Binary quantization using Belief Propagation with decimation over factor graphs of LDGM codes*

Tomáš Filler, Jessica Fridrich

Dept. of Electrical and Computer Engineering
SUNY Binghamton, Binghamton, NY 13902-6000, USA
{tomas.filler,fridrich}@binghamton.edu

Abstract

We propose a new algorithm for binary quantization based on the Belief Propagation algorithm with decimation over factor graphs of Low Density Generator Matrix (LDGM) codes. This algorithm, which we call Bias Propagation (BiP), can be considered as a special case of the Survey Propagation algorithm proposed for binary quantization by Wainwright et al. [8]. It achieves the same near-optimal rate-distortion performance with a substantially simpler framework and 10–100 times faster implementation. We thus challenge the widespread belief that binary quantization based on sparse linear codes cannot be solved by simple Belief Propagation algorithms. Finally, we give examples of suitably irregular LDGM codes that work with the BiP algorithm and show their performance.

1 Introduction

Binary quantization is an important problem for lossy source coding and other fields, such as information hiding [1]. The recent work of Wainwright et al. [8] shows that Low Density Generator Matrix (LDGM) codes combined with Survey Propagation (SP) message-passing algorithms can be used to achieve near-optimal binary quantization in practice. Theoretical properties of regular LDGM codes for binary quantization were studied by Martinian et al. [3]. These results motivated us to study practical algorithms for binary quantization using LDGM codes.

In this paper, we challenge the claim that lossy source coding based on pure Belief Propagation (BP) as opposed to SP cannot give satisfactory results. We propose an approach based on pure Belief Propagation for quantizing random Bernoulli source with $p = \frac{1}{2}$. This algorithm, which we call Bias Propagation (BiP), achieves the same near-optimal rate-distortion performance in comparison with [8]. This work has been motivated by applications of binary quantization in steganography (information hiding), where the problem appears in a slightly more general setting called *weighted binary quantization.*

In particular, we are quantizing a random n-bit source sequence $\mathbf{s}$ to the nearest codeword $\mathbf{c_s}$ from an LDGM code $\mathcal{C}$ with rate $R = \frac{m}{n}$. In the weighted binary quantization problem, we replace the Hamming distance with the weighted distortion measure

*This research was supported by Air Force Research Laboratory, Air Force Material Command, USAF, under the research grant FA8750-04-1-0112.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

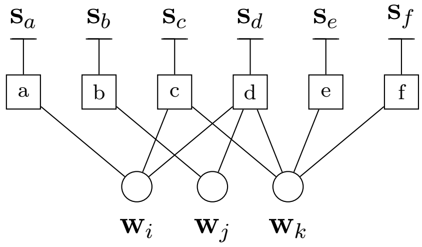


Figure 1: Factor graph representation of a linear code with generator matrix $\mathbf{G}$.

$d_\varrho(\mathbf{s}, \mathbf{c_s}) = \frac{1}{n}\sum_{i=1}^n \varrho_i |\mathbf{s}_i - (\mathbf{c_s})_i|$, where $\varrho_i \in [0,1]$. We call the vector $\varrho = (\varrho_1, \ldots, \varrho_n)$ the *profile* of the weighted binary quantization problem. Our goal is to minimize the average distortion $D_\varrho = \mathbb{E}[d_\varrho(\mathbf{s}, \mathbf{c_s})]$ for a given profile ϱ, where $\mathbb{E}[\cdot]$ is the expectation taken over all possible source sequences $\mathbf{s}$. In case of *uniform profile* $\varrho = (1, \ldots, 1)$, we denote the average distortion as D. The rate-distortion function is in the form $R(D) = 1 - H(D)$ for $D \in [0, 0.5]$ and 0 otherwise, where H is the binary entropy function. Rate-distortion functions for other profiles can be found in [1].

The rest of this paper is structured as follows. In Section 2, we describe LDGM codes and introduce notation. In Section 3, we present a complete derivation of the BiP algorithm using Belief Propagation over factor graphs of LDGM codes. In Section 4, we briefly discuss the reason why an approach based on the BP algorithm should give us satisfactory results. We make a connection to the recent work of Murayama [5]. The paper is concluded in Section 5, where we present some experimental results.

2 LDGM code representation

Codes based on sparse generator matrices are duals of LDPC codes. For a given linear code $\mathcal{C}$ with generator matrix $\mathbf{G} \in \{0,1\}^{n \times m}$, we define the factor graph of this code as a graph $\mathcal{G} = (V, C, E)$ with n *check nodes* $C = \{1, \ldots, n\}$, m *information bits* $V = \{1, \ldots, m\}$, and n *source bits*. An example of a factor graph can be seen in Figure 1. We will use variables $a, b, c \in C$ to denote the check nodes and variables $i, j, k \in V$ to denote the information bits. Each check node $a \in C$ has its associated source bit $\mathbf{s}_a$. Check node a is connected with information bit i , $(a, i) \in E$, if $\mathbf{G}_{a,i} = 1$. Finally, we define the sets $C(i) = \{a \in C \mid (a, i) \in E\}$, $V(a) = \{i \in V \mid (a, i) \in E\}$, and $\overline{V}(a) = V(a) \cup \{\mathbf{s}_a\}$.

The factor graph of an LDGM code is obtained randomly using degree distributions from the edge perspective (ρ, λ), $\rho(x) = \sum_{i=1}^{d_R} \rho_i x^{i-1}$ and $\lambda(x) = \sum_{i=1}^{d_L} \lambda_i x^{i-1}$, where ρ_i and λ_i denote the portion of all edges connected to check nodes and information bits with degree i, respectively. The degree of the check node a is defined as the number of connected information bits (we do not count the associated source bit).

3 Bias Propagation algorithm

Let $\mathcal{C}$ be an LDGM code with generator matrix $\mathbf{G} \in \{0,1\}^{n \times m}$, $\mathbf{s} \in \{0,1\}^n$ a fixed source sequence, and ϱ a given profile. For a given constant vector $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_n)$, we define the following conditional probability distribution over LDGM codewords

$$P(\mathbf{w}|\mathbf{s}; \boldsymbol{\gamma}) = \frac{1}{Z} e^{-2\langle \boldsymbol{\gamma} | \mathbf{Gw} - \mathbf{s} \rangle}, \tag{1}$$

Bias Propagation Algorithm (BiP)

(a) pseudo-code

```
procedure w = BiP(G, s)
  G.B_saa = calc_src_msg(s, gamma) /* (BiP-1) */
  G.S_ai = calc_ai(1, G.B_saa)     /* (BiP-4) */
  while not all_bits_fixed(w)
    bias = BiP_iter(G, s)
    bias = sort(bias)
    if max(|bias|)>t
      num = min(num_max, num_of_bits(|bias|>t))
    else
      num = num_min
    [G,s,w] = dec_most_biased_bits(G,s,w,num)
  end
end
```

```
procedure bias = BiP_iter(G, s)
  G.B_saa = calc_src_msg(s, gamma)           /* (BiP-1) */
  while iter<max_iter
    G.B_ia_old = G.B_ia
    G.B_ia = calc_ia(G.S_ai)                 /* (BiP-2) */
    if iter>start_damp then
      G.B_ia = damping(G.B_ia, G.B_ia_old)   /* (BiP-3) */
    end
    G.S_ai = calc_ai(G.B_ia, G.B_saa))       /* (BiP-4) */
    iter = iter+1
  end
  bias = calc_bias(G.S_ai)                   /* (BiP-5) */
end
```

(b) message-passing update rules

Source message initialization:

$$B^{(\ell)}_{\mathbf{s}_a\to a} = (-1)^{\mathbf{s}_a}\tanh(\gamma_a) \qquad \text{(BiP-1)}$$

Bias update rule:

$$\tilde{B}_{i\to a} = \frac{\prod_{b\in C(i)\backslash\{a\}}\left(1+S^{(\ell-1)}_{b\to i}\right)-\prod_{b\in C(i)\backslash\{a\}}\left(1-S^{(\ell-1)}_{b\to i}\right)}{\prod_{b\in C(i)\backslash\{a\}}\left(1+S^{(\ell-1)}_{b\to i}\right)+\prod_{b\in C(i)\backslash\{a\}}\left(1-S^{(\ell-1)}_{b\to i}\right)}. \qquad \text{(BiP-2)}$$

Equation for damping update in ℓ-th iteration:

$$B^{(\ell)}_{i\to a} = \frac{\sqrt{(1+\tilde{B}_{i\to a})(1+B^{(\ell-1)}_{i\to a})}-\sqrt{(1-\tilde{B}_{i\to a})(1-B^{(\ell-1)}_{i\to a})}}{\sqrt{(1+\tilde{B}_{i\to a})(1+B^{(\ell-1)}_{i\to a})}+\sqrt{(1-\tilde{B}_{i\to a})(1-B^{(\ell-1)}_{i\to a})}}. \qquad \text{(BiP-3)}$$

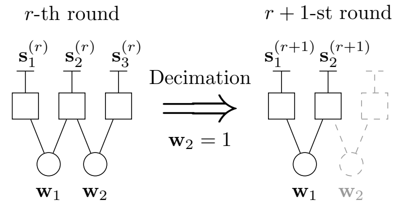
Equation for calculating the final bias B_i in $\hat{\ell}$-th iteration:

$$B_i = \frac{\prod_{b\in C(i)}\left(1+S^{(\hat{\ell})}_{b\to i}\right)-\prod_{b\in C(i)}\left(1-S^{(\hat{\ell})}_{b\to i}\right)}{\prod_{b\in C(i)}\left(1+S^{(\ell)}_{b\to i}\right)+\prod_{b\in C(i)}\left(1-S^{(\ell)}_{b\to i}\right)}. \qquad \text{(BiP-5)}$$

Satisfaction update rule:

$$S^{(\ell)}_{a\to i} = \prod_{j\in\overline{V}(a)\backslash\{i\}} B^{(\ell)}_{j\to a} \qquad \text{(BiP-4)}$$

Graph decimation:



$$\mathbf{s}_1^{(r+1)} = \mathbf{s}_1^{(r)} \qquad \mathbf{s}_2^{(r+1)} = XOR\left(\mathbf{s}_2^{(r)}, \mathbf{w}_2\right)$$

Constant parameters:
`num_max`... max. # of info bits to decimate
`num_min`... min. # of info bits to decimate
`t`... decimation threshold
`gamma` ... check node satisfaction strength
`max_iter` ... max. # of iterations in round
`start_damp` ... # iterations without damping

Figure 2: Summary of the Bias Propagation algorithm.

where $\langle\boldsymbol{\gamma}|\mathbf{Gw}-\mathbf{s}\rangle$ is the dot product of vectors $\boldsymbol{\gamma}$ and $\mathbf{Gw}-\mathbf{s}$ (calculated in binary arithmetic), $\mathbf{w}\in\{0,1\}^m$, and Z is a normalization constant $Z=\sum_{\mathbf{c}\in\mathcal{C}} e^{-2\langle\boldsymbol{\gamma}|\mathbf{c}-\mathbf{s}\rangle}$. The most probable codeword $\mathbf{c_s}=\mathbf{Gw_s}$ is the optimal solution to our original problem. The vector $\boldsymbol{\gamma}$ should be determined from the profile ϱ.

The BiP is an iterative message-passing algorithm that performs bitwise MAP estimation of the optimal vector $\mathbf{w_s}$ for a given source sequence $\mathbf{s}$. This is done in rounds. In r-th round, we use the factor graph $\mathcal{G}^{(r)}$ and source sequence $\mathbf{s}^{(r)}$ to find the most probable bits of $\mathbf{w_s}$ to be fixed. The estimation of these bits is done by `max_iter` message-passing iterations over the factor graph $\mathcal{G}^{(r)}$. In ℓ-th iteration, the bias messages $B^{(\ell)}_{i\to a}$ and the constant source messages $B^{(\ell)}_{\mathbf{s}_a\to a}$ are sent from information bits and source bits to connected check nodes. Check nodes send satisfaction messages $S^{(\ell)}_{a\to i}$ to their connected information bits. Finaly, the most probable information bits are fixed and removed from the graph by the decimation process. This results in a new factor graph $\mathcal{G}^{(r+1)}$ and a source sequence $\mathbf{s}^{(r+1)}$. We describe the algorithm in a condensed form in Figure 2. Later in this section, we derive the update rules.

Given the original factor graph $\mathcal{G}^{(1)}=\mathcal{G}$ and the initial source sequence $\mathbf{s}^{(1)}=\mathbf{s}$, we start the message-passing process by setting $S^{(0)}_{a\to i}=B^{(0)}_{\mathbf{s}_a\to a}$, $\forall a\in C$. The source messages are calculated using equation (BiP-1) and stay constant within one round. The parameter γ_a expresses the strength of the check node a. After `max_iter` message-passing iterations using equations (BiP-2)–(BiP-4), we calculate the final bias B_i for each information bit i using equation (BiP-5). The bias B_i expresses the difference of marginal probabilities $P(\mathbf{w}_i = 0|\mathbf{s};\boldsymbol{\gamma}) - P(\mathbf{w}_i = 1|\mathbf{s};\boldsymbol{\gamma})$. We fix the most biased information bit to $\mathbf{w}_i = 0$ if $B_i > 0$ and $\mathbf{w}_i = 1$ otherwise. Based on the maximal $|B_i|$, we fix `num_min` or `num_max` information bits in each round. The decimation step removes all fixed information bits from the factor graph along with all adjacent edges and checks with degree zero. This results in a new factor graph $\mathcal{G}^{(2)}$. The new source sequence $\mathbf{s}^{(2)}$ is obtained from $\mathbf{s}^{(1)}$ by XOR-ing it with all bits that were connected to the fixed information bits (see Figure 2). The satisfaction messages $S^{(0)}_{a\to i}$ in the second round are initialized with the value of $S^{(\hat{\ell})}_{a\to i}$ messages from the last iteration in the previous round. The decimation step only removes some edges and vertices from the factor graph, but preserves the values of messages associated with each edge. We repeat the described procedure (one round) until we fix all information bits from the original factor graph and output the sequence of information bits as $\mathbf{w}_{\mathbf{s}}$.

3.1 Derivation of the BiP algorithm

We now carry out the derivations for an arbitrary profile ϱ assuming that we know the vector $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_n)$, $\gamma_i \geqslant 0\ \forall i = 1, \ldots, n$, and reformulate the weighted binary quantization problem as bitwise Maximum A Posteriori (MAP) estimation.

First, we factorize the probability distribution (1). For $\mathbf{c} \in \mathcal{C}$, we can find $\mathbf{w} \in \{0,1\}^m$, such that $\mathbf{c} = \mathbf{G}\mathbf{w}$. Thus, we can write

$$P(\mathbf{w}|\mathbf{s};\boldsymbol{\gamma}) = \prod_{a\in C} \psi_a(\mathbf{w}_{V(a)}, \mathbf{s}_a), \tag{2}$$

where $\psi_a\big(\mathbf{w}_{V(a)}, \mathbf{s}_a\big) = e^{\gamma_a}$ if $\mathbf{s}_a = \sum_{i\in V(a)} \mathbf{w}_i$ and $\psi_a\big(\mathbf{w}_{V(a)}, \mathbf{s}_a\big) = e^{-\gamma_a}$ otherwise. We will use the sum-product (Belief Propagation) algorithm [2] to calculate marginal probabilities for each information bit and find the assignment for each information bit in the form

$$\hat{\mathbf{w}}_i = \arg \max_{\mathbf{w}_i\in\{0,1\}} P(\mathbf{w}_i|\mathbf{s}) = \arg \max_{\mathbf{w}_i\in\{0,1\}} \sum_{\sim\mathbf{w}_i} P(\mathbf{w}|\mathbf{s}) = \arg \max_{\mathbf{w}_i\in\{0,1\}} \sum_{\sim\mathbf{w}_i} \prod_{a\in C} \psi_a\big(\mathbf{w}_{V(a)}, \mathbf{s}_a\big). \tag{3}$$

Here, the sum over all information variables without the i-th one is shortened as $\sum_{\sim\mathbf{w}_i}$.

To calculate (3) using the sum-product algorithm efficiently, we simplify the original update equations (see [2] for the original update rules). For our problem, all messages $M^{(\ell)}_{i\to a}$ and $M^{(\ell)}_{a\to i}$ passed in ℓ-th iteration in the original sum-product algorithm are two-component vectors. Using the original update equations, we define

$$R^{(\ell)}_{i\to a} = \frac{M^{(\ell)}_{i\to a}(1)}{M^{(\ell)}_{i\to a}(0)} = \frac{\prod_{b\in C(i)\backslash\{a\}} M^{(\ell-1)}_{b\to i}(1)}{\prod_{b\in C(i)\backslash\{a\}} M^{(\ell-1)}_{b\to i}(0)} = \prod_{b\in C(i)\backslash\{a\}} R^{(\ell-1)}_{b\to i} \tag{4}$$

and for messages $\mathbf{M}_{a\to i}$

$$R^{(\ell)}_{a\to i} = \frac{M^{(\ell)}_{a\to i}(1)}{M^{(\ell)}_{a\to i}(0)} = \frac{\sum_{\mathbf{w}_{V(a)\backslash\{i\}}} \Big[\psi_a(1, \mathbf{w}_{V(a)\backslash\{i\}}, \mathbf{s}_a) \prod_{j\in V(a)\backslash\{i\}} M^{(\ell)}_{j\to a}(\mathbf{w}_j)\Big]}{\sum_{\mathbf{w}_{V(a)\backslash\{i\}}} \Big[\psi_a(0, \mathbf{w}_{V(a)\backslash\{i\}}, \mathbf{s}_a) \prod_{j\in V(a)\backslash\{i\}} M^{(\ell)}_{j\to a}(\mathbf{w}_j)\Big]}. \tag{5}$$

To calculate the last ratio, we will use the definition of function ψ_a and partition the set $\mathbf{w}_{V(a)\setminus\{i\}} = W_{even} \cup W_{odd}$, where $W_{even} = \big\{\mathbf{x} \in \{0,1\}^{|V(a)|-1} \big| \text{ \# of 1 in } \mathbf{x} \text{ is even}\big\}$ and $W_{odd} = \big\{\mathbf{x} \in \{0,1\}^{|V(a)|-1} \big| \text{ \# of 1 in } \mathbf{x} \text{ is odd}\big\}$.

We now assume that $\mathbf{s}_a = 0$ and later remove this assumption. In this special case, we have $\psi_a(0, \mathbf{w}_{V(a)\setminus\{i\}}, \mathbf{s}_a) = e^{\gamma_a}$ for all vectors from W_{even} and $\psi_a(0, \mathbf{w}_{V(a)\setminus\{i\}}, \mathbf{s}_a) = e^{-\gamma_a}$ for all vectors from W_{odd}. Conversely, $\psi_a(1, \mathbf{w}_{V(a)\setminus\{i\}}, \mathbf{s}_a) = e^{-\gamma_a}$ for W_{even} and $\psi_a(1, \mathbf{w}_{V(a)\setminus\{i\}}, \mathbf{s}_a) = e^{\gamma_a}$ for W_{odd}. We can substitute and write $R^{(\ell)}_{a\to i} = \frac{e^{-\gamma_a}A + e^{\gamma_a}B}{e^{\gamma_a}A + e^{-\gamma_a}B}$, where $A = \sum_{W_{even}} \prod_{j\in V(a)\setminus\{i\}} M^{(\ell)}_{j\to a}(\mathbf{w}_j)$, and $B = \sum_{W_{odd}} \prod_{j\in V(a)\setminus\{i\}} M^{(\ell)}_{j\to a}(\mathbf{w}_j)$. We can express both sums (A and B) in terms of the ratios $R^{(\ell)}_{i\to a}$ by dividing them by the constant factor $\prod_{j\in V(a)\setminus\{i\}} M^{(\ell)}_{j\to a}(0)$

$$\begin{aligned}\hat{A} &= \frac{A}{\prod_{j\in V(a)\setminus\{i\}} M^{(\ell)}_{j\to a}(0)} = \sum_{W_{even}} \prod_{j\in V(a)\setminus\{i\}} \frac{M^{(\ell)}_{j\to a}(\mathbf{w}_j)}{M^{(\ell)}_{j\to a}(0)} = \sum_{W_{even}} \prod_{j\in V(a)\setminus\{i\}} \Big(R^{(\ell)}_{j\to a}\Big)^{\mathbf{w}_j} = \\ &= \frac{1}{2}\Big[\underbrace{\prod_{j\in V(a)\setminus\{i\}} \big(1 + R^{(\ell)}_{j\to a}\big)}_{=C} + \underbrace{\prod_{j\in V(a)\setminus\{i\}} \big(1 - R^{(\ell)}_{j\to a}\big)}_{=D}\Big] = \frac{1}{2}(C + D).\end{aligned}$$

Similarly, for B, $\hat{B} = \frac{1}{2}(C - D)$. Finally, we obtain

$$R^{(\ell)}_{a\to i} = \frac{e^{-\gamma_a}\hat{A} + e^{\gamma_a}\hat{B}}{e^{\gamma_a}\hat{A} + e^{-\gamma_a}\hat{B}} = \frac{1 - \frac{e^{\gamma_a} - e^{-\gamma_a}}{e^{\gamma_a} + e^{-\gamma_a}}\frac{D}{C}}{1 + \frac{e^{\gamma_a} - e^{-\gamma_a}}{e^{\gamma_a} + e^{-\gamma_a}}\frac{D}{C}} = \frac{1 - S^{(\ell)}_{a\to i}}{1 + S^{(\ell)}_{a\to i}}, \tag{6}$$

where we used the substitution

$$S^{(\ell)}_{a\to i} = (-1)^{\mathbf{s}_a}\Big(\frac{e^{\gamma_a} - e^{-\gamma_a}}{e^{\gamma_a} + e^{-\gamma_a}}\Big) \prod_{j\in V(a)\setminus\{i\}} \frac{1 - R^{(\ell)}_{j\to a}}{1 + R^{(\ell)}_{j\to a}}. \tag{7}$$

It is easy to see that the case where $\mathbf{s}_a = 1$ can be captured by the given substitution. Using the substitution $B^{(\ell)}_{i\to a} = (1 - R^{(\ell)}_{i\to a})/(1 + R^{(\ell)}_{i\to a})$, we can completely rewrite equations (4) and (5) solely in terms of $B^{(\ell)}_{i\to a}$ and $S^{(\ell)}_{a\to i}$ obtaining thus the final message-passing rules. From (6),

$$\begin{aligned}B^{(\ell)}_{i\to a} &= \frac{1 - R^{(\ell)}_{i\to a}}{1 + R^{(\ell)}_{i\to a}} = \frac{1 - \prod_{b\in C(i)\setminus\{a\}} R^{(\ell-1)}_{b\to i}}{1 + \prod_{b\in C(i)\setminus\{a\}} R^{(\ell-1)}_{b\to i}} = \frac{1 - \prod_{b\in C(i)\setminus\{a\}} \frac{1 - S^{(\ell-1)}_{b\to i}}{1 + S^{(\ell-1)}_{b\to i}}}{1 + \prod_{b\in C(i)\setminus\{a\}} \frac{1 - S^{(\ell-1)}_{b\to i}}{1 + S^{(\ell-1)}_{b\to i}}} = \\ &= \frac{\prod_{b\in C(i)\setminus\{a\}}\big[1 + S^{(\ell-1)}_{b\to i}\big] - \prod_{b\in C(i)\setminus\{a\}}\big[1 - S^{(\ell-1)}_{b\to i}\big]}{\prod_{b\in C(i)\setminus\{a\}}\big[1 + S^{(\ell-1)}_{b\to i}\big] + \prod_{b\in C(i)\setminus\{a\}}\big[1 - S^{(\ell-1)}_{b\to i}\big]}\end{aligned}$$

and from (7)

$$S^{(\ell)}_{a\to i} = \prod_{j\in \overline{V}(a)\setminus\{i\}} B^{(\ell)}_{j\to a},$$

where the source message $B^{(\ell)}_{\mathbf{s}_a\to a}$ is defined as $B^{(\ell)}_{\mathbf{s}_a\to a} = (-1)^{\mathbf{s}_a}\tanh(\gamma_a)$. After `max_iter` iterations, we compute the final bias B_i using the last satisfaction messages $S^{(\hat{\ell})}_{a\to i}$

$$B_i = \frac{P(0) - P(1)}{P(0) + P(1)} = \frac{1 - \frac{P(1)}{P(0)}}{1 + \frac{P(1)}{P(0)}} = \frac{1 - \prod_{b\in C(i)} R^{(\hat{\ell})}_{b\to i}}{1 + \prod_{b\in C(i)} R^{(\hat{\ell})}_{b\to i}} = \frac{\prod_{b\in C(i)}\big[1 + S^{(\hat{\ell})}_{b\to i}\big] - \prod_{b\in C(i)}\big[1 - S^{(\hat{\ell})}_{b\to i}\big]}{\prod_{b\in C(i)}\big[1 + S^{(\hat{\ell})}_{b\to i}\big] + \prod_{b\in C(i)}\big[1 - S^{(\hat{\ell})}_{b\to i}\big]}.$$

Thus, we just obtained equations (BiP-1), (BiP-2), (BiP-4), and (BiP-5) from Figure 2.

3.2 Dealing with cycles in the factor graph

The sum-product algorithm is exact (gives exact results) when the underlying graph is a tree. However, many researchers reported good results even for graphs with cycles. In principle, short cycles cause the messages to oscilate. The oscillations can be suppressed using a procedure called "damping". A similar approach was introduced in the context of statistical mechanics by Pretti [6].

Using equation (4), we can write

$$\ln R_{i\to a}^{(\ell)} = \ln \prod_{b\in C(i)\setminus\{a\}} R_{b\to i}^{(\ell-1)} = \sum_{b\in C(i)\setminus\{a\}} \ln R_{b\to i}^{(\ell-1)}. \tag{8}$$

In other words, the update rule (BiP-2) is a sum of logarithmic terms. Thus, we can use the arithmetic mean in this representation to calculate the output message in the ℓ-th iteration by averaging the input messages from iterations $\ell-1$ and $\ell-2$. This "low-pass temporal filter" will prevent large changes to output messages. Using (8), we can write the output ratio after applying the damping procedure as $\ln R_{i\to a}^{(\ell)} = \frac{1}{2}\Big(\sum_{b\in C(i)\setminus\{a\}} \ln R_{b\to i}^{(\ell-1)} + \sum_{b\in C(i)\setminus\{a\}} \ln R_{b\to i}^{(\ell-2)}\Big)$, and hence

$$R_{i\to a}^{(\ell)} = \Big(\prod_{b\in C(i)\setminus\{a\}} R_{b\to i}^{(\ell-1)} \cdot \prod_{b\in C(i)\setminus\{a\}} R_{b\to i}^{(\ell-2)}\Big)^{\frac{1}{2}}. \tag{9}$$

Using $B_{i\to a}^{(\ell)} = (1 - R_{i\to a}^{(\ell)})/(1 + R_{i\to a}^{(\ell)})$ and equations (9) and (6), we obtain equation (BiP-3) that is used for damping in the BiP algorithm. To obtain the message $B_{i\to a}^{(\ell)}$ in practice, we calculate the temporary bias message $\hat{B}_{i\to a}$ using equation (BiP-2). Finaly, $B_{i\to a}^{(\ell)}$ is obtained from messages $\hat{B}_{i\to a}$ and $B_{i\to a}^{(\ell-1)}$ using equation (BiP-3).

4 Convergence of the BiP algorithm

While deriving the BiP algorithm, we tacitly assumed that for some information bits the magnitude of the bias at the end of each round is high, $|B_i| \approx 1$. Such bits have a high tendency to be fixed to some value without generating too much distortion. To obtain a small final distortion, this condition should be fulfiled in each round for some bits. From practical experiments, we know that there exist good degree distributions satisfying this condition, while other distributions, such as regular distributions, do not build up sufficient bias magnitudes in each round and thus produce very high distortion. This observation was also made in [8].

We say that the BiP algorithm converges in r-th round if there exists at least one information bit $i \in V$ that fulfills $|B_i| > \mathtt{t}$ for some constant threshold $\mathtt{t}$. Moreover, we say that the BiP algorithm converges if it converges in all its rounds. A part of our future research is to find a condition for the class of degree distributions for which the BiP algorithm converges. This condition could be used for construction of degree distributions optimized for the BiP algorithm in a manner similar to density evolution in analysis of LDPC codes for channel coding [7].

Murayama [5] developed an approach for lossy source coding based on a modified Belief Propagation algorithm and provided results for regular LDGM codes with a fixed check degree 2. This algorithm was based on a standard approach used in channel coding:

Rate: 0.37
$\rho(x) = 0.2710x + 0.2258x^2 + 0.1890x^5 + 0.0614x^6 + 0.2528x^{13}$
$\lambda(x) = 0.9522x^9 + 0.0478x^{10}$

Rate: 0.5
$\rho(x) = 0.1787x + 0.1762x^2 + 0.1028x^5 + 0.1147x^6 + 0.0122x^{12} + 0.0479x^{13} + 0.1159x^{14} +$
$+ 0.2516x^{39}$
$\lambda(x) = 0.9988x^9 + 0.0012x^{10}$

Rate: 0.65
$\rho(x) = 0.2454x + 0.1921x^2 + 0.1357x^5 + 0.0838x^6 + 0.1116x^{12} + 0.0029x^{14} + 0.0222x^{15} +$
$+ 0.0742x^{28} + 0.1321x^{32}$
$\lambda(x) = 0.4987x^5 + 0.5013x^6$

Rate: 0.75
$\rho(x) = 0.2912x + 0.1892x^2 + 0.0408x^4 + 0.0873x^5 + 0.0074x^6 + 0.1126x^7 + 0.0926x^{15} +$
$+ 0.0187x^{20} + 0.1241x^{32} + 0.0361x^{39}$
$\lambda(x) = 0.8016x^4 + 0.1984x^5$

Figure 3: List of good degree distributions used for generating the results.

run the BP algorithm over the factor graph of an LDGM code and threshold all bits based on the final LLRs. However, this approach cannot be used for a general LDGM code. The key point here is the degree of all check nodes. This idea is strongly connected with the BiP algorithm. Due to sufficient number of check nodes with degree 2, the BiP algorithm converges in its first round. The same observation can be made in other rounds.

We can think of the BiP algorithm as a generalized approach described by Murayama. In terms of the BiP algorithm, he only uses one round and decimates all information bits at once. The BiP algorithm utilizes more rounds via decimation and thus it can use more general degree distributions. By using irregular degree distributions, we can achieve near-optimal rate-distortion performance. Good LDGM codes can be obtained from degree distributions optimized for channel coding, especially for the BSC channel. This choice, which was also suggested in [8], was motivated by the work of Martinian et al. [4] who pointed out a tight connection between capacity achieving codes for the BEC channel and their dual codes used for binary quantization in the erasure case.

5 Results

We implemented the BiP algorithm in C++, where the messages were represented using single precision numbers. The update equations were manually optimized using Intel's SSE1 extension so that the cache memory was used in an optimal way. All results presented in this work were obtained using an Intel Core2 X6800 2.93GHz CPU machine with 2GB RAM. We ran the machine on Linux in 64 bit mode. All C++ code was optimized to 64 bit mode and compiled using Intel C++ 9.0 compiler. The speed of this algorithm (throughput) was measured while both CPU cores were utilized. We ran the same algorithm on both cores, resulting in a 1.7–1.8 times higher throughput.

In Figure 3, we present degree distributions from edge perspective that were used for generating all results. Some distributions were obtained from the LdpcOpt site (`http://lthcwww.epfl.ch/research/ldpcopt/`) and optimized for the BSC channel.

To compare our results with the work of Wainwright et al. [8], we implemented their algorithm while using similar optimization approaches. In the case of an irregular code

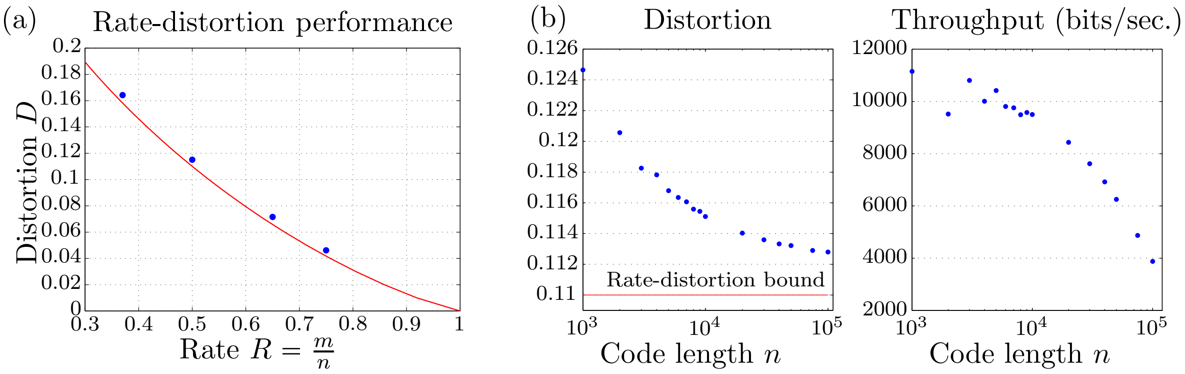


Figure 4: (a) rate-distortion performance plot for selected irregular codes of length $n = 10^4$. (b) distortion and throughput plots for various code lengths for irregular code with rate $R = 0.5$. In all graphs, each point represents an average over 100 trials.

with rate $R = 0.5$ and length $n = 5000$, we achieved throughput 1013 bits/sec. The BiP algorithm produced the same distortion with throughput 10415 bits/sec. The BiP algorithm is a special case of their algorithm for $w_{sou} = w_{info} = 0$.

Figure 4 contains simulation results for various rates and various code lengths for a uniform profile ϱ. All results were generated using the following parameter values: $\texttt{t} = 0.8$, $\texttt{max\_iter} = 25$, $\texttt{start\_damp} = 10$, $\texttt{num\_max} = 0.01 \times m$, $\texttt{num\_min} = 0.001 \times m$. We set $\boldsymbol{\gamma} = (\gamma, \ldots, \gamma)$, where $\gamma = 0.8$, $\gamma = 1.07$, $\gamma = 1.3$, $\gamma = 1.5$ for rates $R = 0.37$ to $R = 0.75$, respectively.

We now present the results for a linear profile of weights (which is important for applications in steganography [1]), and perform weighted binary quantization using the BiP algorithm. For a source sequence $\mathbf{s}$ of length n, we define the linear profile $\varrho = (\varrho_1, \ldots, \varrho_n)$ as $\varrho_i = 2(n-i)/n$. From [1] (Appendix A), the rate-distortion bound in this weighted case will be achieved if each bit is flipped with probability $p_a(1) = e^{-\zeta\varrho_a}/(1 + e^{-\zeta\varrho_a})$ for each source bit $a \in C$.

To find optimal values of γ_a, we use the following iterative approach. Start with $\gamma_a^{(0)} = \gamma$, where γ is taken from the uniform weight case. Find an estimate of $p_a(1)$ (denote it $\hat{p}_a(1)$) by quantizing k random source sequences. We calculate the estimate as an arithmetic average and finally use a moving average filter to smooth the resulting sequence. Comparing the estimate $\hat{p}_a(1)$ with $p_a(1)$, we finally set $\gamma_a^1 = \gamma_a^0 + c\big(\hat{p}_a(1) - p_a(1) - (\hat{p} - p)\big)$, where $\hat{p}$ and p is the arithmetic average of $\hat{p}_a(1)$ and $p_a(1)$, respectively, and c is a constant (in practice, we use $c = 3$).

Usually, 10 iterations were sufficient to obtain good results. In Figure 5, we present the resulting $\hat{p}_a(1)$ and γ_a for rate $R = 0.5$. A cubic polynomial was fit through the data points (it uses a centralized variable x). In Figure 6 (b), we use this polynomial and show how the BiP depends on the code length. We can see that the throughput is roughly the same as for the non-weighted BiP. In Figure 6 (a), we present the overall comparison of weighted vs. non-weighted BiP algorithm for all degree distributions. Here, we use the ordinary BiP algorithm ($\boldsymbol{\gamma} = (\gamma, \ldots, \gamma)$) and measure the distortion using the weighted norm (linear case). Although the degree distributions were taken from Figure 3 and were thus not optimized for the weighted case, the weighted BiP algorithm still achieves very good results.

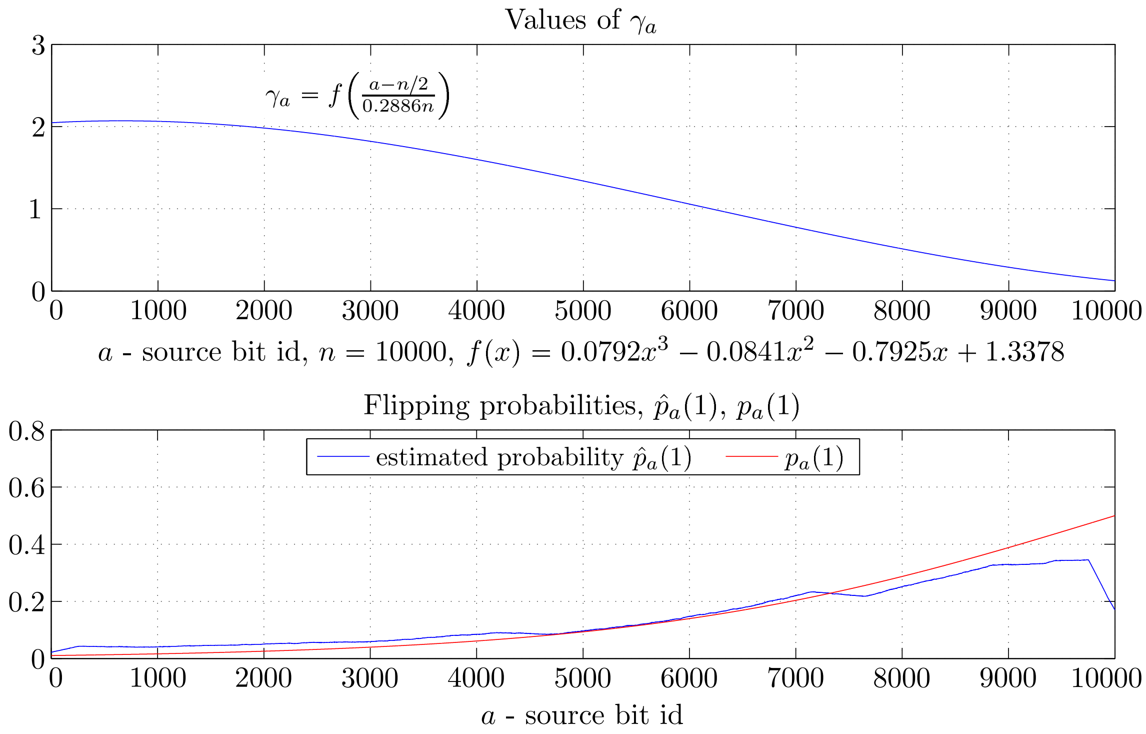


Figure 5: Flipping probabilities for linear profile ϱ, $R = 0.5$, $\zeta = 4.544$. Values of γ_a were obtained using an iterative approach for $n = 10000$ and interpolated by a cubic polynomial.

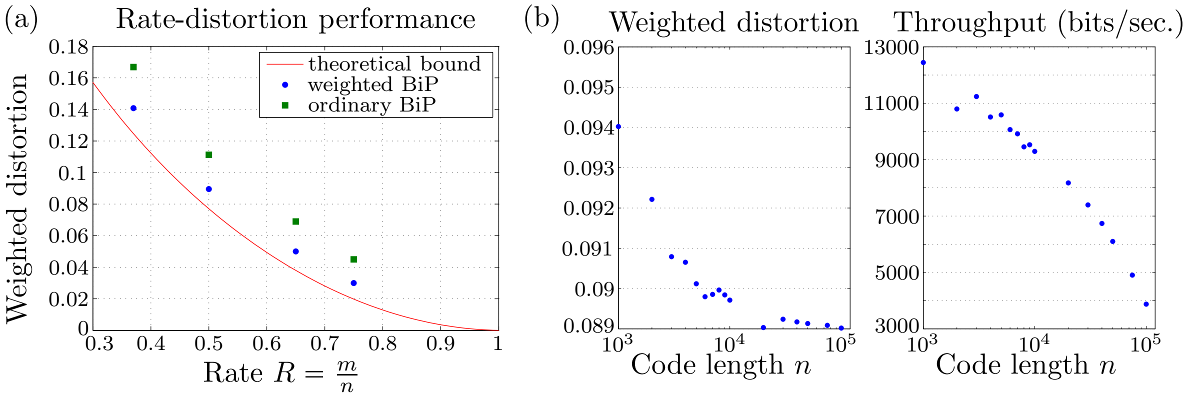


Figure 6: (a) overall comparison of weighted vs. non-weighted BiP algorithm for a linear profile ϱ, $n = 10000$. (b) results from using weighted BiP algorithm for linear profile ϱ using different code lengths, $R = 0.5$. In all graphs, each point represents an average over 100 trials.

6 Conclusions and future work

We proposed a new algorithm for binary quantization based on the Belief Propagation algorithm with decimation over factor graphs of LDGM codes. We call the algorithm Bias Propagation (BiP). Using the Bias Propagation algorithm, we significantly reduced the complexity of binary quantization using LDGM codes in comparison with [8]. We believe this reduction is new and constitutes an important contribution as it allows us to theoretically study the algorithm in the future. We postpone this problem to our future work. We showed that the algorithm based on pure Belief Propagation has the same rate-distortion performance as the algorithm based on Survey Propagation [8].

References

[1] J. Fridrich and T. Filler. Practical methods for minimizing embedding impact in steganography. In E.J. Delp and P.W. Wong, editors, *Proceedings SPIE Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX, San Jose, CA*, volume 6505, pages 02.1–02.15, January 29–February 1 2007.

[2] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[3] E. Martinian and M. J. Wainwright. Analysis of LDGM and compound codes for lossy compression and binning. In *Workshop on Information Theory and its Applications, San Diego, CA*, February 2006.

[4] E. Martinian and J. S. Yedidia. Iterative quantization using codes on graphs. In *Allerton Conference on Control, Computing, and Communication*, October 2003.

[5] T. Murayama. Thouless-Anderson-Palmer approach for lossy compression. *Physical Review E*, 69(3):035105, 2004.

[6] M. Pretti. A message-passing algorithm with damping. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11):P11008, 2005.

[7] T. Richardson and R. Urbanke. *Modern coding theory.* Cambridge University Press, 2007. Not published yet, download from http://lthcwww.epfl.ch/mct/.

[8] M. J. Wainwright and E. Maneva. Lossy source encoding via message-passing and decimation over generalized codewords of LDGM codes. In *Proceedings of the International Symposium on Information Theory, Adelaide, Australia*, September 2005.