

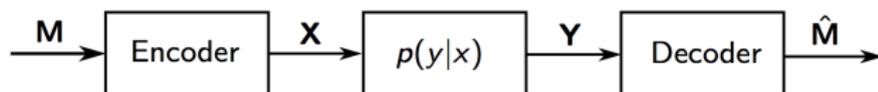
Monte Carlo Methods for Randomized Likelihood Decoding



Alankrita Bhatt, Jiun-Ting Huang, Young-Han Kim, J. Jon Ryu, and Pinar Sen

Department of Electrical and Computer Engineering
University of California, San Diego

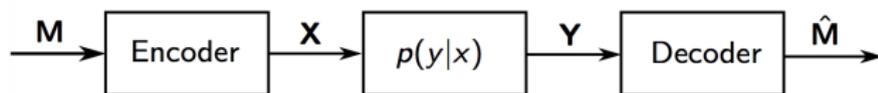
2018 Allerton Conference on Communication, Control, and Computing
October 3, 2018



- MAP decoder

$$\hat{\mathbf{m}}_{\text{MAP}}(\mathbf{y}) = \arg \max_{\mathbf{m}} p(\mathbf{m}|\mathbf{y})$$

- (+) Optimal: minimizes $P\{\hat{\mathbf{M}} \neq \mathbf{M}\}$
- (-) Exponential complexity implementation



- MAP decoder

$$\hat{\mathbf{m}}_{\text{MAP}}(\mathbf{y}) = \arg \max_{\mathbf{m}} p(\mathbf{m}|\mathbf{y})$$

- (+) Optimal: minimizes $P\{\hat{\mathbf{M}} \neq \mathbf{M}\}$
- (-) Exponential complexity implementation

- Holy grail of channel coding
 - Near-optimal performance
 - Low-complexity implementation

- Suboptimal decoding: Generate $\hat{\mathbf{M}} = \hat{\mathbf{M}}_{\text{RL}}$ according to the posterior

$$p(\mathbf{m}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}(\mathbf{m}))}{\sum_{\mathbf{m}'} p(\mathbf{y}|\mathbf{x}(\mathbf{m}'))}$$

- Suboptimal decoding: Generate $\hat{\mathbf{M}} = \hat{\mathbf{M}}_{\text{RL}}$ according to the posterior

$$p(\mathbf{m}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}(\mathbf{m}))}{\sum_{\mathbf{m}'} p(\mathbf{y}|\mathbf{x}(\mathbf{m}'))}$$

- Proof device [YAG13]: Plays a role of joint typicality decoding

$$p(\mathbf{m}|\mathbf{y}) = \frac{2^{i(\mathbf{y};\mathbf{x}(\mathbf{m}))}}{\sum_{\mathbf{m}'} 2^{i(\mathbf{y};\mathbf{x}(\mathbf{m}'))}}$$

where

$$i(\mathbf{y}; \mathbf{x}) = \log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})}$$

- Suboptimal decoding: Generate $\hat{\mathbf{M}} = \hat{\mathbf{M}}_{\text{RL}}$ according to the posterior

$$p(\mathbf{m}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}(\mathbf{m}))}{\sum_{\mathbf{m}'} p(\mathbf{y}|\mathbf{x}(\mathbf{m}'))}$$

- Proof device [YAG13]: Plays a role of joint typicality decoding

$$p(\mathbf{m}|\mathbf{y}) = \frac{2^{i(\mathbf{y};\mathbf{x}(\mathbf{m}))}}{\sum_{\mathbf{m}'} 2^{i(\mathbf{y};\mathbf{x}(\mathbf{m}'))}}$$

where

$$i(\mathbf{y}; \mathbf{x}) = \log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})}$$

- Provable error bound [KKM⁺16, LCV17, BHK⁺18]

$$P\{\hat{\mathbf{M}}_{\text{RL}} \neq \mathbf{M}\} \leq 2P\{\hat{\mathbf{M}}_{\text{MAP}} \neq \mathbf{M}\}$$

- Suboptimal decoding: Generate $\hat{\mathbf{M}} = \hat{\mathbf{M}}_{\text{RL}}$ according to the posterior

$$p(\mathbf{m}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}(\mathbf{m}))}{\sum_{\mathbf{m}'} p(\mathbf{y}|\mathbf{x}(\mathbf{m}'))}$$

- Proof device [YAG13]: Plays a role of joint typicality decoding

$$p(\mathbf{m}|\mathbf{y}) = \frac{2^{i(\mathbf{y};\mathbf{x}(\mathbf{m}))}}{\sum_{\mathbf{m}'} 2^{i(\mathbf{y};\mathbf{x}(\mathbf{m}'))}}$$

where

$$i(\mathbf{y}; \mathbf{x}) = \log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})}$$

- Provable error bound [KKM⁺16, LCV17, BHK⁺18]

$$P\{\hat{\mathbf{M}}_{\text{RL}} \neq \mathbf{M}\} \leq 2P\{\hat{\mathbf{M}}_{\text{MAP}} \neq \mathbf{M}\}$$

- **Near-optimal performance** requirement is satisfied

How to efficiently generate a sample from $p(\mathbf{m}|\mathbf{y})$?

- Due to normalization, $p(\mathbf{m}|\mathbf{y})$ is rather hard to compute
- For most channels, computing the likelihood $p(\mathbf{y}|\mathbf{x}(\mathbf{m}))$ is straightforward
- Most Monte Carlo (MC) sampling algorithms rely only on this unnormalized posterior

How to efficiently generate a sample from $p(\mathbf{m}|\mathbf{y})$?

- Due to normalization, $p(\mathbf{m}|\mathbf{y})$ is rather hard to compute
- For most channels, computing the likelihood $p(\mathbf{y}|\mathbf{x}(\mathbf{m}))$ is straightforward
- Most Monte Carlo (MC) sampling algorithms rely only on this unnormalized posterior

Markov Chain Monte Carlo decoding

- Run an ergodic Markov chain on the message space (or the codeword space)
- Distribution of the Markov chain should converge to the posterior $p(\mathbf{m}|\mathbf{y})$
- Pioneered by Neal [Nea01] (see also [MM09])
- **Speed of convergence** is the main computational bottleneck

We present several MC or MCMC methods to implement RL decoding

- Rejection sampling
- Gibbs sampling
- Metropolis sampling

To demonstrate the performance of these MC decoders, the following toy example is used

- Code: A (40,20) irregular LDPC code (10^6 messages)
- Channel: BSC with crossover probability $p = 0.04$

Main idea

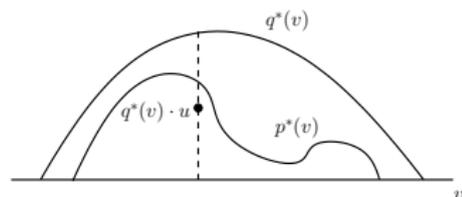
- Target pmf $p(v) = p^*(v)/Z_p$ is difficult to sample from
- Find another pmf $q(v) = q^*(v)/Z_q$ such that
 - $q(v)$ is easy to sample from
 - The ratio $p^*(v)/q^*(v) \leq 1$ and is easy to compute

Main idea

- Target pmf $p(v) = p^*(v)/Z_p$ is difficult to sample from
- Find another pmf $q(v) = q^*(v)/Z_q$ such that
 - $q(v)$ is easy to sample from
 - The ratio $p^*(v)/q^*(v) \leq 1$ and is easy to compute

Algorithm — at the t^{th} iteration

- (1) Sample $V^{(t)} \sim q(v)$
- (2) Draw $U^{(t)} \sim \text{Unif}([0, 1]) \perp\!\!\!\perp V^{(1)}, V^{(2)}, \dots, V^{(t)}$
- (3) If $U^{(t)} \leq p^*(v)/q^*(v)$, output $V^{(t)}$ and stop;
otherwise, reject $V^{(t)}$ and repeat from (1) for $V^{(t+1)}$

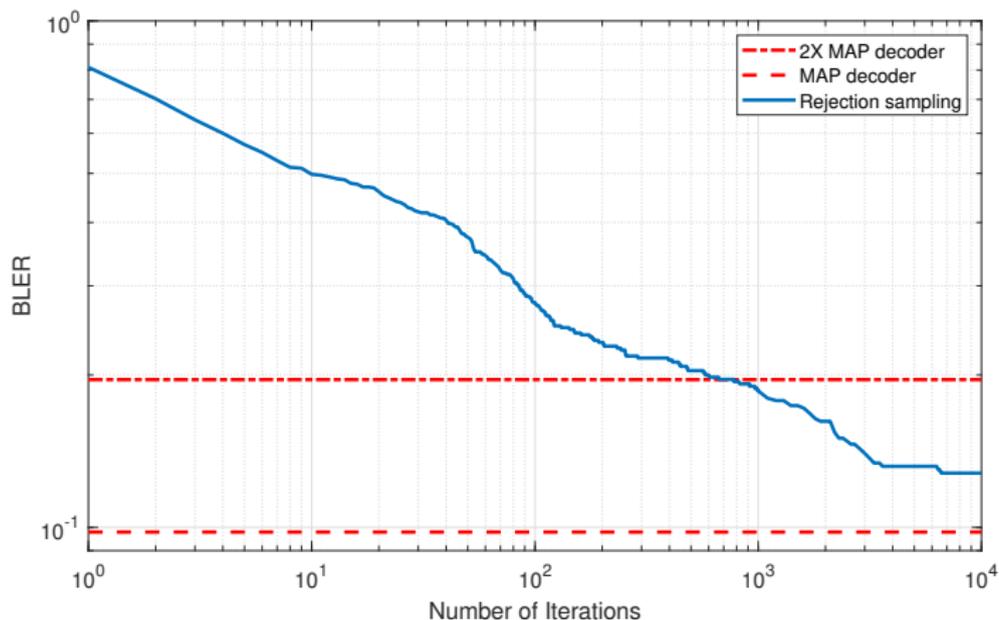


- If τ is the stopping time, then $V^{(\tau)} \sim p(v)$ and $\mathbf{E}[\tau] = Z_p/Z_q \ (\sim |\mathcal{V}|)$

Rejection Sampling Decoder — Implementation

- Decode the channel output $\mathbf{y} = [\underbrace{\mathbf{y}_1}_{k \text{ bits}} \quad \underbrace{\mathbf{y}_2}_{n-k \text{ bits}}]$ of a systematic linear code
- For $p_{\mathbf{y}}^*(\mathbf{m}) = p(\mathbf{y}|\mathbf{x}(\mathbf{m})) = (\rho/(1-\rho))^{d_H(\mathbf{x}(\mathbf{m}),\mathbf{y})} (1-\rho)^n$, choose

$$q_{\mathbf{y}}^*(\mathbf{m}) = (\rho/(1-\rho))^{d_H(\mathbf{m},\mathbf{y}_1)} (1-\rho)^n, \quad Z_q = (1-\rho)^{n-k}$$



Main idea

- Target k -variate pmf $p(\mathbf{v})$ is difficult to sample from
- Sampling from the conditional marginal distribution $p(v_i | \mathbf{v}_{-i})$ may be easier
- Run a Markov chain based on **coordinate-wise sampling**

Main idea

- Target k -variate pmf $p(\mathbf{v})$ is difficult to sample from
- Sampling from the conditional marginal distribution $p(v_i | \mathbf{v}_{-i})$ may be easier
- Run a Markov chain based on **coordinate-wise sampling**

Algorithm — at the t^{th} iteration

v_1	v_2	...	?	...	v_k
			i^{th}		

- (1) Randomly pick a coordinate $i \sim \text{Unif}[k]$
- (2) Sample the i -th coordinate $V_i^{(t)} \sim p(v_i | \mathbf{v}_{-i}^{(t-1)})$
- (3) Fix the other coordinates $v_j^{(t)} = v_j^{(t-1)}, j \in [k] \setminus \{i\}$, and repeat from (1) for $V^{(t+1)}$

Main idea

- Target k -variate pmf $p(\mathbf{v})$ is difficult to sample from
- Sampling from the conditional marginal distribution $p(v_i | \mathbf{v}_{-i})$ may be easier
- Run a Markov chain based on **coordinate-wise sampling**

Algorithm — at the t^{th} iteration

v_1	v_2	...	?	...	v_k
			i^{th}		

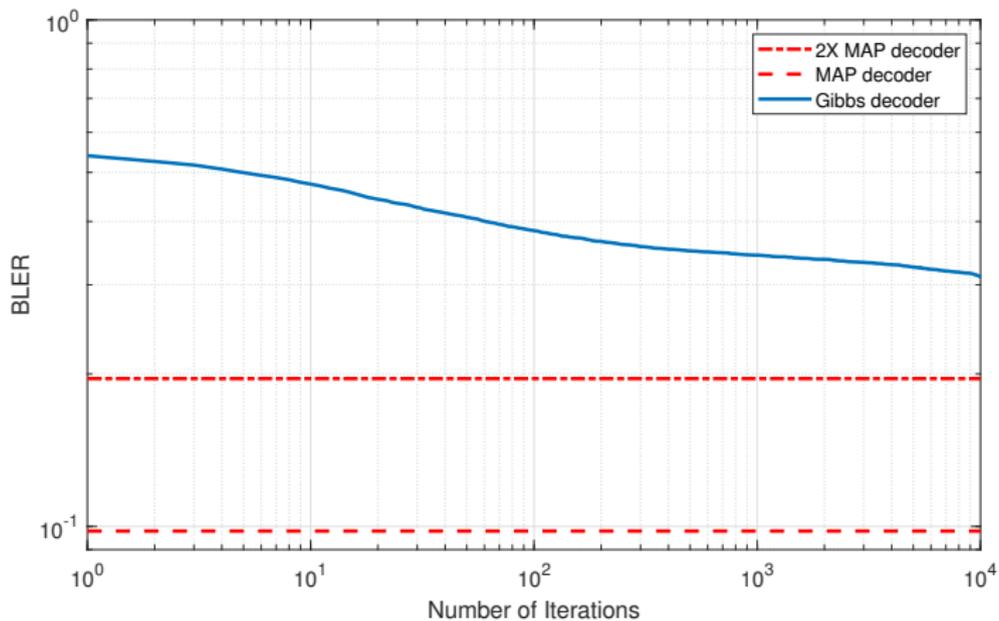
- (1) Randomly pick a coordinate $i \sim \text{Unif}[k]$
- (2) Sample the i -th coordinate $V_i^{(t)} \sim p(v_i | \mathbf{v}_{-i}^{(t-1)})$
- (3) Fix the other coordinates $v_j^{(t)} = v_j^{(t-1)}$, $j \in [k] \setminus \{i\}$, and repeat from (1) for $V^{(t+1)}$

Connection between Gibbs and BP decoding

- Both involve iterative message passing between check nodes and variable nodes
- The update rule for BP is deterministic, whereas the one for Gibbs is random
- BP is heuristic, whereas Gibbs is asymptotically exact

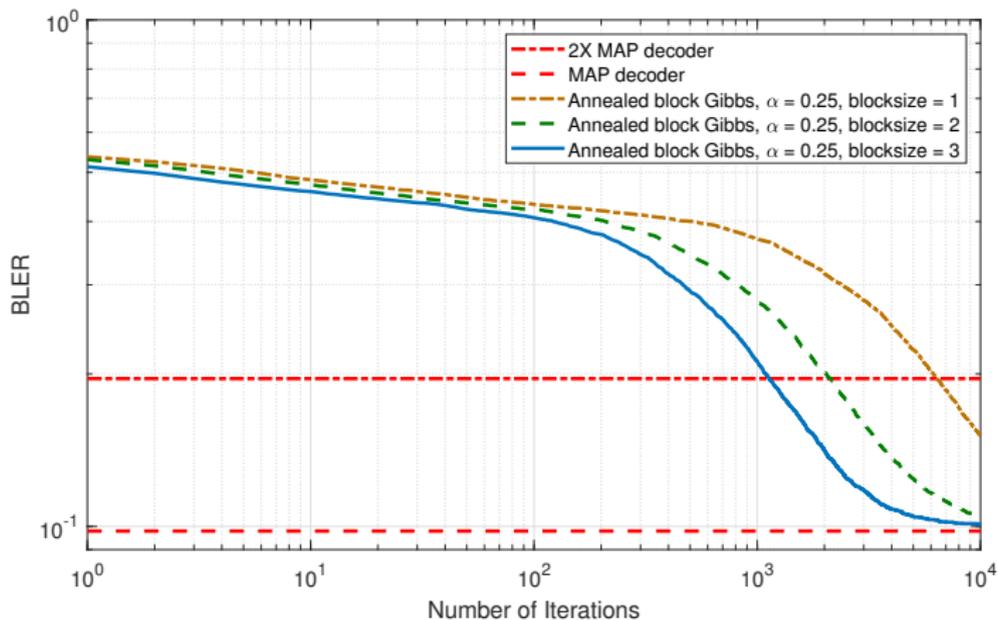
Gibbs Decoder — Implementation

- Target pmf: $p(\mathbf{m}|\mathbf{y})$
- At the t^{th} iteration, generate $M_i^{(t)} \sim p(m_i|\mathbf{y}, \mathbf{m}_{-i}^{(t-1)})$



Techniques for speeding up Gibbs sampling

- Temperature control: Sample from the annealed pmf $p_\alpha(\mathbf{m}|\mathbf{y}) \propto p^\alpha(\mathbf{m}|\mathbf{y})$
- Soft parity constraint [Nea01]: Perform random walk on the codeword space
- Block sampling: Update a set B of coordinates at once $M_B^{(t)} \sim p(m_B|\mathbf{y}, \mathbf{m}_{-B}^{(t-1)})$



Main idea

- Find **another Markov chain** over the same state space that is easy to sample from
- Make the MC converge to the target distribution via **acceptance/rejection** steps

Main idea

- Find **another Markov chain** over the same state space that is easy to sample from
- Make the MC converge to the target distribution via **acceptance/rejection** steps

Algorithm — at the t^{th} iteration

- (1) Propose a new move $v^{(t)} \rightarrow v'$ by the underlying (reversible) MC
- (2) Accept/reject the proposal depending on likelihood ratio $p(v')/p(v^{(t)})$:
 - if $p(v')/p(v^{(t)}) \geq 1$, set $v^{(t+1)} = v'$;
 - if $p(v')/p(v^{(t)}) < 1$, set $v^{(t+1)} = v'$ with probability; $p(v')/p(v^{(t)})$
 - otherwise, fix $v^{(t+1)} = v^{(t)}$
- (3) Repeat from (1) for $V^{(t+1)}$

Main idea

- Find **another Markov chain** over the same state space that is easy to sample from
- Make the MC converge to the target distribution via **acceptance/rejection** steps

Algorithm — at the t^{th} iteration

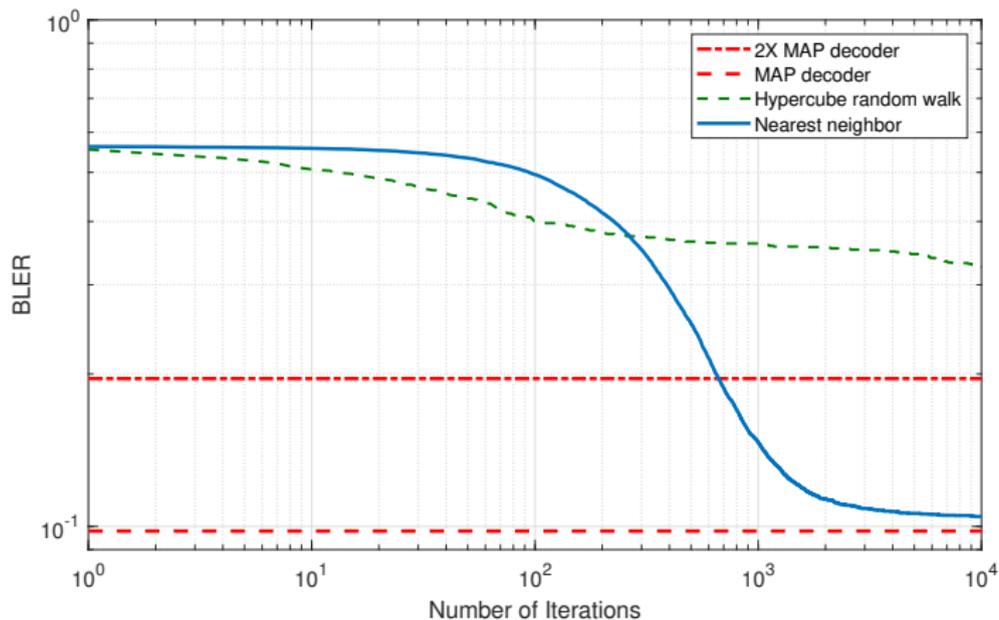
- (1) Propose a new move $v^{(t)} \rightarrow v'$ by the underlying (reversible) MC
- (2) Accept/reject the proposal depending on likelihood ratio $p(v')/p(v^{(t)})$:
 - if $p(v')/p(v^{(t)}) \geq 1$, set $v^{(t+1)} = v'$;
 - if $p(v')/p(v^{(t)}) < 1$, set $v^{(t+1)} = v'$ with probability; $p(v')/p(v^{(t)})$
 - otherwise, fix $v^{(t+1)} = v^{(t)}$
- (3) Repeat from (1) for $V^{(t+1)}$

Connection to other sampling methods

- Rejection sampling: Close to slice sampling, a special case of Metropolis
- Gibbs sampling: Irrejectable proposal based on the target conditional marginal pmf

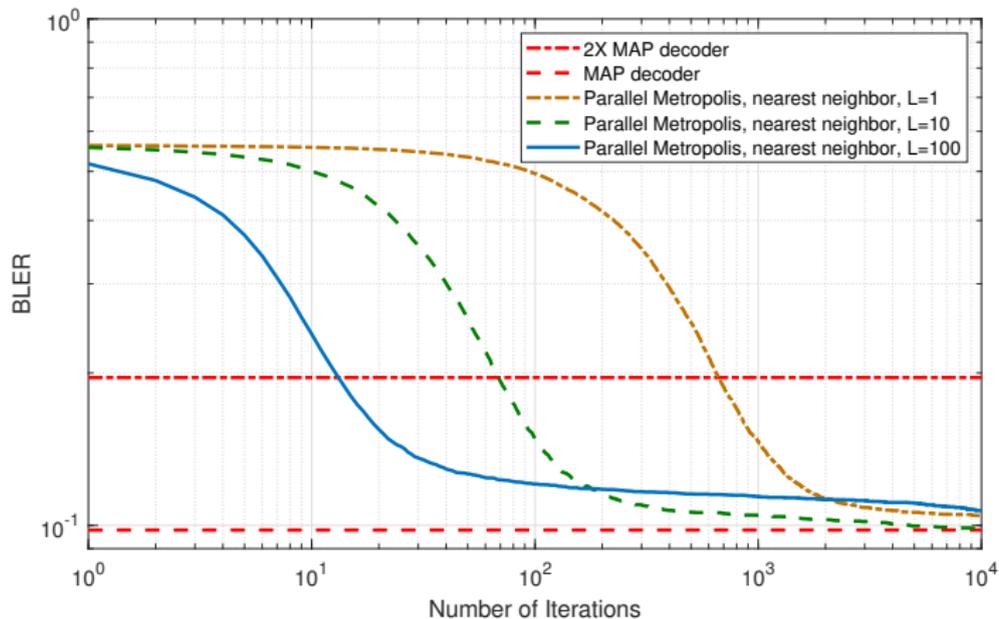
Metropolis Decoder — Implementation

- Target pmf: $p(\mathbf{m}|\mathbf{y})$
- Choice of underlying MC strongly affects convergence time:
 - Hypercube
 - I.I.D. (slice sampling)
 - Nearest neighbor



Techniques for speeding up Metropolis sampling

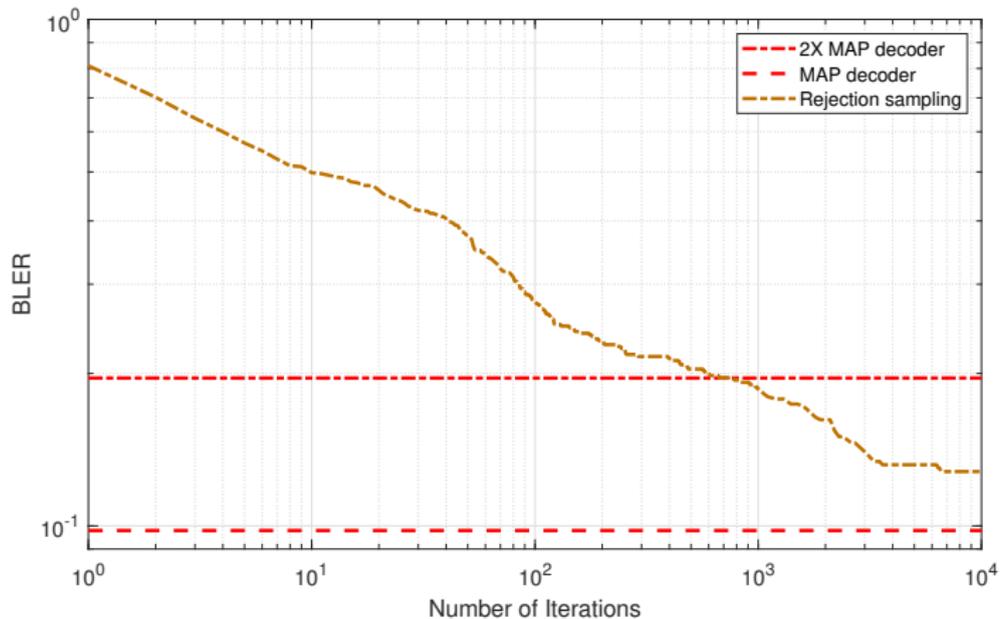
- Temperature control (annealing) and soft parity
- Parallelization: At each iteration
 - (1) Take $L > 1$ samples from the underlying MC
 - (2) Propose the most probable sample \mathbf{m}' for accept/reject



- Monte Carlo methods for RL decoder

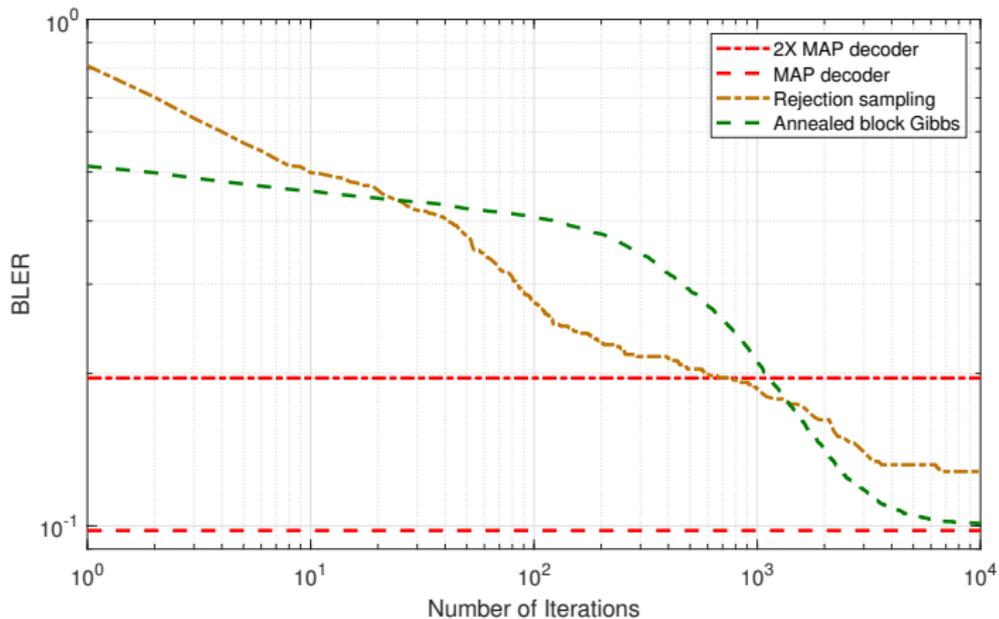
Concluding Remarks

- Monte Carlo methods for RL decoder



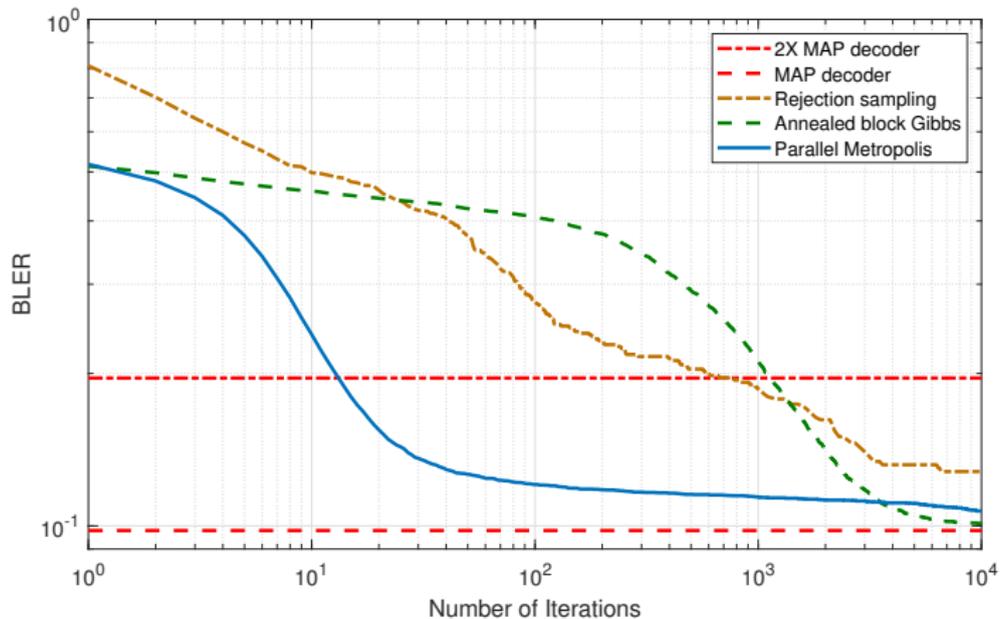
Concluding Remarks

- Monte Carlo methods for RL decoder



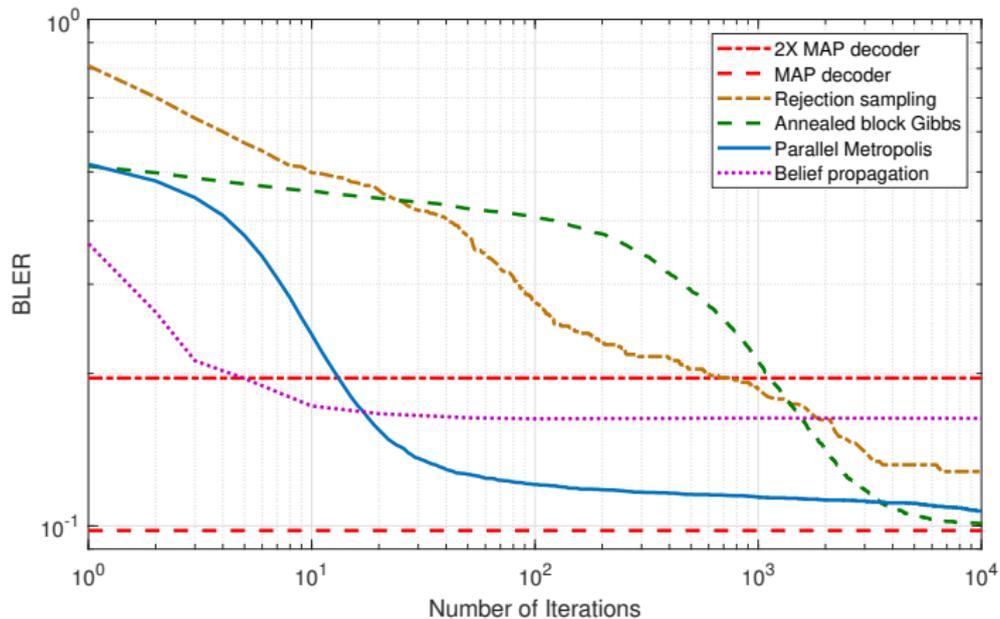
Concluding Remarks

- Monte Carlo methods for RL decoder



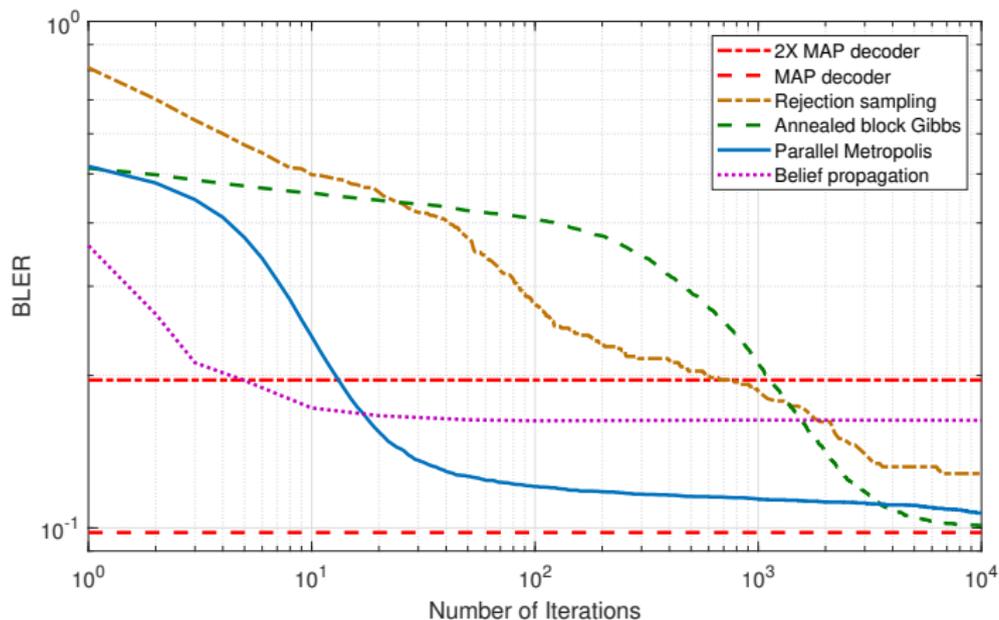
Concluding Remarks

- Monte Carlo methods for RL decoder



Concluding Remarks

- Monte Carlo methods for RL decoder



- Many open problems (larger codes, further speedup, circuit implementation)

-  Alankrita Bhatt, Jiun-Ting Huang, Young-Han Kim, J. Jon Ryu, and Pinar Sen, *Variations on a theme by Liu, Cuff, and Verdú: The power of posterior sampling*, Proc. IEEE Inf. Theory Workshop, 2018.
-  S. Kudekar, S. Kumar, M. Mondelli, H.D. Pfister, and R. Urbanke, *Comparing the bit-MAP and block-MAP decoding thresholds of Reed-Muller codes on BMS channels*, Proc. IEEE Int. Symp. Inf. Theory (Barcelona, Spain), July 2016, pp. 1755–1759.
-  Jingbo Liu, Paul Cuff, and Sergio Verdú, *On α -decodability and α -likelihood decoder*, Proc. 55th Ann. Allerton Conf. Comm. Control Comput. (Monticello, IL), October 2017.
-  Marc Mezard and Andrea Montanari, *Information, Physics, and Computation*, Oxford University Press, 2009.
-  Radford M. Neal, *Monte Carlo decoding of LDPC codes*, Tech. report, Dept. of Computer Science, University of Toronto, 2001, Presented at *ICTP Works. Statis. Phys. Capacity-Approaching Codes*.
-  M. H. Yassaee, M. R. Aref, and A. Gohari, *A technique for deriving one-shot achievability results in network information theory*, Proc. IEEE Int. Symp. Inf. Theory (Istanbul, Turkey), 2013, pp. 1151–1155.