Efficient data compression from statistical physics of codes over finite fields

A. Braunstein,[1,2,3] F. Kayhan,[4] and R. Zecchina[1,2,3]
[1]*Dipartimento di Fisica and Center for Computational Sciences, Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129 Torino, Italy*
[2]*HuGeF, Via Nizza 52, I-10126 Torino, Italy*
[3]*Collegio Carlo Alberto, Via Real Collegio 30, I-10024 Moncalieri, Italy*
[4]*Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129 Torino, Italy*
(Received 1 September 2011; published 14 November 2011)

In this paper we discuss a novel data compression technique for binary symmetric sources based on the cavity method over GF(q), the Galois Field of order q. We present a scheme of low complexity and near-optimal empirical performance. The compression step is based on a reduction of a sparse low-density parity-check code over GF(q) and is done through the so-called reinforced belief-propagation equations. These reduced codes appear to have a nontrivial geometrical modification of the space of codewords, which makes such compression computationally feasible. The computational complexity is $O(dnq \log_2 q)$ per iteration, where d is the average degree of the check nodes and n is the number of bits. For our code ensemble, decompression can be done in a time linear in the code's length by a simple leaf-removal algorithm.

DOI: 10.1103/PhysRevE.84.051111 PACS number(s): 02.50.−r, 89.90.+n, 75.10.Hk

I. INTRODUCTION

The relation between information theory and statistical mechanics of disordered systems has been long established [1,2]. Since then, various techniques from statistical physics of disordered systems have been used not only to assess the theoretical bounds of the achievable performance but also to provide practical encoding and decoding methods for lossy data compression. In particular, both cavity method and replica symmetry breaking techniques have been used to demonstrate the Shannon results and assess the performance of codes defined on sparse factor graphs [3–6].

In this paper we address the classical problem of finding an efficient lossy compression scheme for a generic binary symmetric source. This objective is reached by exploiting some unexpected features of the cavity method when applied to graphical codes defined over a finite field algebra of high order.

Given any realization $\mathbf{y} \in \{0,1\}^n$ of a symmetric Bernoulli process $\mathbf{Y}$, the goal is to compress $\mathbf{y}$ by mapping it to a shorter binary vector such that an approximate reconstruction of $\mathbf{y}$ is possible within a given fidelity criterion. More precisely, suppose $\mathbf{y}$ is mapped to the binary vector $\mathbf{x} \in \{0,1\}^k$ with $k < n$ and $\hat{\mathbf{y}}$ is the reconstructed source sequence. The quantity $R = k/n$ is called the compression rate. The fidelity or distortion is measured by the Hamming distance $d_H(\mathbf{y},\hat{\mathbf{y}}) = (1/n)\sum_{i=1}^{n} |y_i - \hat{y}_i|$. The goal is to minimize the average Hamming distortion $D = \mathbb{E}[d_H(\mathbf{Y},\hat{\mathbf{Y}})]$ for any given rate. The asymptotic limit, known as the rate-distortion function, is given by $R(D) = 1 - H(D)$ for any $D \in [0,0.5]$ where $H(D) = -D\log_2 D - (1-D)\log_2(1-D)$ is the binary entropy function.

Our approach in this paper is based on Low-Density Parity-Check (LDPC) codes. Let $\mathcal{C}$ be a LDPC code with $k \times n$ generator matrix $\mathbf{G}$ and $m \times n$ parity check matrix $\mathbf{H}$. Encoding in lossy compression can be implemented like decoding in error correction. Given a source sequence $\mathbf{y}$, we look for a codeword $\hat{\mathbf{y}} \in \mathcal{C}$ such that $d_H(\mathbf{y},\hat{\mathbf{y}})$ is minimized. The compressed sequence $\mathbf{x}$ is obtained as the k information bits that satisfy $\hat{\mathbf{y}} = \mathbf{G}^T\mathbf{x}$.

Even though LDPC codes have been successfully used for various types of lossless data compression schemes [7], and the existence of ensembles that asymptotically achieve the Shannon's bound for binary symmetric sources has been proved [8], they have not been fully explored for lossy data compression. It is partially due to the long-standing problem of finding a practical source-coding algorithm for LDPC codes, and partially because Low-Density Generator Matrix (LDGM) codes, as dual of LDPC codes, seemed to be more adapted for source coding and have received more attention in the past few years.

In Ref. [9], Martinian and Yedidia show that quantizing a ternary memoryless source coding with erasures is dual of the transmission problem over a binary erasure channel. They also prove that LDGM codes, as dual of LDPC codes, combined with a modified Belief Propagation (BP) algorithm can saturate the corresponding rate-distortion bound. Following their pioneering work, LDGM codes have been extensively studied for lossy compression by several researchers [10–15]. In a series of parallel works, several researchers have used techniques from statistical physics to provide nonrigorous analysis of LDGM codes [3,5,16]. However, LDGM codes seem to perform well only for rates smaller than 0.5. As we will see, our proposed LDPC codes perform very near to the rate distortion bound for rates larger than 0.5. For smaller rates the loss in performance can be compensated by increasing the complexity (number of iterations) of our coding scheme.

In terms of practical algorithms, lossy compression is still an active research topic. In particular, an asymptotically optimal low-complexity compressor with near-optimal empirical performance has not been found yet. Almost all suggested algorithms have been based on some kind of decimation of belief or survey propagation that suffers a computational complexity of $O(n^2)$ [11,15,16]. One exception is the algorithm proposed by Murayama [5]. When the generator matrix is ultrasparse (US), the algorithm was empirically shown to perform very near to the associated capacity needing $O(n)$ computations. A generalized form of this algorithm, called Reinforced Belief Propagation (RBP) [17], was used in a dual setting [18], for ultrasparse LDPC codes (US-LDPC) over GF(2) for lossy

 ©2011 American Physical Society

compression [19]. The main drawback in both cases is the nonoptimality of US structures over GF(2) [5,10,12]. As we will see, this problem can be overcome by increasing the size of the finite field.

Estimation of the weight-enumerating function shows that randomly constructed US-LDPC codes over GF(q) nearly achieve the rate-distortion bound for $q \geqslant 64$. Despite this, practical encoding for these codes is a hard task. The main problem seems to stem from geometrical properties of the configuration space: As the codes are good for channel coding, solutions are isolated and well separated. This characteristic is known to make the encoding problem difficult to solve for iterative and local algorithms [20,21]. To improve this step we introduce the ensemble of *b-reduced* US-LDPC codes, which by eliminating a logarithmic number of constraints from US-LDPC codes just multiplies the number of codewords by a polynomial. This change has a negligible effect in the rate, while having a large effect on the performance of the scheme. Indeed, this modification not only improves the convergence of the RBP algorithm on encoding, but also provides us with a simple efficient decoding algorithm.

The rest of this paper is organized as follows. Section II reviews the code ensemble which we use for lossy compression. Section III describes the RBP algorithm over GF(q). We also discuss briefly the complexity and implementation of the RBP algorithm. In Sec. IV we describe iterative encoding and decoding for our ensemble and then present the corresponding simulation results in Sec. V. In Sec. VI we discuss some properties of the geometry of the space of codewords in relation to the reduction procedure. A brief discussion on further research is given in Sec. VII.

II. LDPC CODES OVER GF(q)

In this section we introduce the US-LDPC codes over GF(q). As we will see later, near Shannon's bound lossy compression is possible using these codes and BP-like iterative algorithms.

A. (λ,ρ) Ensemble of GF(q) LDPC codes

We follow the methods and notations in Ref. [22] to construct irregular bipartite factor graphs. What distinguishes GF(q) LDPC codes from their binary counterparts is that each edge (i,j) of the factor graph has a label $h_{i,j} \in \mathrm{GF}(q)\setminus\{0\}$. In other words, the nonzero elements of the parity-check matrix of a GF(q) LDPC codes are chosen from the nonzero elements of the field GF(q). Denoting the set of variable nodes adjacent to a check node j by $\mathcal{N}(j)$, a word $\mathbf{c}$ with components in GF(q) is a codeword if at each check node j the equation $\sum_{i\in\mathcal{N}(j)} h_{i,j}c_i = 0$ holds.

An ensemble of GF(q) LDPC codes is characterized by two generating polynomials $\lambda(x) = \sum_{i=1}^{d_v} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{i=1}^{d_c} \rho_i x^{i-1}$ where λ_i (ρ_i) denotes the fraction of edges incident on variable(check) nodes of degree i and d_v (d_c) is maximum variable (check) node degree.

A (λ,ρ) GF(q) LDPC code can be constructed from a (λ,ρ) LDPC code by random independent and identically distributed selection of the matrix coefficients with uniform probability from GF(q)\{0}. Note that this may not be an optimal way for selecting the coefficients. For more details on code construction and coefficient selection we refer the readers to Refs. [23] and [24].

B. Code construction for lossy compression

It is well known that the parity check matrix of a GF(q) LDPC code, optimized for binary input channels, is much sparser than the one of a binary LDPC code with same parameters [23,25]. In particular, when $q \geqslant 2^6$, the best error rate results on binary input channels is obtained with the lowest possible variable node degrees, i.e., when almost all variable nodes have degree two. Such codes have been called *ultrasparse* or *cyclic* LDPC codes in the literature. In the rest of this paper we call a LDPC code ultrasparse (US-LDPC) if all variable nodes have degree two. We will mainly concentrate on codes in which the parity check's degree distribution is concentrated on at most two different degree values, for any given rate.

Given a linear code $\mathcal{C}$ and an integer b, a b-reduction of $\mathcal{C}$ is the code obtained by randomly eliminating b parity-check nodes of $\mathcal{C}$. For reasons to be made clear in Sec. IV, we are mainly interested in b-reduction of GF(q) US-LDPC codes for small values of b ($1 \leqslant b \leqslant 5$).

GF(q) US-LDPC codes have been extensively studied for transmission over noisy channels [25–27]. The advantage of using such codes is twofold. On the one hand, by moving to sufficiently large fields, it is possible to obtain near-capacity-achieving codes. On the other hand, the extreme sparseness of the factor graph is well suited for iterative message-passing decoding algorithms. Despite the state-of-the-art performance of moderate length GF(q) US-LDPC channel codes, they have been less studied for lossy compression, the main reason being the lack of fast suboptimal algorithms. In the next section we present the RBP algorithm over GF(q) and then show that practical encoding for lossy compression is possible by using RBP as the encoding algorithm for the ensemble of b-reduced US-LDPC codes.

III. REINFORCED BELIEF PROPAGATION ALGORITHM IN GF(q)

In this section first we briefly review the RBP equations over GF(q), and then we discuss in some detail the complexity of the algorithm following Declercq and Fossorier [27].

A. BP and RBP equations

The GF(q) Belief Propagation (BP) algorithm is a straightforward generalization of the binary case, where the messages are q-dimensional vectors.

Let μ_{vf}^{ℓ} denote the message vector form variable node v to check node f at the ℓth iteration. For each symbol $a \in$GF(q), the ath component of μ_{vf}^{ℓ} is the probability that variable v takes the value a and is denoted by $\mu_{vf}^{\ell}(a)$. Similarly, μ_{fv}^{ℓ} denotes the message vector from check node f to variable node v at the iteration ℓ and $\mu_{fv}^{\ell}(a)$ is its ath component. Also let $\mathcal{N}(v)$ [$\mathcal{M}(f)$] denote the set of check (variable) nodes adjacent to v (f) in a given factor graph.

Constants μ_v^1 are initialized according to the prior information. The BP updating rules can be expressed as follows:

Local function to variable:

$$\mu^{\ell}_{fv}(a) \propto \sum_{\mathrm{Conf}_{(v,f)}(a)} \prod_{v' \in \mathcal{M}(f)\setminus\{v\}} \mu^{\ell}_{v'f}(a); \tag{1}$$

Variable to local function:

$$\mu^{\ell+1}_{vf}(a) \propto \mu^{1}_{v}(a) \prod_{f' \in \mathcal{N}(v)\setminus\{f\}} \mu^{\ell}_{f'v}(a), \tag{2}$$

where $\mathrm{Conf}_{(v,f)}(a)$ is the set of all configurations of variables in $\mathcal{M}(f)$ that satisfy the check node f when the value of variable v is fixed to a. We define the marginal function of variable v at iteration $\ell+1$ as

$$\mathbf{g}^{\ell+1}_{v}(a) \propto \mu^{1}_{v}(a) \prod_{f \in \mathcal{N}(v)} \mu^{\ell}_{fv}(a). \tag{3}$$

The algorithm converges after t iterations if and only if for all variables v and all function nodes f,

$$\mu^{t+1}_{fv} = \mu^{t}_{fv}$$

up to some precision ϵ. A predefined maximum number of iterations $\ell_{\max}$ and the precision parameter ϵ are the input to the algorithm.

RBP is a generalization of BP in which the messages from variable nodes to check nodes are modified as follows:

$$\mu^{\ell+1}_{vf}(a) \propto \left[\mathbf{g}^{\ell}_{v}(a)\right]^{\gamma(\ell)} \mu^{1}_{v}(a) \prod_{f' \in \mathcal{N}(v)\setminus\{f\}} \mu^{\ell}_{f'v}(a), \tag{4}$$

where $\gamma(\ell) : [0,1] \longrightarrow [0,1]$ is a nondecreasing function and $\mathbf{g}^{\ell}_{v}$ is the marginal function of variable v at iteration ℓ. The marginals for RBP are defined as

$$\mathbf{g}^{\ell+1}_{v}(a) \propto \left[\mathbf{g}^{\ell}_{v}(a)\right]^{\gamma(\ell)} \mu^{1}_{v}(a) \prod_{f \in \mathcal{N}(v)} \mu^{\ell}_{fv}(a). \tag{5}$$

Intuitively, RBP equations can be thought as a sort of "soft-decimation" procedure. Indeed, in a decimation procedure [28], the BP equations are iterated until convergence, and then an infinite external field with the same sign of the local field is applied to one or more variables, and the process is repeated (until all variables receive an infinite field). In the RBP procedure, every variable receives a finite external field that is proportional to its own local field [the proportionality factor being $\gamma(\ell)$]. Moreover, the two time scales (convergence and external field update) are intermixed.

It is convenient to define γ to be

$$\gamma(\ell) = 1 - \gamma_0 \gamma_1^{\ell}, \tag{6}$$

where γ_0, γ_1 are in $[0,1]$.

Note that when $\gamma_1 = 1$, RBP is the same as the algorithm presented in Ref. [5] for lossy data compression. In this case it is easy to show that the only fixed points of RBP are configurations that satisfy all the constraints.

B. Efficient implementation

Ignoring the normalization factor in (2), to compute all variable to check-node messages at a variable node of degree d_v we need $O(qd_v)$ computations. A naive implementation of GF(q) BP has computational complexity of $O(d_f^2 q^2)$ operations at each check node of degree d_f. This high complexity is mainly due to the sum in (1), which can be interpreted as a discrete convolution of probability density functions. Efficient implementations of function to variable node messages based on the Discrete Fourier Transform (DFT) have been proposed by several authors; see, for example, Refs. [23,27,29,30] and the references therein. The procedure consists in using the identity $\bigodot_{v' \in \mathcal{M}(f)\setminus\{v\}} \mu_{v'f} = \mathcal{F}^{-1}[\prod_{v' \in \mathcal{M}(f)\setminus\{v\}} \mathcal{F}(\mu_{v'f})]$ where the $\bigodot$ symbol denotes convolution of functions over GF(q), and the product on the right-hand side is the pointwise product of real-valued functions.

Assuming $q = 2^p$, the Fourier transform of each message $\mu_{v'f}$ needs $O(qp)$ computations, and hence the total computational complexity at check node f can be reduced into $O(d_f^2 qp)$. This complexity can be further reduced to $O(d_f qp)$ by using the fact that $\prod_{v' \in \mathcal{M}(f)\setminus\{v\}} \mathcal{F}(\mu_{vf}) = \prod_{v' \in \mathcal{M}(f)} \mathcal{F}(\mu_{v'f})/\mathcal{F}(\mu_{vf})$, or alternatively by using the summation strategy described in Ref. [31], which has the same complexity but is numerically more stable. Therefore, the total number of computations per iteration is $O(dqpn)$, where d is the average check-node degree. A prototype C++ implementation of these equations is provided in source form [32].

IV. ITERATIVE LOSSY COMPRESSION

In the following three subsections we first describe a simple method for identifying information bits of a b-reduced US-LDPC code and then present a near-optimal scheme for iterative compression (encoding) and linear decompression (decoding).

A. Identifying a set of information bits

For b-reduced US-LDPC codes, one can use the *leaf removal* (LR) algorithm to find the information bits in linear time. In the rest of this section we briefly review the LR algorithm and show that 1-reduction (removal of a sole check node) of a US-LDPC code significantly changes the intrinsic structure of the factor graph of the original code.

The main idea behind the LR algorithm is that a variable on a leaf of a factor graph can be fixed in such a way that the check node to which it is connected is satisfied [33]. Given a factor graph, LR starts from a leaf and removes it as well as the check node it is connected to. LR continues this process until no leaf remains. The residual subgraph is called the *core*. Note that the core is independent of the order in which leaves (and hence the corresponding check nodes) are removed from the factor graph. This implies that also the number of steps needed to find the core does not depend on the order on which leaves are chosen.

While US-LDPC codes have a complete core, i.e., there is no leaf in their factor graph, the b-reduction of these codes have empty core. Our simulations also indicate that even 1-reduction of a code largely improves the encoding under RBP algorithm (see Sec. V). How RBP exploits this property is the subject of ongoing research.

As we have mentioned, the LR algorithm can be also used to find a set of information bits of a given US-LDPC code. Let us examine the LR algorithm in more detail. At any step t of LR algorithm, a leaf variable node v_t attached to a factor node f_t is selected. Denote by F_t the remaining leaf variable nodes

attached to check node f_t (F_t could be empty if there are no other leaves attached to it). Now we remove check node f_t and leaf nodes in $F_t \cup \{v_t\}$, and repeat. Under the hypothesis that the original graph was connected, this process is guaranteed to finish at some time T with the empty graph, as at each step except the last one, at least one leaf is created.

It is easy to see that at each step, if we fix the values of all variables except those in $F_t \cup \{v_t\}$, then for each configuration of variables in F_t, the value of variable v_t is uniquely determined. Therefore, $\cup_{t=1\ldots T} F_t = \{w_1, \ldots, w_{N-T}\}$ will form a set of information bits. Indeed, using the ordering of the variable indices $v_T, \ldots, v_1, w_1, \ldots, w_{N-T}$, the check matrix becomes upper triangular, and each solution can be found by back substitution in linear time once information bits are fixed.

B. Iterative encoding

Suppose a code of rate R and a source sequence $\mathbf{y}$ is given. In order to find the codeword $\hat{\mathbf{y}}$ that minimizes $d_H(\hat{\mathbf{y}}, \mathbf{y})$, we will employ the RBP algorithm with a strong prior $\mu_v^1(a) = \exp[-L d_H(y_v, a)]$ centered around $\mathbf{y}$. The sequence of information bits of $\hat{\mathbf{y}}$ is the compressed sequence and is denoted by $\mathbf{x}$. In order to process the encoding in GF(q), we first need to map $\mathbf{y}$ into a sequence in GF(q). This can be simply done by grouping p bits together and using the binary representation of the symbols in GF(q).

C. Linear decoding

Given the sequence of information bits $\mathbf{x}$, the goal of the decoder is to find the corresponding codeword $\hat{\mathbf{y}}$. This can be done by calculating the $\mathbf{G}^T\mathbf{x}$, which in general needs $O(n^2)$ computations. One of the advantages of our scheme is that it allows for a linear complexity iterative decoding. The decoding can be performed by iteratively fixing variables following the inverse steps of the LR algorithm; at each step t only one noninformation bit is unknown (variable v_t), and its value can be determined from the parity check f_t. For a sparse parity-check matrix, the number of needed operations is $O(n)$. It is straightforward to show that a code has an empty core if and only if there exists a permutation of columns of the corresponding parity-check matrix $\mathbf{H}$ such that $h_{ij} \neq 0$ for $i = j$ and $h_{ij} = 0$ for all $i > j$. The decoding procedure is equivalent to back-substitution on this permutated triangular matrix.

V. SIMULATION RESULTS

A. Approximating the weight enumeration function by BP

Given an initial vector $\mathbf{y}$, and a probability distribution $P(\mathbf{c})$ over all configurations, the P-average distance from $\mathbf{y}$ can be computed by

$$D_P(\mathbf{y}) = \sum_i \sum_{c_i} P(c_i) d_H(c_i, y_i), \tag{7}$$

where $P(c_i)$ is the set of marginals of P. On the other hand, the entropy of the distribution P is defined by

$$S(P) = -\sum_{\mathbf{c}} P(\mathbf{c}) \log P(\mathbf{c}). \tag{8}$$

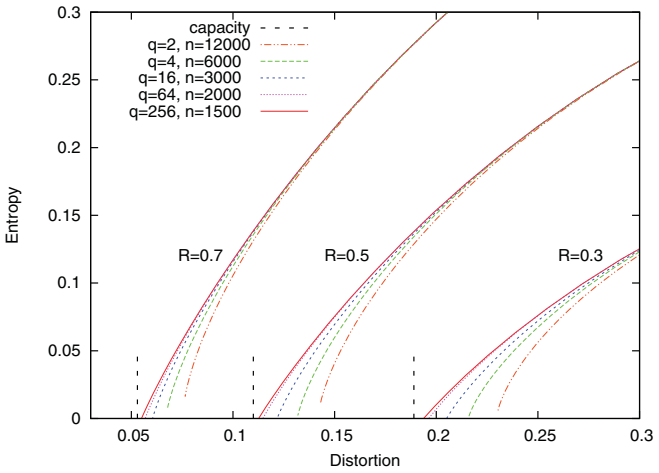

FIG. 1. (Color online) The approximate WEF of GF(q) US-LDPC codes as a function of q for a same block length in binary digits.

Even though it is a hard problem to calculate analytically both marginals and $S(P)$ of a given code, one may approximate them using messages of the BP algorithm at a fixed point [34]. Assuming the normalized distance is asymptotically a self-averaging quantity for our ensemble, $S(P)$ represents the logarithm of the number of codeword at distance $D_P(\mathbf{y})$ from $\mathbf{y}$. By applying a prior distribution on codewords given by $\exp[-L d_H(\mathbf{c}, \mathbf{y})]$, one is able to sample the subspace of codewords at different distances from $\mathbf{y}$.

Figure 1 demonstrates the weight enumerator function (WEF) of random GF(q) US-LDPC codes for rates 0.3, 0.5, and 0.7 and field orders 2, 4, 16, 64, and 256. The block length is normalized so that it corresponds to $n = 12\,000$ binary digits.

Though BP is not exact over loopy graphs, we conjecture that the WEF calculated for US-LDPC codes is asymptotically exact. This hypothesis can be corroborated by comparing the plot in Fig. 1 with the simulation results we obtained by using the RBP algorithm (Fig. 3).

B. Performance

For the simplicity of the analysis, in all our simulations the parameter γ_1 of the RBP algorithm is fixed to one, and therefore the function γ is constant. We also fix the maximum number of iterations to $\ell_{\max} = 300$. If RBP does not converge after 300 iterations, we simply restart RBP with a new random scheduling. The maximum number of trials allowed in our simulations is $T_{\max} = 5$. The encoding performance depends on several parameters such as γ_0, L, the field order q, and the block length n. In the following we first fix n, q, and L, in order to see how the performance changes as a function of γ_0.

1. *Performance as a function of γ_0*

We will show that with this choice of $\gamma(\ell) = 1 - \gamma_0$ there is a tradeoff, controlled by γ_0, between three main aspects of the performance, namely, average distortion, average number of iterations, and average number of trials. The simulations in this subsection are done for a 5-reduced GF(64) US-LDPC code with length $n = 1600$ and rate $R = 0.33$. The factor graph is made by *Progressive-Edge-Growth* (PEG)

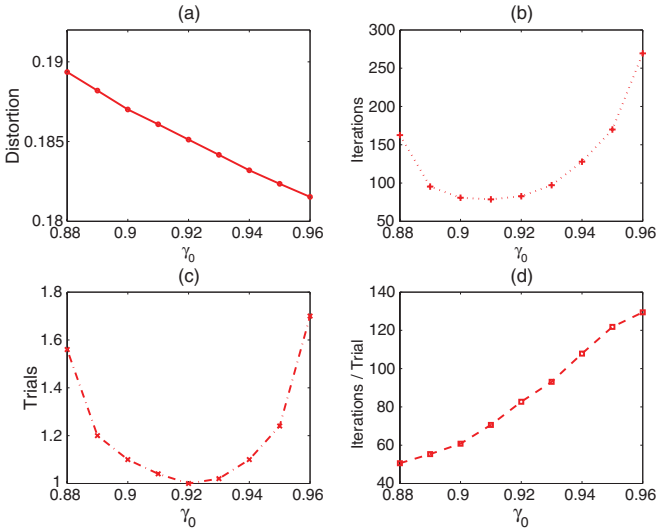

FIG. 2. (Color online) Performance as a function of γ_0 for a PEG graph with $n = 1600$ and $R = 0.33$. The averages are taken over 50 samples. (a) Average distortion as a function of γ_0. For $\gamma_0 > 0.96$ the RBP does not converge within 300 iterations. (b) The average number of iterations. (c) The average number of trials. (d) The average number of iterations needed for each trial. Note that even though average number of iterations show a steep increase as a function of γ_0, the average number of iterations needed per trial increases only linearly.

construction [26]. The rate is chosen purposefully from a region where our scheme has the weakest performance. The Shannon's distortion bound for this rate is approximately 0.1754. Note that the nonmonotonous behavior of RBP as a function of γ_0 could be a result of two concurrent phenomena: For small γ_0 the reinforcement dynamics is too fast and may drive the system to non-codewords; for large γ_0 the reinforcement contribution is small, and the system does not achieve polarization under the predefined iteration bound. In the latter case, better performance may be achieved with $\gamma_1 < 1$ or simply with a different choice of $\gamma(\ell)$.

In Fig. 2 we plot the performance as a function of γ_0. For $\gamma_0 = 0.92$ we achieve a distortion of $D = 0.1851$ needing only 83 iterations on average and without any need to restart RBP for 50 samples. By increasing γ_0 to 0.96, one can achieve an average distortion of 0.1815, which is only 0.15 dB away from the rate-distortion bound needing 270 iterations in average. However, as can be seen in Fig. 2(d), the average number of iterations needed per trial increases only linearly as a function of γ_0.

2. *Performance as a function of R and q*

Figure 3 shows the distortion obtained by randomly generated 5-reduced GF(q) US-LDPC codes for $q = 2$, 16, 64, and 256. The block length is fixed to $n = 12\,000$ binary digits. For each given code with rate larger than or equal to 0.3, we choose γ_0 and L so that the average number of trials does not exceed 2 and the average number of iterations remains less than 300. The optimized values of γ_0 and L are found by simulations and are reported in Table I for $q = 256$. Under these two conditions, we report distortion corresponding to

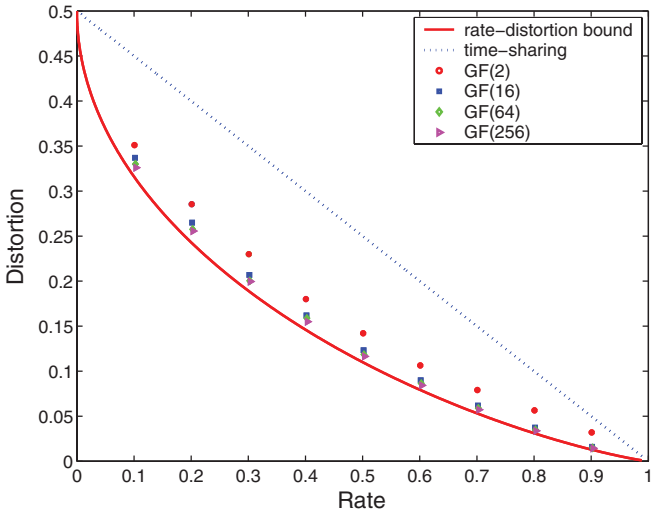

FIG. 3. (Color online) The rate-distortion performance of GF(q) LDPC codes encoded with RBP algorithm for $q = 2$, 16, 64, and 256. The block length is 12 000 binary digits, and each point is the average distortion over 50 samples.

best values of the two parameters averaged over 50 samples. For codes with rates smaller than 0.3, one needs to allow for larger number of iterations and trials.

As the data in Table I indicate, by increasing the rate, both L and $1 - \gamma_0$ increase. Larger values of L impose stronger prior values, indicating that the initialized message distribution is more centered around $\mathbf{y}$. Note that in high rates, if L is not chosen large enough, the loss in performance is substantial. On the other hand, γ_0 regulates the reinforcement needed. Values very near to one for low rates indicate essentially the failure of reinforced strategy. This is not surprising, since in the absence of a codeword near $\mathbf{y}$, forcing BP to find a solution is useless.

3. *Reduction effect on performance of US-LDPC codes*

As we have mentioned, 5-reduced LDPC codes have been used in our simulations. The reduction improves both the convergence of the RBP algorithm and the performance of the our scheme. In Fig. 4 we show how the performance changes as a function of b. The simulations in this subsection are done for a GF(64) US-LDPC code of length $n = 1600$ and rate $R = 0.33$ with PEG constructed factor graph.

VI. DISCUSSION ON REDUCED FACTOR GRAPHS

Our results indicate that the scheme proposed in this paper outperforms the existing methods for lossy compression by low-density structures in both performance and complexity. The main open problem is to understand and analyze the behavior of RBP over b-reduced US-LDPC codes.

TABLE I. The optimal values for L and γ_0 obtained experimentally for $q = 256$.

Rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
L	1.1	1.3	1.5	1.7	1.9	2.3	2.4	2.8	3.8
γ_0	0.98	0.96	0.94	0.92	0.92	0.90	0.90	0.88	0.88

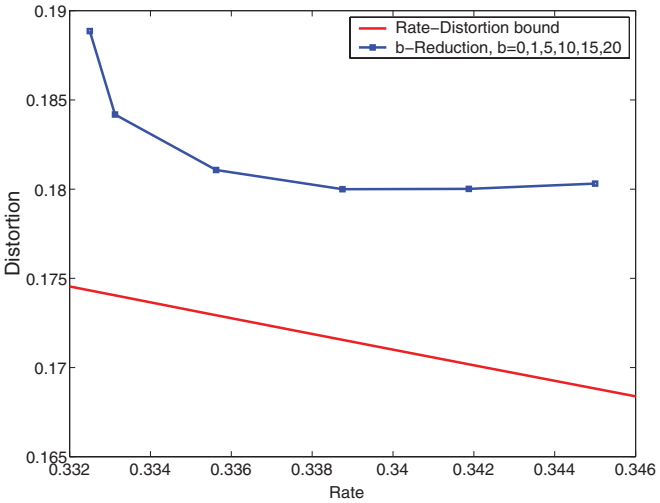

FIG. 4. (Color online) Performance as a function of b for a PEG graph with $n = 1600$ and $R = 0.33$ over GF(64). The averages are taken over 50 samples.

We would like to add a few words to the role of b-reduction. For simplicity, let us concentrate on $q = 2$, though the argument is general. First, note that by removing a parity check node from a code, the number of codewords is doubled. This increment has an asymptotically negligible effect on the compression rate since it increases by only $1/n$, while the robustness may increase. More generally, it is possible to significantly alter the geometry of the solution space while maintaining (asymptotically) the compression rate: For instance, adding a path $\{\mathbf{c} = \mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k = \mathbf{0}\}$ of new codewords from each codeword $\mathbf{c}$ of a given code to the codeword $\mathbf{0}$, such that $d_H(\mathbf{d}_t, \mathbf{d}_{t+1}) = 1/n$ and $k \leqslant n$, multiplies the number of codewords by at most n and thus increases the rate by at most $\log n/n$, which is asymptotically negligible. On the other hand, the codeword space becomes "star-shaped" and thus connected on the hypercube geometry. Note that such modified codes may be terrible for channel coding, as the separation properties may have been severely worsened (e.g., the minimum distance of the code becomes 1).

We think that a similar phenomenon could take place on b-reduced codes. On the one hand, the asymptotic rate for source coding under the proposed scheme is only increased by b/n, and the performance assuming MAP encoding can only improve. On the other hand, we believe that the implied modification of the geometry could ease the task of our iterative encoder. Indeed, it is well known that large separation between solutions makes the problem very hard for iterative and local algorithms [20,21]. In the following we briefly explain some asymptotic implications of 1-reduction on the weight-enumerating function of the US-LDPC code ensemble.

1. 1-Reduced US-LDPC codes

As we have mentioned, canceling a single check node increases the the cardinality of the code by the factor q. As we will see, for each codeword $\mathbf{c}$ of the original US-LDPC, there are created $q - 1$ new codewords, which all have a distance $O(\log n)$ from $\mathbf{c}$. In other words, a cluster of new codewords emerges for each codeword $\mathbf{c}$. In order to see this fact, let v and v' denote two variables of degree one after removing the parity check a. With a probability that approaches one, the check node a and both variables v and v' belong to a loop of length $O(\log n)$ of the original factor graph as $n \to \infty$. After removing a, this loop is broken, and for any codeword of the original factor graph one can obtain new codewords by assigning to v any value from the finite field and changing accordingly the values of all variables in the broken loop. Note that this can be done because all variables in the broken loop have degree two, and v' can be adjusted to satisfy the last checknode in the path from v to v'.

VII. CONCLUSIONS AND PERSPECTIVES

Our main goal in this paper is to provide a low-complexity coding scheme for lossy data compression with near rate-distortion bound performance. We propose a practical iterative encoding-decoding scheme that exploits the geometrical structure of the so-called reduced US low-density parity-check codes. Our proposed algorithm for encoding can be considered as a soft decimation strategy for belief propagation algorithm. The complexity per iteration at the iterative encoder depends linearly on both the length of the code and the order of the field on which the code is defined. The decoding algorithm is based on a leaf removal algorithm that has linear complexity on the proposed sparse factor graphs.

We have investigated the behavior of our scheme for various field orders and parameters of the proposed algorithm. In particular, we approximately calculate the weight-enumerating function of US-LDPC codes as a function of field order using the BP algorithm. Our estimations show that US-LDPC codes over GF(q) nearly achieve the rate-distortion bound for $q \geqslant 64$. Though BP is not exact over loopy graphs, we conjecture that the WEF calculated for US-LDPC codes is asymptotically exact. This hypothesis is corroborated by the simulation results we obtained by using RBP algorithm.

Our research can be expanded in several directions. For example, it is interesting to study other US ensembles sharing similar properties, e.g., where just a certain fraction of variable nodes of degree one is allowed. Several directions could be explored in order to obtain more efficient coding schemes: other choices of the reinforcement rate $\gamma(\ell)$, choices of random codes and coefficient selection, and a $L \to \infty$ version of the encoder along the lines of [27], as it could allow much lower computational complexity. Work is in progress in these directions.

ACKNOWLEDGMENTS

The authors wish to thank Guido Montorsi and Abolfazl Ramezanpour for valuable suggestions and useful discussions. RZ acknowledges ERC grant OPTINF 267915. Support from EC grant STAMINA 265496 is also acknowledged by AB and RZ.

[1] N. Sourlas, Nature (London) **339**, 693 (1989).
[2] H. Nishimori, *Statistical Physics of Spin Glasses and Information Processing* (Oxford University Press, Oxford, 2001).
[3] T. Hosaka and Y. Kabashima, Physica A **365**, 113 (2006).
[4] K. Mimura, J. Phys. A: Math. Theor. **42**, 135002 (2009).
[5] T. Murayama, Phys. Rev. E **69**, 035105(R) (2004).
[6] T. Murayama and M. Okada, J. Phys. A: Math. Gen. **36**, 11123 (2003).
[7] G. Caire, S. Shamai, and S. Verdu, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* (American Mathematical Society, Providence, RI, 2004).
[8] Y. Matsunaga and H. Yamamoto, IEEE Trans. Inf. Theory **49**, 2225 (2003).
[9] E. Martinian and J. Yedidia, Proceedings of Allerton Conference on Communication, Control and Computing, 131 (2003).
[10] A. Dimakis, M. Wainwright, and K. Ramchandran, IEEE Information Theory Workshop, 650 (2007).
[11] T. Filler and J. Fridrich, Proceedings of Allerton Conference on Communication, Control and Computing, 495 (2007).
[12] S. Kudekar and R. Urbanke, Int. Symp. on Turbo Codes and Related Topics, 379 (2008).
[13] E. Martinian and M. J. Wainwright, IEEE Int. Symp. Inf. Theory 484 (2006).
[14] E. Martinian and M. J. Wainwright, IEEE Trans. Inf. Theory **55**, 1061 (2009).
[15] M. J. Wainwright, E. Maneva, and E. Martinian, IEEE Trans. Inf. Theory **56**, 1351 (2010).
[16] S. Ciliberti, M. Mezard, and R. Zecchina, Phys. Rev. Lett. **95**, 038701 (2005).
[17] A. Braunstein and R. Zecchina, Phys. Rev. Lett. **96**, 030201 (2006).
[18] A. Braunstein, F. Kayhan, G. Montorsi, and R. Zecchina, IEEE Int. Symp. on Inf. Theory, 1891 (2007).
[19] F. Kayhan and T. Tanaka, International Symposium on Turbo Codes and Related Topics, 396 (2008).
[20] L. Dall'Asta, A. Ramezanpour, and R. Zecchina, Phys. Rev. E **77**, 031118 (2008).
[21] L. Zdeborová and F. Krzákała, Phys. Rev. E **76**, 031131 (2007).
[22] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, IEEE Trans. Inf. Theory **47**, 585 (2001).
[23] A. Bennatan and D. Burshtein, IEEE Trans. Inf. Theory **52**, 549 (2006).
[24] D. MacKay, http://www.inference.phy.cam.ac.uk/mackay/CodesGallager.html (2003).
[25] M. C. Davey and D. MacKay, IEEE Comm. Lett. **2**, 165 (1998).
[26] X. Y. Hu and E. Eleftheriou, IEEE Int. Conf. on Communications, 528 (2004).
[27] D. Declercq and M. Fossorier, IEEE Trans. Communication Theory **55**, 633 (2007).
[28] A. Braunstein, M. Mezard, and R. Zecchina, Rand. Struct. Algorithms **27**, 201 (2005).
[29] T. J. Richardson and R. Urbanke, IEEE Trans. Inf. Theory **47**, 599 (2001).
[30] D. J. C. MacKay, IEEE Trans. Inf. Theory **45**, 399 (1999).
[31] A. Braunstein, R. Mulet, and A. Pagnani, BMC Bioinformatics **9**, 240 (2008).
[32] http://www.polito.it/cmp/code/gfrbp (2011).
[33] M. Mezard, F. Ricci-Tersenghi, and R. Zecchina, J. Stat. Phys. **111**, 505 (2003).
[34] J. S. Yedidia, W. T. Freeman, and Y. Weiss, IEEE Trans. Inf. Theory **51**, 2282 (2005).