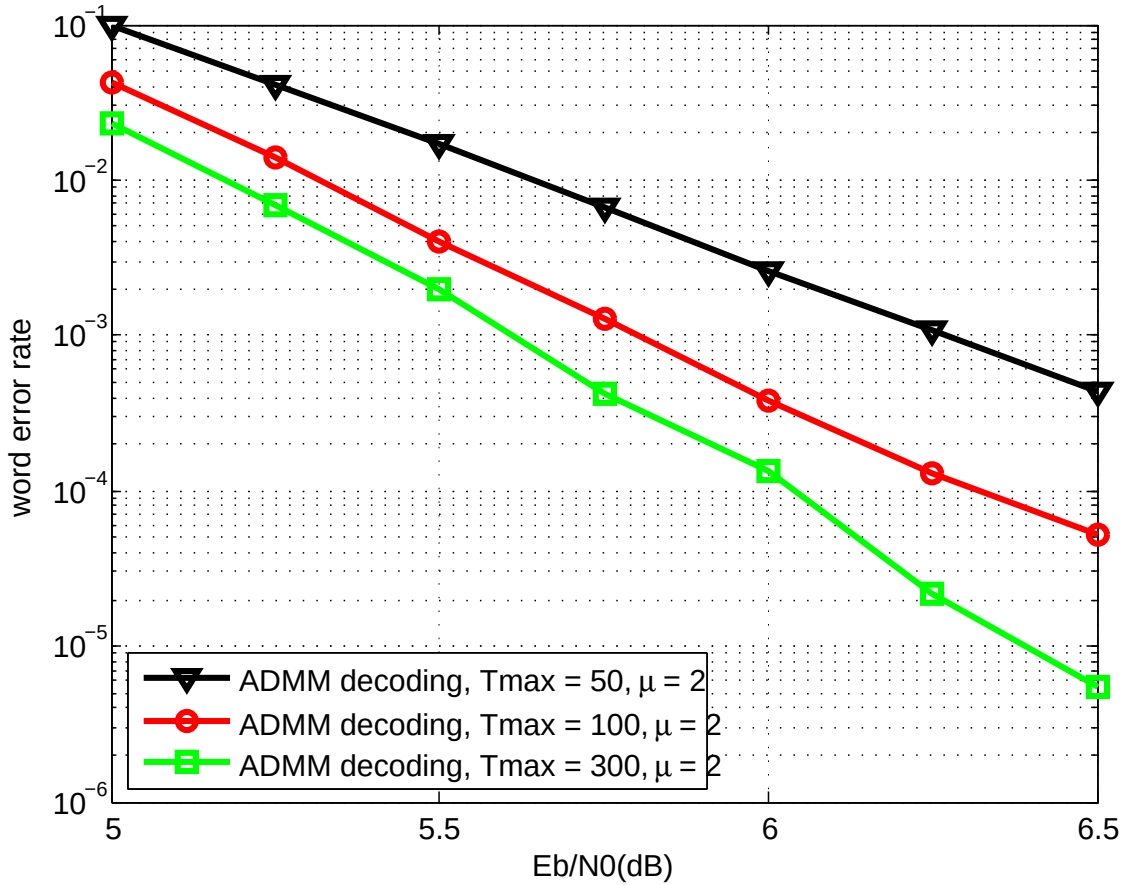
Simulation for (1057,244) LPDC code. Fix $T_{\max} = 300$, $\epsilon = 1e-4$



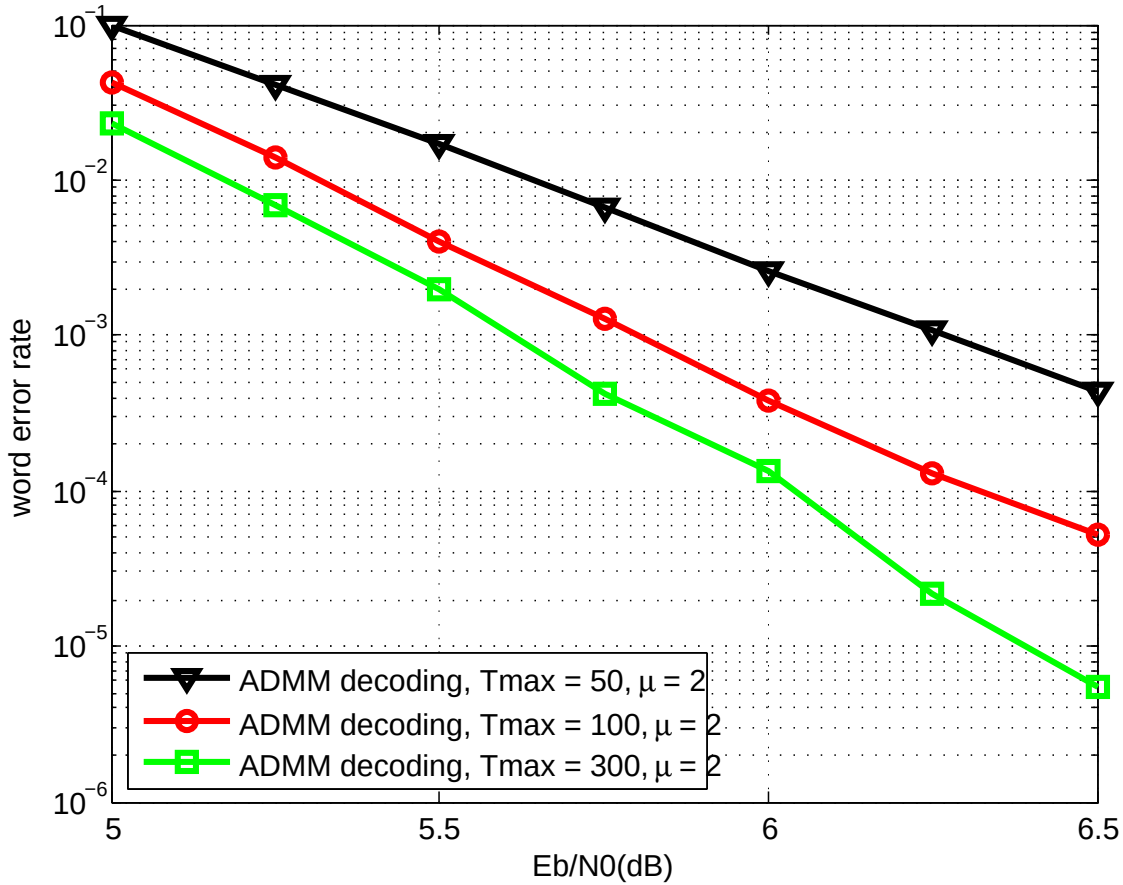
Simulation for (1057,244) LPDC code. Fix $T_{\max} = 300$, $\epsilon = 1e-4$



Simulation for (1057,244) LPDC code. $\epsilon = 1e-4$, $\mu = 2$

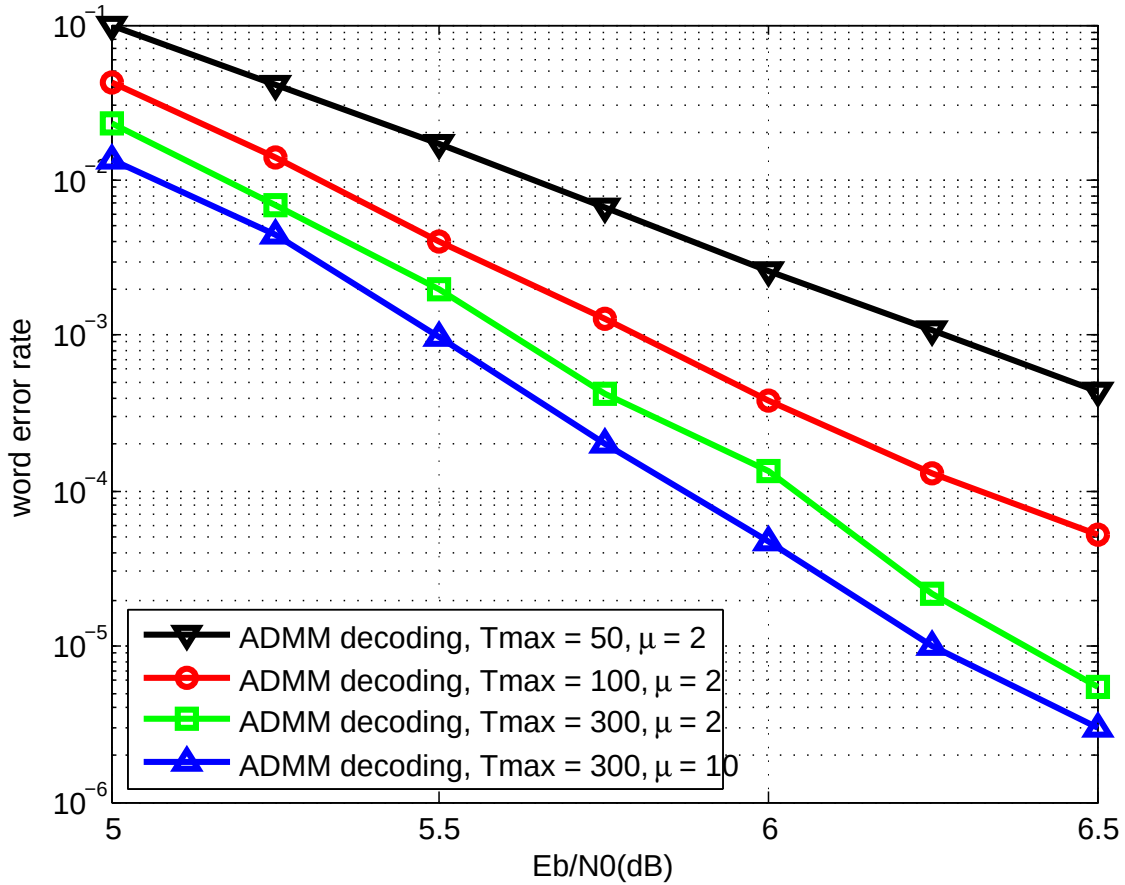


Simulation for (1057,244) LPDC code. $\epsilon = 1e-4$, $\mu = 2$

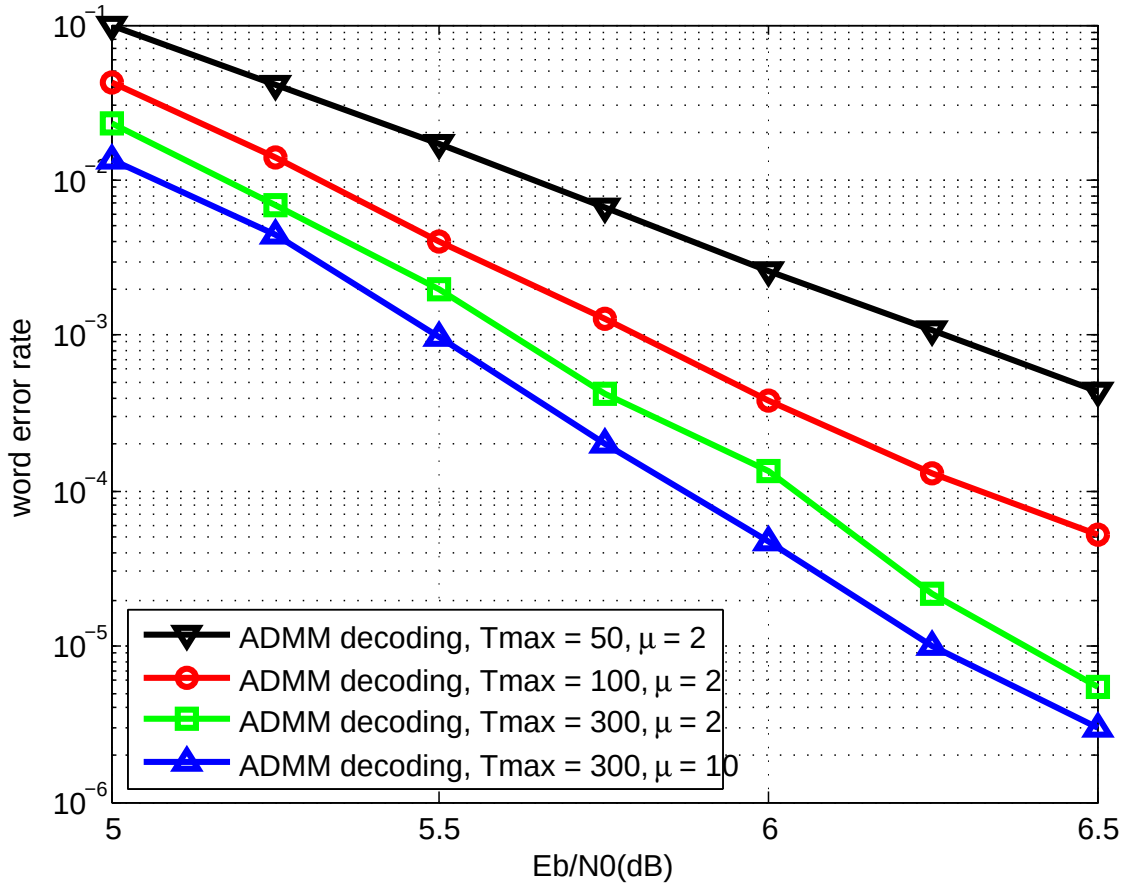


Wait! We have seen that larger μ gives better WER performance.

Simulation for (1057,244) LPDC code. $\epsilon = 1e-4$

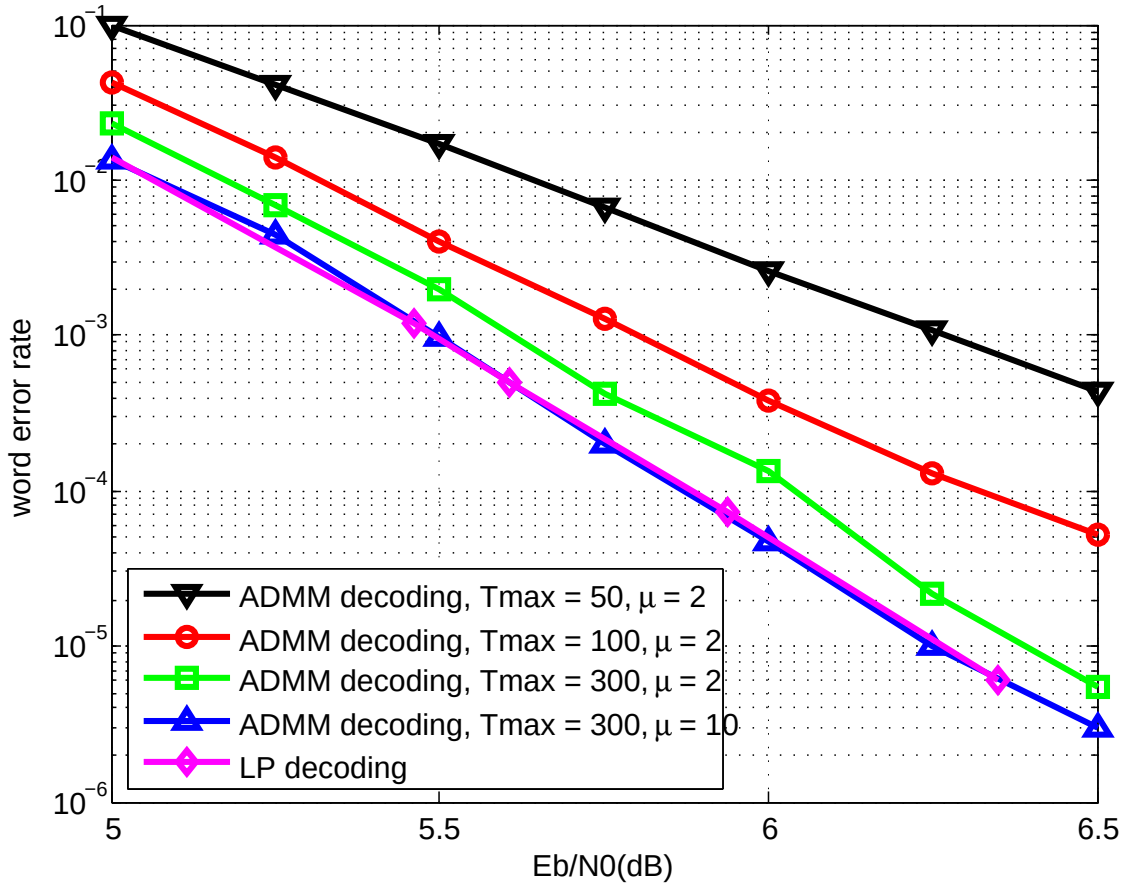


Simulation for (1057,244) LPDC code. $\epsilon = 1e-4$



Is this performance good enough?

Simulation for (1057,244) LPDC code. $\epsilon = 1e-4$



Same error performance as LP decoding using simplex method.