Danmarks Tekniske Universitet DTU

Asymmetric Slepian-Wolf coding using LDPC codes

Course report by
Ronny Mueller

October 18, 2022

Contents

1 Introduction

The ever-increasing flow of information and data in our modern world sets high demands on the capabilities of current and future methods of communication. One essential requirement is the ability to send information reliable between parties over a possibly very noisy channel. This can be achieved to some degree by encoding the sent message before transmission, i.e. by adding specific redundant bits to the message. This encoded message can then be decoded at its destination. If the decoding is successful, the original information can be reconstructed, even if the transmitted message is heavily disturbed by noise. While the best possible performance of such error correcting codes in the asymptotic regime has long been know due to the Noisy-channel coding theorem, the creation, implementation, and evaluation of practically usable codes is still a field of ongoing research.

A promising family of codes is that of the low-density parity-check code (LDPC). They were discovered by Gallager in his thesis in the early 60's [1]. As their implementation was impractical at this time, they have been overlooked for a long time. In 1982, Tanner introduced the so called Tanner graph [2], an intuitive visual representation of LDPC codes. It was only in the mid 90's the good performance of LDPC codes was noticed by different groups, including MacKay and Neal. The construction, encoding and decoding of LDPC codes underwent many improvements and have seen a variety of modifications since their original design by Gallager. LDPC codes have seen a growing attention due to their good performance, resulting in widespread usage in the recent years, including as numerous standards of communication, e.g. 4G and 5G.

Distributed source coding (Slepian-Wolf coding) describes settings in which we have multiple correlated data sources that are encoded separately but decoded jointly at a single decoder. In a world in which possible data sources (sensors) become increasingly cheaper and smaller, coding schemes for distributed source coding find many applications, e.g. handheld video communications, wireless sensor networks, and multiview video [3, 4].

It is therefore the goal of this project to analyse and compare the performance of different LDPC codes in the setting of Slepian-Wolf coding.

List of Figures

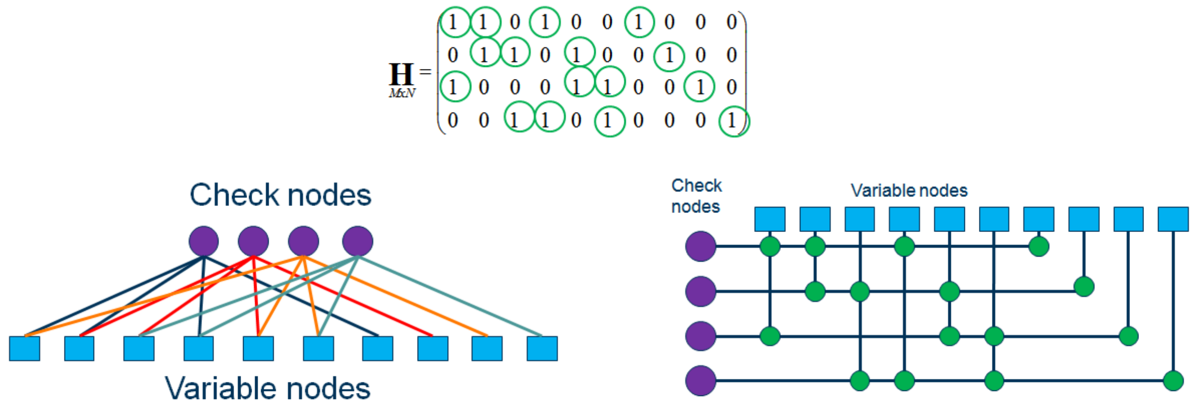
$\underset{M\times N}{\mathbf{H}} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$
Check nodes
Variable nodes
Check nodes
Variable nodes

Figure 1: Representation of an LDPC code as a Tanner graph. The upper panel shows the parity check matrix **H** of the LDPC code. On the left, one can see the corresponding Tanner graph, where each check node is connected to all the variable nodes in its corresponding row of **H**. [1]

2 Theory

In this section, we describe the two major building blocks of this work: LDPC codes and their decoding, as well as Slepian-Wolf coding.

2.1 LDPC codes

Low-density parity-check codes are linear block codes that have a very low density, as the name suggests. They are often represented using a Tanner graph, where the message bits are described as variable nodes, while the check equations of the parity check matrix are called the check nodes. Each check node is connected to all variable nodes that take part in the corresponding parity check. See Fig. 1 for a visualization. The degree of a given node describes the number of edges connected to that node. A code can then be further described by its degree distributions, the variable degree distribution and the check node degree distribution. If the degree of all variable nodes and the degree of all check nodes are constant, the code is called regular. Otherwise, it is referred to as irregular.

2.1.1 Belief propagation and the sum-product algorithm

As LDPC codes usually have a very large block size n, exact maximum likelihood decoding is infeasible. Instead, a belief propagation algorithm is used on the underlying Tanner graph of the parity check matrix, where messages are being sent between the variable and check nodes. This results in a sufficiently accurate approximation of the maximum likelihood codeword while using only a low amount of computing power. The goal of the

[1] By Kirlf - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=85648008

decoder is to create a good estimate of the posterior distribution $p(\mathbf{x}|\mathbf{y},\mathbf{H})$, where $\mathbf{x}$ is the original message encoded by the sender. The algorithm can be described using the following steps, as in [5]:

- *Initialization.* Let $p_n^0 = \mathrm{P}(c_n = 0|y_n)$ denote the prior likelihood that bit n is zero, and $p_n^1 = \mathrm{P}(c_n = 1|y_n)$ denote the prior likelihood that it is 1. The probabilities can be calculated using the characteristics of the channel.

- *Horizontal step.* The probability r_{mn}^0 that the check m is satisfied if the value of bit n is 0 dependent on all other variable nodes connected to that check is given by

$$r_{mn}^0 = \sum_{c_{n'}:n' \in \mathcal{N}(m)\backslash n} \mathrm{P}(z_m|c_n = 0, \{c_{n'} \in \mathcal{N}(m)\backslash n\})\Pi_{c_{n'} \in \mathcal{N}(m)\backslash n} q_{mn'}^{c_{n'}},$$

  where $\mathcal{N}(m)\backslash n$ denotes all the variable nodes taking part in the current check m, with exception of the current variable node n. z_m is defined by $\mathbf{z} = \mathbf{Hc}$. Analogously, the probability r_{mn}^1 is given by

$$r_{mn}^1 = \sum_{c_{n'}:n' \in \mathcal{N}(m)\backslash n} \mathrm{P}(z_m|c_n = 1, \{c_{n'} \in \mathcal{N}(m)\backslash n\})\Pi_{c_{n'} \in \mathcal{N}(m)\backslash n} q_{mn'}^{c_{n'}}.$$

- *Vertical step.* We can now update the probability that bit n has the value 0 or 1 dependent on the messages received by the check nodes as

$$q_{mn}^0 = \alpha_{mn} p_n^0 \Pi_{m' \in \mathcal{M}(n)\backslash m} r_{m'n}^0,$$

  and

$$q_{mn}^1 = \alpha_{mn} p_n^1 \Pi_{m' \in \mathcal{M}(n)\backslash m} r_{m'n}^1,$$

  where the α_{mn} are normalization constants, and $\mathcal{M}(n)\backslash m$ is the set of all check nodes that are connected to the current variable node n except for the current check m.

- *Marginalization.* The posterior distribution can be calculated anytime using that

$$\begin{aligned} q_n^0 &= \alpha_n p_n^0 \Pi_{m \in \mathcal{M}(n)} r_{mn}^0 \\ q_n^1 &= \alpha_n p_n^1 \Pi_{m \in \mathcal{M}(n)} r_{mn}^1 \end{aligned},$$

  where the α_ns are again normalization constants.

After initialization, the horizontal and vertical steps are repeated until either a set maximum number of iterations is reached or an early stopping criterion is met, e.g. when $0 = \mathbf{Hc}'$, with $\mathbf{c}'$ being the hard decision codeword that results from the posterior.

As practical implementations of LDPC codes often require very long block sizes to achieve the wanted performance, the implementation of the decoder needs to be very efficient. The most important step is to transform the equations into the logarithmic domain. We can further express our belief about the bit being 1 or 0 using the corresponding ratio instead of storing two separate values. For the initial likelihood, this corresponds to the log-likelihood-ratio LLR,

$$\mathrm{LLR}(c_i) = \log(\frac{\mathrm{P}(c_i = 0|y_i)}{\mathrm{P}(c_i = 1|y_i)}).$$

For QPSK and the AWGN channel, the LLR can be conveniently calculated to be

$$\ln\frac{p_0}{p_1} = \ln\frac{\alpha_{mn}\exp(-1/2\sigma^2(y_n+1)^2)}{\alpha_{mn}\exp(-1/2\sigma^2(y_n-1)^2)} = \frac{-1}{2\sigma^2}((y_n+1)^2-(y_n-1)^2) = -\frac{2y_n}{\sigma^2}.$$

The update equation for the r-messages in the log-domain is given by

$$Lr_{mn} = 2\mathrm{atanh}(\Pi_{n'\in\mathcal{N}(m)\backslash n}\tanh(0.5Lq_{mn'})), \tag{1}$$

and the update for the q-messages by

$$Lq_{mn} = \mathrm{LLR}(n) + \sum_{m'\in\mathcal{M}(n)\backslash m} Lr_{m'n}.$$

The decoder has been implemented using Python, where Numba has been used to accelerate the critical steps, i.e. horizontal step, vertical step, and marginalization. The parity check matrix is stored in the compressed sparse row format (CSR). An index-map from compressed sparse column format to CSR is created to allow for fast column indexing during the vertical step and marginalization.

2.2 Slepian-Wolf coding

Slepian-Wolf coding refers to the joint decoding of two separate, lossless compressed correlated sources. Let's assume that we have two correlated data streams, X and Y, being observed at encoders 1 and 2, respectively. They both encode their incoming data with the rate R_1 and R_2 choosing a codeword that can be indexed by a number in $[1, 2^{nR_1}]$ and $[1, 2^{nR_2}]$, respectively, where n is the number of bits compressed into one codeword. The encoder outputs are then used as a joint input to a single decoder. The decoder then outputs two decoded sequences, X' and Y'. Achievable systems are then those for which the probability that the original messages X and Y differ from the decoded ones X' and Y' can be made arbitrarily small by increasing the size of the input n. The achievable systems are then characterized by three inequalities according to the Slepian-Wolf Theorem:

$$\begin{aligned} R_1 &\geq H(X|Y) \\ R_2 &\geq H(Y|X) \\ R_1 + R_2 &\geq H(X,Y). \end{aligned}$$

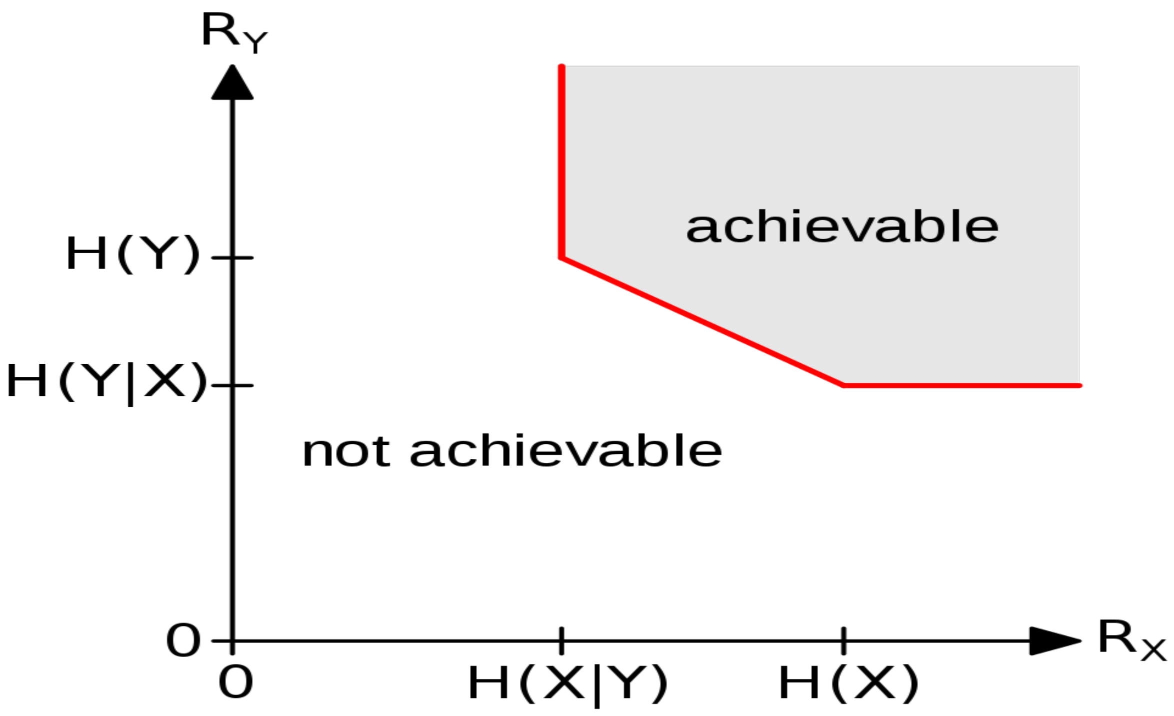

Figure 2: Area of achievable systems according to the Slepian-Wolf theorem.[2]

This poses a clear improvement compared to the case of separate encoders, where $R_1 + R_2 \geq H(X) + H(Y)$.
In this work we will consider a special case, the asymmetric distributed source coding scenario with side information at the receiver[6]. It is assumed that Y is sent at a Rate $R_2 = H(Y)$ and is recovered perfectly at the decoder. This allows in theory to send X using the rate given by the Slepian-Wolf bound of $R_1 = H(X|Y)$.

3 Methods

In this work, we will simulate the case of an asymmetric Slepian-Wolf problem, which can be expressed in the setting of channel coding. The explicit setting will be as follows, similar to the description in [7]. We take three major assumptions to paint the scene, following [8]:

1. The input data are $\mathbf{X} = [\mathbf{X_1}, ...\mathbf{X_n}]$ and $\mathbf{Y} = [\mathbf{Y_1}, ..., \mathbf{Y_n}]$, where X_i and Y_i are i.i.d. uniform binary random variables.

2. The two streams are correlated with $\mathrm{P}(X_i = Y_i) = 1 - p$ and $\mathrm{P}(X_i \neq Y_i) = p$, with $p < 0.5$.

[2]By Fvultier - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=75382948

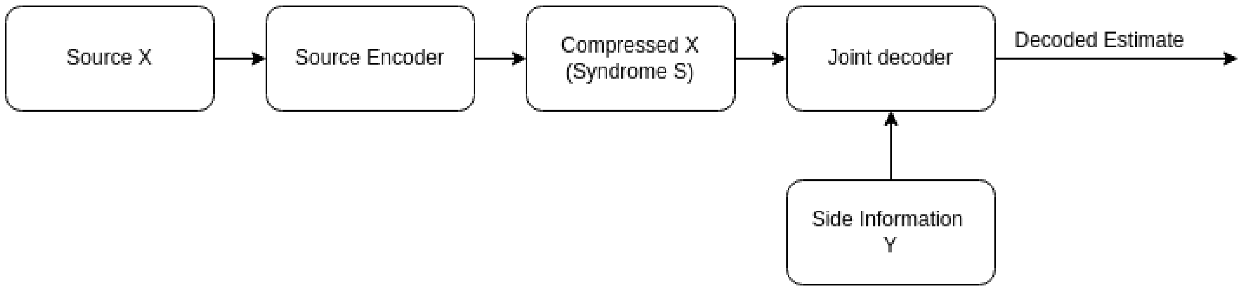

Figure 3: Simulation setting seen as a source coding setup.

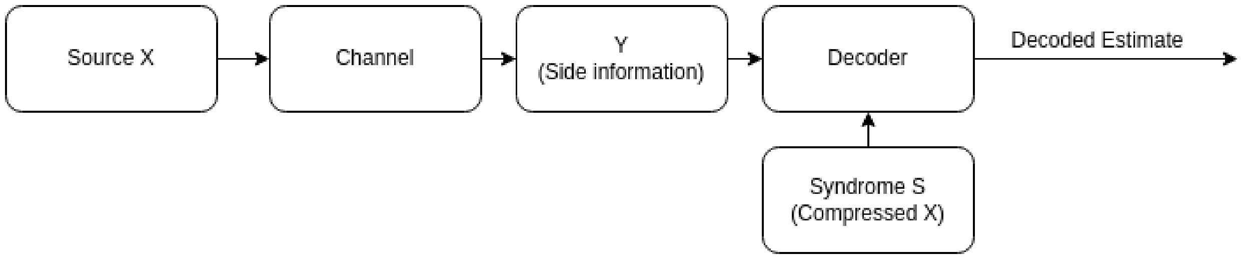

Figure 4: Simulation setting seen in a channel coding setting.

3. $\mathbf{Y}$ is available lossless at the decoder, having being compressed using the rate $R_2 = H(Y)$.

The incoming data $\mathbf{X}$ is encoded at the Source encoder and compressed at a rate $R_1 \leq H(\mathbf{X}|\mathbf{Y})$. The compression is done by using the parity check matrix $\mathbf{H}$ of an LDPC code with rate R_1, such that the compressed codeword is the syndrome $\mathbf{S}$. The compression rate is given by

$$R_{\text{compression}} = R_1 = \frac{n-k}{n} = \frac{m}{n} = 1 - R_{\text{LDPC}}. \tag{2}$$

The joint decoder then uses the compressed input $\mathbf{S}$ as well as the side information $\mathbf{Y}$ to estimate the original data $\mathbf{X}$ using a sum-product algorithm. The syndrome information is used by adding a sign factor $(-1)^{S_m}$ into equation (2.1.1). The source coding setting is visualized in Fig. 3.

The same situation can also be viewed in a channel coding setting. $\mathbf{X}$ will then be the input to a binary symmetric channel and $\mathbf{Y}$ is the corresponding channel output. The compressed version of the input, $\mathbf{S}$, then acts as a syndrome. Note that in this simple example, the conditional entropy $H(X|Y)$ that bounds the compression rate, as proven by the Slepian-Wolf theorem, is simply given by the crossover probability p of the equivalent symmetric binary channel, $H(X|Y) = H_b(p)$ [9], where $H_p(\cdot)$ is the binary entropy.

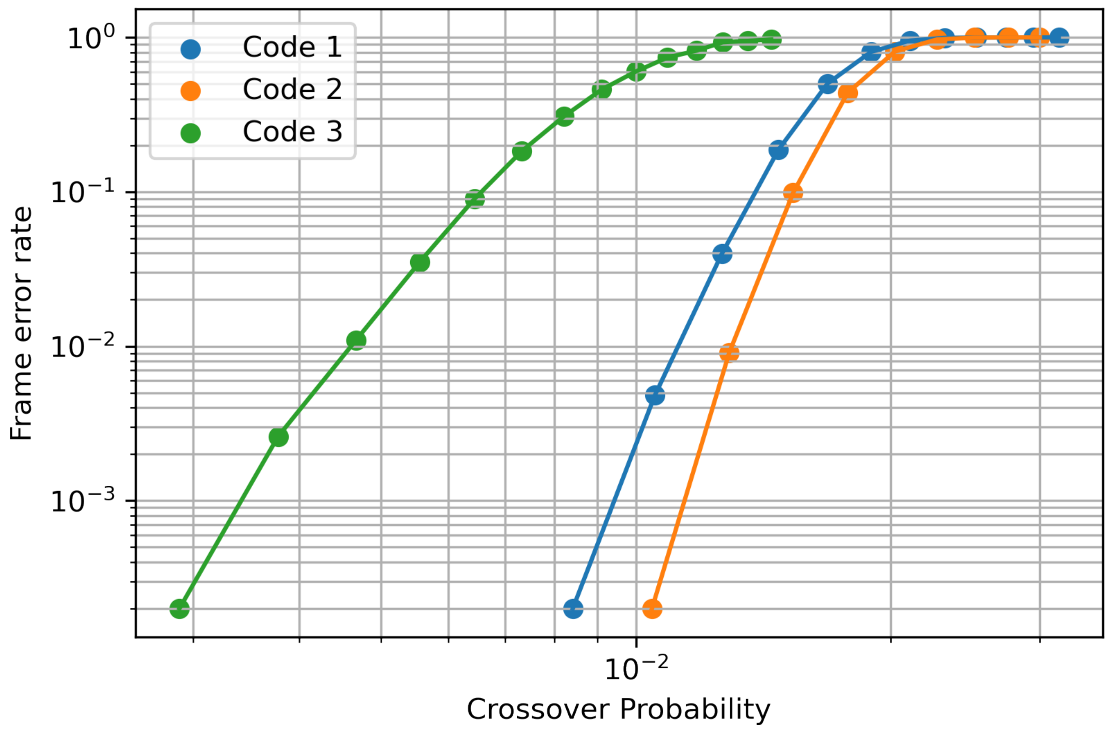

Figure 5: Frame error rate for the three simulated LDPC codes.

4 Results

We simulate 3 different LDPC codes to test their performance in the setting of decoding with side information at the receiver. All codes are taken from the freely available MacKay online archive[10]. The characteristics of the codes can be seen in Table 4. Each point has been simulated using between 10000 and 100000 random initial messages, where the number of used samples decreases with the crossover probability. The decoding is done as described in the Methods section using a sum-product algorithm with a maximum iteration count of 50. The simulation software is implemented in C++, the source code can be seen in the respective repository.

Code index	1	2	3
Blocksize n	4096	4096	1908
Syndrome length m	737	738	212
Column weight	3	4	4
Target rate	0.82	0.82	0.89

5 Conclusion

In this work, we analyzed the performance of short LDPC codes in the setting of asymmetric Slepian-Wolf coding. We showed how one can view this scenario either from a source-coding point of view or as a channel-coding problem. As already observed by MacKay, LDPC codes with column weight 4 outperform their respective version with column weight 3. It remains to remark that while the chosen setup seems very specific and oversimplified, it has direct application in the Information reconciliation phase of a quantum key distribution system [7].

References

[1] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[2] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[3] A. J. Aljohani, S. X. Ng, and L. Hanzo, "Distributed source coding and its applications in relaying-based transmission," *IEEE Access*, vol. 4, pp. 1940–1970, 2016.

[4] V. Stankovic, L. Stankovic, and S. Cheng, "Distributed source coding: Theory and applications," *European Signal Processing Conference*, 01 2010.

[5] D. J. C. MacKay, *Information Theory, Inference amp; Learning Algorithms*. Cambridge University Press, 2002.

[6] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proceedings DCC 2002. Data Compression Conference*, pp. 252–261, 2002.

[7] K. Kasai, R. Matsumoto, and K. Sakaniwa, "Information reconciliation for qkd with rate-compatible non-binary ldpc codes," in *2010 International Symposium On Information Theory Its Applications*, pp. 922–927, 2010.

[8] A. Liveris, Z. Xiong, and C. Georghiades, "Compression of binary sources with side information at the decoder using ldpc codes," *IEEE Communications Letters*, vol. 6, no. 10, pp. 440–442, 2002.

[9] J. Chen, D.-K. He, and A. Jagmohan, "The equivalence between slepian-wolf coding and channel coding under density evolution," *IEEE Transactions on Communications*, vol. 57, no. 9, pp. 2534–2540, 2009.

[10] "MacKay code archive." `https://www.inference.org.uk/mackay/codes/data.html`. Accessed: 2022-14-10.