Polar Decoding on Sparse Graphs with Deep Learning

Weihong Xu[1,2], Xiaohu You[2], Chuan Zhang[1,2], Yair Be'ery[3]

[1] Lab of Efficient Architectures for Digital-communication and Signal-processing (LEADS)

[2] National Mobile Communications Research Laboratory, Southeast University, Nanjing, China

[3] School of Electrical Engineering, Tel-Aviv University, Israel

wh.xu@seu.edu.cn

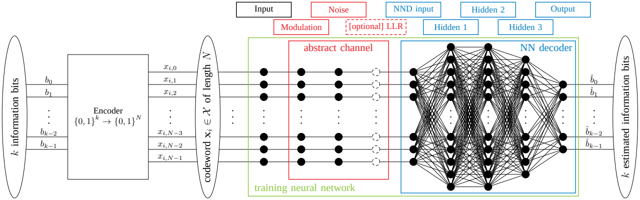
Oct. 29, 2018

Outline

Outline

1. Related Work
2. Deep Learning for Polar Codes on Sparse Graphs
3. Results and Analysis
4. Conclusion

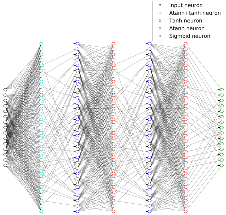
Neural Network Decoder for Polar Codes



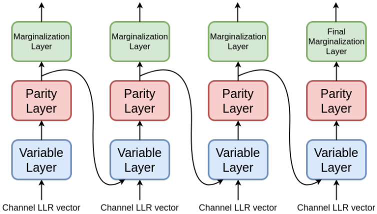
- Construct polar decoder based on fully-connected neural networks.
- **Pros & cons**:
  - ∗ Near-optimal performance for very short codes.
  - ∗ Hard to be extended to long codes.
  - ∗ Prohibitive complexity of NN inference.

[1] [Gruber, Cammerer, Hoydis, *et al.*, *CISS* 2017]

Neural Network Decoder for Linear Codes



Feed-forward network

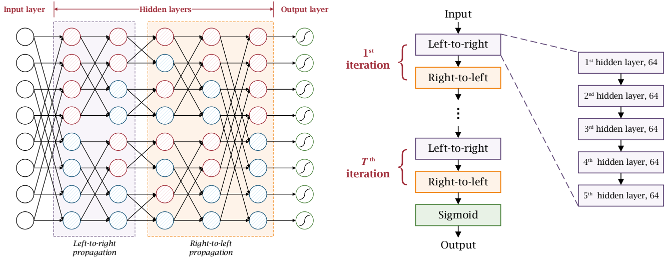


Recurrent network

- Tanner graph is unfolded into feed-forward or recurrent neural networks.
- **Pros & cons**:
  - ∗ Improving performance through training.
  - ∗ Easy to co-operate with other decoding methods, such as permutation.
  - ∗ **Complexity**: Near-BP. **Latency**: $2I$

[2][Nachmani, Marciano, Lugosch, *et al.*, *JSTSP* 2018]

Neural Network Decoder for Polar Codes



- Construct polar decoder based on factor graph.
- **Pros & cons**:
  - ∗ Near-BP performance.
  - ∗ Easy to extend.
  - ∗ **Complexity**: $\mathcal{O}(IN\log_2 N)$ with min-sum. **Latency**: $2I\log_2 N$

[3] [Xu, Wu, Ueng, *et al.*, *SiPS* 2017]

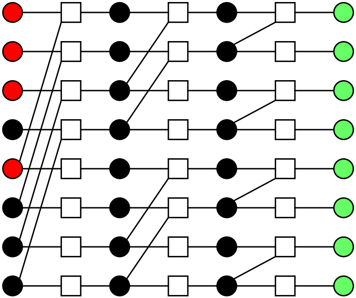
Some Questions

- Substantial weights are unfriendly for implementation.
- Is it necessary to parameterize every edge?
- Constructing neural network decoder of polar codes on Tanner graph?

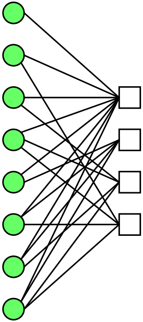
Outline

1. Related Work
2. Deep Learning for Polar Codes on Sparse Graphs
3. Results and Analysis
4. Conclusion

Two Types of Factor Graphs

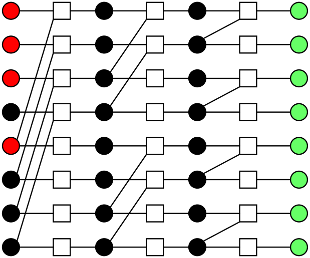


G-matirx-based



LDPC-like

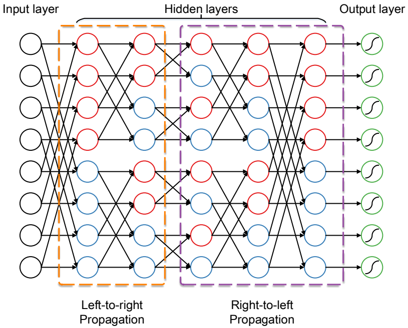
BP Decoding for Polar Codes



- **Original BP Decoding**:

$$\begin{cases} L_{i,j}^{(t)} = g(L_{i+1,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}), \\ L_{i,j+N/2^i}^{(t)} = g(L_{i+1,j}^{(t-1)}, R_{i,j}^{(t)}) + L_{i+1,j+N/2^i}^{(t-1)}, \\ R_{i+1,j}^{(t)} = g(R_{i,j}^{(t)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}), \\ R_{i+1,j+N/2^i}^{(t)} = g(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}) + R_{i,j+N/2^i}^{(t)}, \end{cases}$$

Optimization on G-matrix Factor Graph

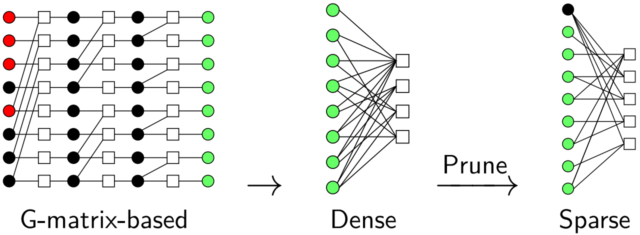


- Each node is approximated and parameterized:

$$g(x, y) = \ln \frac{1 + e^{x+y}}{e^x + e^y} \approx \alpha_{i,j} \times \mathsf{sign}(x)\mathsf{sign}(y) \times \min(|x|, |y|).$$

[3][Xu, Wu, Ueng, *et al.*, *SiPS* 2017]

BP Decoding for Polar Codes



▷ **BP Decoding on Tanner Graph**:

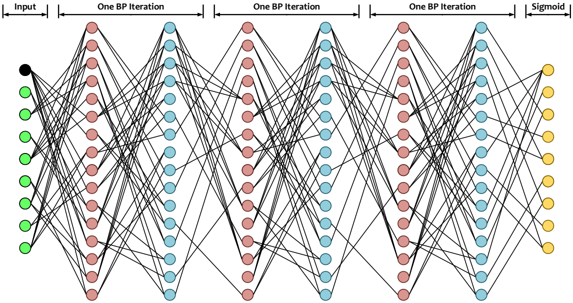
$$L_{t,e=(v,c)} = L_v + \sum_{e'=(c',v),c'\neq c} L_{t-1,e'},$$

$$L_{t,e=(c,v)} = \left(\prod_{e'=(v',c),v'\neq v} \mathrm{sign}\left(L_{t,e'}\right) \right) \times 2\tanh^{-1}\left(\prod_{e'=(v',c),v'\neq v} \tanh\left(\frac{\left|L_{t,e'}\right|}{2} \right) \right).$$

---

[4] [Cammerer, Ebada, Elkelesh, *et al.*, *ISIT* 2018]

Optimization on Tanner Graph

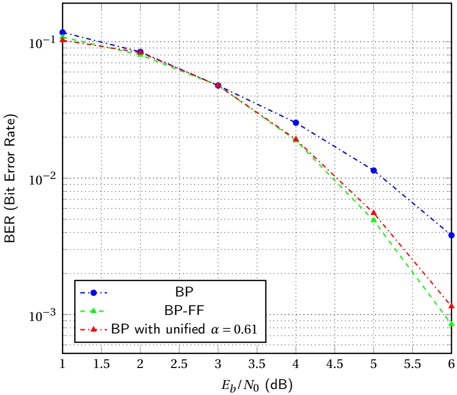


▷ Assign weights $\alpha_{i,e,e'}$ to check-to-variable edges:

$$x_{i,e=(c,v)} = \alpha_{i,e,e'} \times \prod_{e'} \mathsf{sign}(x_{i-1,e'}) \times \min_{e'}(|x_{i-1,e'}|).$$

An Example on BCH $(63, 36)$

- ▷ Using one unified weight achieves comparative performance.
- ▷ Assigning weights to every edge is redundant.



[2] [Nachmani, Marciano, Lugosch, *et al.*, *JSTSP* 2018]

Weights Reduction

- Assigning weights to every edge is redundant.
- Restrict training weights to one α parameter:

$$x_{i,e=(c,v)} = \alpha \times \prod_{e'} \mathsf{sign}(x_{i-1,e'}) \times \min_{e'}(|x_{i-1,e'}|).$$

Training Methods

- The output LLRs are squashed to $(0,1)$ probability by sigmoid:

$$o_i = \sigma(L_i) = \frac{1}{1 + e^{-L_i}}.$$

- **Binary cross entropy (BCE)** is adopted as loss function:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{o}) = -\frac{1}{N} \sum x_i \log(o_i) + (1 - x_i) \log(1 - o_i).$$

- **Optimization target**: Search optimal parameters or their combination resulting in minimum BCE loss.

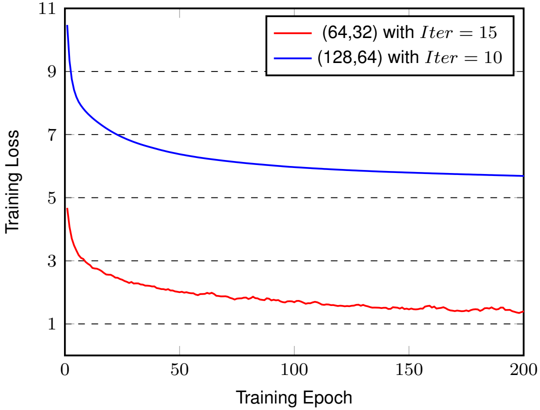
$$\alpha^* = \arg\min_{\alpha} \mathcal{L}(\boldsymbol{x}, \boldsymbol{o}).$$

Outline

Experiment on Tanner Graph

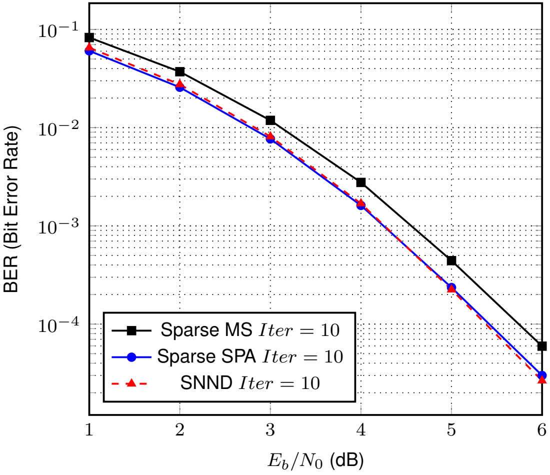
Parameters	Value
Code Length	64, 128, 256
Channel	AWGN with BPSK
SNR Range	1, 2, 3, 4, 5, 6
Optimizer	Mini-batch SGD with Adam
Learning Rate	Lr=0.001
Weights Initialization	$\mathcal{N} \sim (\mu = 1, \sigma = 0.1)$
Training Samples per SNR	20
Training mini-batch Size	120
Training Codewords	All zero codewords with noise
Validation Set Size	50000 per SNR
Validation Codewords	Random codewords with noise

Training Results

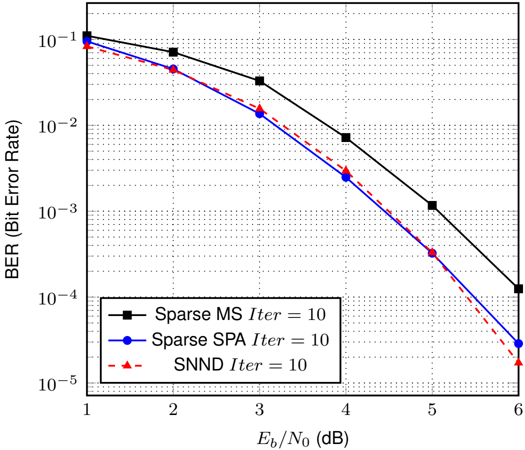


- Converge at around 150 epochs.

Performance Comparison with $N = 64$



Performance Comparison with $N = 128$



Optimizing with Just One Weight

- Constraint multiple weights to one unified α.
- **Initialization**: Initializing α to one provides a good starting point (equivalent to Min-sum):

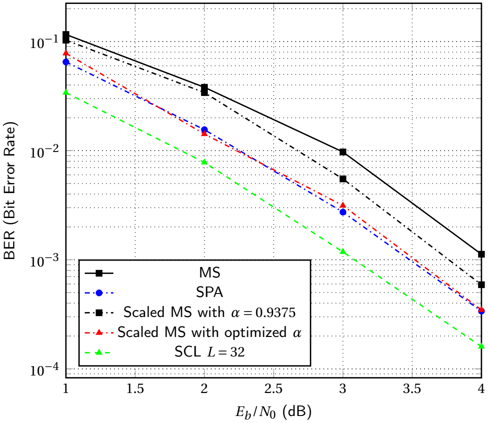
$$x_{i,e=(c,v)} = \prod_{e'} \mathsf{sign}(x_{i-1,e'}) \times \min_{e'}(|x_{i-1,e'}|).$$

- Obtain good parameter through adaptive training instead of greedy searching as in [5].
- Train by unfolding to 10 iterations and test with 50 iterations.

[5] [Yuan and Parhi, *TSP* 2014]

Performance Comparison with $N = 128$

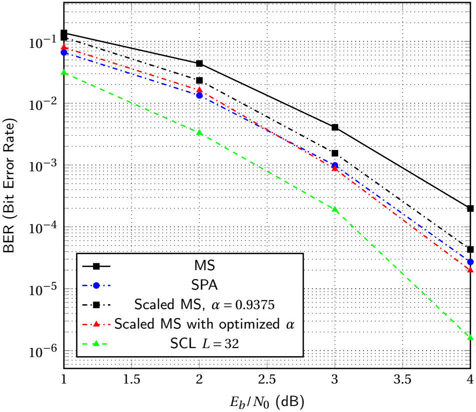
- Trained α is close to 0.85.
- 0.2 dB gain over empirical scaling factor $\alpha = 0.9375$ suggested in [5].



[5] [Yuan and Parhi, *TSP* 2014]

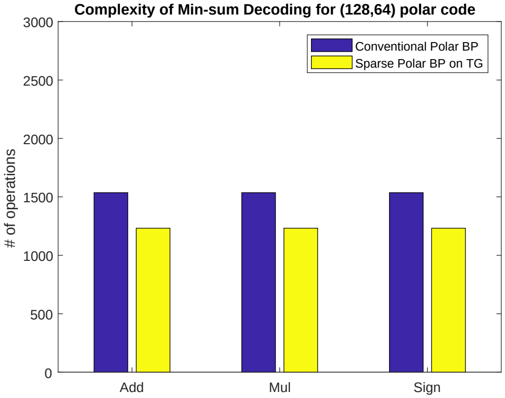
Performance Comparison with $N = 256$

- 0.1 dB gain over empirical scaling factor $\alpha = 0.9375$ suggested in [5].

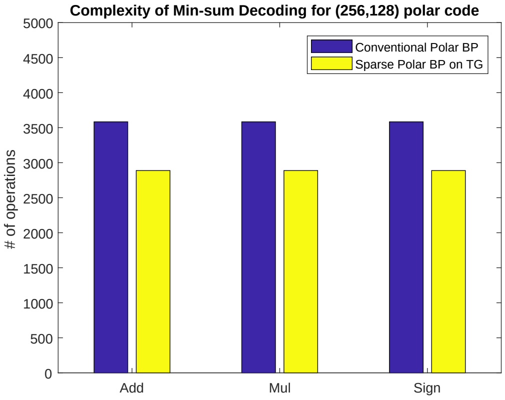


[5] [Yuan and Parhi, *TSP* 2014]

Complexity Comparison



Complexity Comparison



▷ About 25% complexity reduction compared with original polar BP.

Comparison of Decoding Latency

- **Latency on G-matrix factor graph**: $2I \log_2 N$.
- **Latency on sparse Tanner graph**: $2I$ - code length independent.
- **Latency reduction**: $\log_2 N$.

Outline

Conclusion

- **Polar neural network decoder based on Tanner graph.**
  - Reduced decoding complexity
  - Higher parallelism
- **Reduction of training weights.**
  - Restricting training weights to only one
- **Optimizing polar codes on two types of graphs.**
  - Tanner graph
  - Original factor graph

Reference

1. T. Gruber, S. Cammerer, J. Hoydis, *et al.*, "On deep learning-based channel decoding," in *Annual Conference on Information Sciences and Systems (CISS)*, IEEE, 2017, pp. 1–6
2. E. Nachmani, E. Marciano, L. Lugosch, *et al.*, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018
3. W. Xu, Z. Wu, Y.-L. Ueng, *et al.*, "Improved polar decoder based on deep learning," in *IEEE International Workshop on Signal Processing Systems (SiPS)*, 2017, pp. 1–6
4. S. Cammerer, M. Ebada, A. Elkelesh, *et al.*, "Sparse graphs for belief propagation decoding of polar codes," in *IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 1465–1469
5. B. Yuan and K. K. Parhi, "Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6496–6506, 2014

Thanks for Your Attention!

Q & A