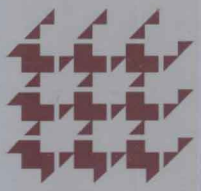

DIMACS

Series in Discrete Mathematics and Theoretical Computer Science

Volume 68

Algebraic Coding Theory and Information Theory

DIMACS Workshop
Algebraic Coding Theory and Information Theory
December 15–18, 2003
Rutgers University
Piscataway, New Jersey

A. Ashikhmin
A. Barg
Editors

American Mathematical Society

DIMACS

Series in Discrete Mathematics
and Theoretical Computer Science

Volume 68

Algebraic Coding Theory and Information Theory

DIMACS Workshop
Algebraic Coding Theory and Information Theory
December 15–18, 2003
Rutgers University
Piscataway, New Jersey

A. Ashikhmin
A. Barg
Editors

Center for Discrete Mathematics
and Theoretical Computer Science
A consortium of Rutgers University, Princeton University,
AT&T Labs–Research, Bell Labs (Lucent Technologies),
NEC Laboratories America, and Telcordia Technologies
(with partners at Avaya Labs, HP Labs, IBM Research,
Microsoft Research, and Stevens Institute of Technology)

American Mathematical Society

This DIMACS volume contains papers presented at the workshop on "Algebraic Coding Theory and Information Theory" held at the DIMACS Center, Rutgers University, Piscataway, New Jersey in December 2003.

2000 *Mathematics Subject Classification.* Primary 94A24, 94A29, 94B15, 94B25, 94B35, 94B70.

Library of Congress Cataloging-in-Publication Data

Algebraic coding theory and information theory : DIMACS workshop, algebraic coding theory and information theory, December 15–18, 2003, Rutgers University, Piscataway, New Jersey / A. Ashikhmin, A. Barg, editors.

p. cm. — (DIMACS series in discrete mathematics and theoretical computer science, ISSN 1052-1798 ; v. 68)

Includes bibliographical references.

ISBN 0-8218-3626-9 (alk. paper)

1. Information theory in mathematics—Congresses. 2. Coding theory—Congresses. 3. Algebraic logic—Congresses. I. Ashikhmin, A. (Alexei), 1966– II. Barg, Alexander, 1960– III. Series.

QA10.4.A44 2005
003′.54—dc22 2005047070

Algebraic Coding Theory and Information Theory

Foreword

A workshop on Algebraic Coding Theory and Information Theory was held on December 15–18, 2003 at Rutgers University. We would like to express our appreciation to Alexei Ashikhmin, Alexander Barg and Iwan Duursma for their efforts to organize and plan this successful conference.

The workshop was part of the 2001–2005 Special Focus on Computational Information Theory and Coding. We extend our thanks to Robert Calderbank, Chris Rose, Amin Shokrollahi, Emina Soljanin, and Sergio Verdú for their work as special focus organizers.

The workshop brought together theoreticians and practitioners working on algebraic coding theory and information/communications theory, with an emphasis on establishing links between these areas. The main themes of the workshop included linear channel coding, graph-theoretic ideas as they apply to codes and lattices, Reed-Solomon codes, and considerations involving capacity. These are all major themes in current research in algebraic coding theory.

DIMACS gratefully acknowledges the generous support that makes these programs possible. Special thanks go to the National Science Foundation, the New Jersey Commission on Science and Technology, and to DIMACS partners at Rutgers, Princeton, AT&T Labs–Research, Bell Labs, NEC Laboratories America, and Telcordia Technologies, and affiliate partners Avaya Labs, HP Labs, IBM Research, Microsoft Research, and Stevens Institute of Technology.

Fred S. Roberts
Director

Robert Tarjan
Co-Director for Princeton

Preface

Recent years have witnessed an increased interest in the study of problems in theoretical communication whose solution relies on the synergy of methods of coding and information theory. The present volume collects papers presented at or inspired by the workshop "Algebraic Coding Theory and Information Theory" held at DIMACS, Rutgers University, Piscataway, New Jersey in December 2003. The workshop was part of a 4-year Special Focus on Computational Information Theory and Coding held by DIMACS and supported by the NSF. The volume opens with the articles of G. Caire et al. and G. I. Shamir that employ diverse ideas from linear channel coding in designing new methods of universal lossless data compression. The next four papers (by K. W. Shum and I. F. Blake, A. Barg and G. Zémor, M. R. Sadeghi and D. Panario, and J. S. Yedidia) are devoted to the use of graph-theoretic ideas in construction and decoding of codes and lattices. M. El-Khamy and R. J. McEliece study optimal multiplicity assignment in soft-decision list decoding algorithms of Reed-Solomon codes. The papers by S. Litsyn and A. Shpunt and G. Kramer and S. Savari are devoted to capacity results in various transmission scenarios. Finally, the paper by R. G. Cavalcante et al. puts forward a new approach to the design of signal constellations based on allocations of signal points on surfaces.

We are very grateful to the DIMACS Center for providing financial and organizational support for the workshop. In the initial stages of the workshop preparation we were helped by Iwan Duursma of the University of Illinois at Urbana-Champaign to whom we express our sincere gratitude.

Alexei Ashikhmin
Math. Research
Bell Labs, Lucent Technologies
Murray Hill, NJ 07974
aea@lucent.com

Alexander Barg
Dept. of ECE
University of Maryland
College Park, MD 20742
abarg@umd.edu

38 **Rebecca N. Wright and Peter G. Neumann, Editors,** Network Threats

37 **Boris Mirkin, F. R. McMorris, Fred S. Roberts, and Andrey Rzhetsky, Editors,** Mathematical Hierarchies and Biology

36 **Joseph G. Rosenstein, Deborah S. Franzblau, and Fred S. Roberts, Editors,** Discrete Mathematics in the Schools

35 **Dingzhu Du, Jun Gu, and Panos M. Pardalos, Editors,** Satisfiability Problem: Theory and Applications

34 **Nathaniel Dean, Editor,** African Americans in Mathematics

33 **Ravi B. Boppana and James F. Lynch, Editors,** Logic and Random Structures

32 **Jean-Charles Grégoire, Gerard J. Holzmann, and Doron A. Peled, Editors,** The SPIN Verification System

31 **Neil Immerman and Phokion G. Kolaitis, Editors,** Descriptive Complexity and Finite Models

30 **Sandeep N. Bhatt, Editor,** Parallel Algorithms: Third DIMACS Implementation Challenge

29 **Doron A. Peled, Vaughan R. Pratt, and Gerard J. Holzmann, Editors,** Partial Order Methods in Verification

28 **Larry Finkelstein and William M. Kantor, Editors,** Groups and Computation II

27 **Richard J. Lipton and Eric B. Baum, Editors,** DNA Based Computers

26 **David S. Johnson and Michael A. Trick, Editors,** Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge

25 **Gilbert Baumslag, David Epstein, Robert Gilman, Hamish Short, and Charles Sims, Editors,** Geometric and Computational Perspectives on Infinite Groups

24 **Louis J. Billera, Curtis Greene, Rodica Simion, and Richard P. Stanley, Editors,** Formal Power Series and Algebraic Combinatorics/Séries Formelles et Combinatoire Algébrique, 1994

23 **Panos M. Pardalos, David I. Shalloway, and Guoliang Xue, Editors,** Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding

22 **Panos M. Pardalos, Mauricio G. C. Resende, and K. G. Ramakrishnan, Editors,** Parallel Processing of Discrete Optimization Problems

21 **D. Frank Hsu, Arnold L. Rosenberg, and Dominique Sotteau, Editors,** Interconnection Networks and Mapping and Scheduling Parallel Computations

20 **William Cook, László Lovász, and Paul Seymour, Editors,** Combinatorial Optimization

19 **Ingemar J. Cox, Pierre Hansen, and Bela Julesz, Editors,** Partitioning Data Sets

18 **Guy E. Blelloch, K. Mani Chandy, and Suresh Jagannathan, Editors,** Specification of Parallel Algorithms

17 **Eric Sven Ristad, Editor,** Language Computations

16 **Panos M. Pardalos and Henry Wolkowicz, Editors,** Quadratic Assignment and Related Problems

15 **Nathaniel Dean and Gregory E. Shannon, Editors,** Computational Support for Discrete Mathematics

14 **Robert Calderbank, G. David Forney, Jr., and Nader Moayeri, Editors,** Coding and Quantization: DIMACS/IEEE Workshop

13 **Jin-Yi Cai, Editor,** Advances in Computational Complexity Theory

12 **David S. Johnson and Catherine C. McGeoch, Editors,** Network Flows and Matching: First DIMACS Implementation Challenge

Titles in This Series

68 **A. Ashikhmin and A. Barg, Editors,** Algebraic Coding Theory and Information Theory

67 **Ravi Janardan, Michiel Smid, and Debasish Dutta, Editors,** Geometric and Algorithmic Aspects of Computer-Aided Design and Manufacturing

66 **Piyush Gupta, Gerhard Kramer, and Adriaan J. van Wijngaarden, Editors,** Advances in Network Information Theory

65 **Santosh S. Vempala,** The Random Projection Method

64 **Melvyn B. Nathanson, Editor,** Unusual Applications of Number Theory

63 **J. Nešetřil and P. Winkler, Editors,** Graphs, Morphisms and Statistical Physics

62 **Gerard J. Foschini and Sergio Verdú, Editors,** Multiantenna Channels: Capacity, Coding and Signal Processing

61 **M. F. Janowitz, F.-J. Lapointe, F. R. McMorris, B. Mirkin, and F. S. Roberts, Editors,** Bioconsensus

60 **Saugata Basu and Laureano Gonzalez-Vega, Editors,** Algorithmic and Quantitative Real Algebraic Geometry

59 **Michael H. Goldwasser, David S. Johnson, and Catherine C. McGeoch, Editors,** Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges

58 **Simon Thomas, Editor,** Set Theory: The Hajnal Conference

57 **Eugene C. Freuder and Richard J. Wallace, Editors,** Constraint Programming and Large Scale Discrete Optimization

56 **Alexander Barg and Simon Litsyn, Editors,** Codes and Association Schemes

55 **Ding-Zhu Du, Panos M. Pardalos, and Jie Wang, Editors,** Discrete Mathematical Problems with Medical Applications

54 **Erik Winfree and David K. Gifford, Editors,** DNA Based Computers V

53 **Nathaniel Dean, D. Frank Hsu, and R. Ravi, Editors,** Robust Communication Networks: Interconnection and Survivability

52 **Sanguthevar Rajasekaran, Panos Pardalos, and D. Frank Hsu, Editors,** Mobile Networks and Computing

51 **Pierre Hansen, Patrick Fowler, and Maolin Zheng, Editors,** Discrete Mathematical Chemistry

50 **James M. Abello and Jeffrey Scott Vitter, Editors,** External Memory Algorithms

49 **Ronald L. Graham, Jan Kratochvíl, Jaroslav Nešetřil, and Fred S. Roberts, Editors,** Contemporary Trends in Discrete Mathematics

48 **Harvey Rubin and David Harlan Wood, Editors,** DNA Based Computers III

47 **Martin Farach-Colton, Fred S. Roberts, Martin Vingron, and Michael Waterman, Editors,** Mathematical Support for Molecular Biology

46 **Peng-Jun Wan, Ding-Zhu Du, and Panos M. Pardalos, Editors,** Multichannel Optical Networks: Theory and Practice

45 **Marios Mavronicolas, Michael Merritt, and Nir Shavit, Editors,** Networks in Distributed Computing

44 **Laura F. Landweber and Eric B. Baum, Editors,** DNA Based Computers II

43 **Panos Pardalos, Sanguthevar Rajasekaran, and José Rolim, Editors,** Randomization Methods in Algorithm Design

42 **Ding-Zhu Du and Frank K. Hwang, Editors,** Advances in Switching Networks

41 **David Aldous and James Propp, Editors,** Microsurveys in Discrete Probability

40 **Panos M. Pardalos and Dingzhu Du, Editors,** Network Design: Connectivity and Facilities Location

39 **Paul W. Beame and Samuel R Buss, Editors,** Proof Complexity and Feasible Arithmetics

Contents

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume **68**, 2005

Fountain Codes for Lossless Data Compression

Giuseppe Caire, Shlomo Shamai, Amin Shokrollahi, and Sergio Verdú

ABSTRACT. This paper proposes a universal variable-length lossless compression algorithm based on fountain codes. The compressor concatenates the Burrows-Wheeler block sorting transform (BWT) with a fountain encoder, together with the closed-loop iterative doping algorithm. The decompressor uses a Belief Propagation algorithm in conjunction with the iterative doping algorithm and the inverse BWT. Linear-time compression/decompression complexity and competitive performance with respect to state-of-the-art compression algorithms are achieved.

1. Introduction

Noiseless data compression is a key information technology used in innumerable data storage and transmission applications ranging from computer operating systems to modems to lossy compression standards. Although the state-of-the-art has reached a certain level of maturity with data compression algorithms that asymptotically attain the fundamental information theoretic limits with linear computational complexity, they suffer from several shortcomings when used in packetized noisy channels. Mainly for this reason, no payload compression is currently implemented in third-generation standards for high-speed wireless data transmission.

It is known that *linear* fixed-length encoding can achieve for asymptotically large blocklength the minimum achievable compression rate for memoryless sources [**1**] and for arbitrary (not necessarily stationary/ergodic) sources [**2**].

After initial attempts [**3, 4, 5**] to construct linear lossless codes were nonuniversal, limited to memoryless sources and failed to reach competitive performance with standard data compression algorithms, the interest in linear data compression waned. Recently [**6, 7, 8**] came up with a universal lossless data compression algorithm based on irregular low-density parity-check codes which has linear encoding and decoding complexity, can exploit source memory and in the experiments for binary sources presented in [**6, 7, 8**] showed competitive performance with respect to standard compressors such as `gzip`, PPM and `bzip`.

The scheme of [**6, 7, 8**] was based on the important class of sparse-graph error correcting codes called low-density parity check (LDPC) codes. The block-sorting

1991 *Mathematics Subject Classification.* Primary 68P30, 94A29; Secondary 94A45, 62B10.
Key words and phrases. Noiseless Data Compression, Universal algorithms, Error Correcting Codes, Source Coding, Sources with Memory, Block Sorting Transform.
This research was partially supported by NSF Grant CCR-0312879.

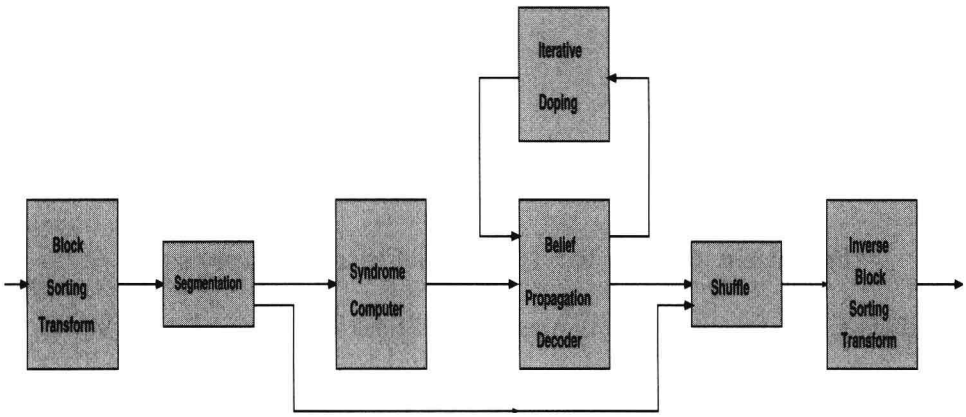

FIGURE 1. Compression/Decompression Scheme for Noiseless Channel

transform (or Burrows-Wheeler transform (BWT)) [**9**] shown in Figure 1 is a one-to-one transformation, which performs the following operation: it generates all cyclic shifts of the given data string and sorts them lexicographically. The last column of the resulting matrix is the BWT output from which the original data string can be recovered, knowing the *BWT index* which is the location of the original sequence in the matrix. The BWT shifts redundancy in the memory to redundancy in the marginal distributions. The redundancy in the marginal distributions is then much easier to exploit at the decoder as the decoding complexity is independent of the complexity of the source model (in particular, the number of states for Markov sources). The output of the BWT (as the blocklength grows) is asymptotically piecewise i.i.d. for stationary ergodic tree sources. The length, location, and distribution of the i.i.d. segments depend on the statistics of the source. The existing universal BWT-based methods for data compression generally hinge on the idea of compression for a memoryless source with an adaptive procedure that learns implicitly the local distribution of the piecewise i.i.d. segments, while forgetting the effect of distant symbols.

In the data compression algorithm of [**6, 7**], the compression is carried out by multiplication of the Burrows-Wheeler Transform of the source string with the parity-check matrix of an error correcting code. Of particular interest are LDPC codes since the belief propagation (BP) decoder is able to incorporate the time-varying marginals at the output of the BWT in a very natural way. The nonidentical marginals produced at the output of the BWT have a synergistic effect with the BP algorithm which is able to iteratively exploit imbalances in the reliability of variable nodes. The universal implementation of the algorithm where the encoder identifies the source segmentation and describes it to the decompressor is discussed in [**8**].

An important ingredient in the compression scheme of [**6, 7, 8**] is the ability to do decompression at the compressor. This enables to tune the choice of the codebook to the source realization and more importantly it enables the use of the Closed-Loop Iterative Doping (CLID) algorithm of [**6, 2**]. This is an efficient algorithm which enables zero-error variable-length data compression with performance which is quite competitive with that of standard data compression algorithms.

In this paper, instead of adopting irregular low-density parity check codes of a given rate approximately matched to the source we adopt a different approach based on rateless *fountain codes*. This class of codes turns out to be more natural for variable-length data compression applications than standard block codes and achieves in general comparable performance to the LDPC-based scheme of [**6, 7, 8**].

The rest of the paper is organized as follows. Section 2 reviews the main features of fountain codes for channel coding. Section 3 gives a brief summary of the principle of belief propagation decoding which is common to both channel and source decoding. Our scheme for data compression with fountain codes is explained in detail in Section 4 in the setting of nonuniversal compression of binary sources. For further background on linear codes for data compression and the closed-loop iterative doping algorithm the reader is referred to [**2**]. The issues related to implementation with nonbinary alphabets are discussed in Section 5, while the modelling module necessary for universal application is discussed in Section 6. Section 7 shows several experiments and comparisons of redundancy with off-the-shelf data compression algorithms run with synthetic sources.

2. Fountain Codes for Channel Coding

Fountain codes [**10, 11**] form a new class of sparse-graph codes designed for protection of data against noise in environments where the noise level is not known a-priori. To achieve this, a fountain code produces a potentially limitless stream of output symbols for a given vector of input symbols. In practical applications, each output symbol is a linear function of the input symbols, and the output symbols are generated independently and randomly, according to a distribution which is chosen by the designer. The sequence of linear combinations of input symbols is known at the decoder. For example, they can be generated by pseudorandom generators with identical seeds.

The *overhead* of the decoding algorithm for a fountain code over a given communication channel is measured as the fraction of additional output symbols necessary to achieve the desired residual error rate. Here, the phrase "additional" is meant to be with respect to the Shannon limit of the underlying channel.

The decoding process for fountain codes is as follows: the receiver collects output symbols and estimates for each received output symbol the amount of information in that symbol. For example, when output symbols are bits, this measure of information can be obtained from the log-likelihood ratio (LLR) of the received bit. If this ratio is λ, then the empirical mutual information between the information symbol and its LLR equals $1 - h(p)$, where $p = 1/(1 + \exp(\lambda))$. The decoder stops collecting output bits as soon as the accumulated information carried by the observed channel outputs exceeds $(1 + \omega)k$, where ω is the overhead associated with the fountain code, and k is the number of input symbols. Then the decoder uses its BP decoding algorithm to recover the input symbols from the information contained in the output symbols.

If the amount of information in the received output symbols is unknown but the channel is known to be memoryless stationary with capacity $C(\lambda)$ parameterized in the single output LLR λ, then decoding of fountain codes can be accomplished as follows: the decoder starts with an optimistic guess λ_1 of the channel parameter, collects an appropriate number of output symbols n_1 such that $k/n_1 = C(\lambda_1) - \delta$, where $\delta > 0$ is some positive rate margin that enforces successful decoding with

high probability, and applies BP decoding based on the guess λ_1. In case the BP decoder is unsuccessful, a predetermined number of additional output symbols is collected such that the total number of output symbols is n_2, and the BP decoding is applied to the new graph using LLR $\lambda_2 = C^{-1}(k/n_2 + \delta)$. In case the BP decoder is unsuccessful, the same process is repeated using $n_3 > n_2$ output symbols and so on, until decoding is successful. In practice, instead of resetting the BP decoder at each new decoding attempt and especially if the initial guess λ_1 is likely to be not far from the true channel parameter, it might be more convenient to keep the same BP decoder running while incorporating the additional collected output symbols.

Discovered by Michael Luby [**10**], LT-codes are one of the first classes of efficient fountain codes for the erasure channel. An extension of this class of codes is the class of Raptor codes [**12**]. These classes of codes are very well suited for solving the compression problem, because the compression algorithm translates to a channel coding problem for a discrete memoryless channel with time-varying transition probability matrix which depends on the source. In applications, it is undesirable to tune the choice of the code to the sequence of transition probabilities, as in universal applications they are not known beforehand. In this case a fountain code is more amenable for universal implementation than other classes of efficient channel codes, such as irregular LDPC codes [**13**], mainly because a single code can be used regardless of the source rate.

For ease of exposition we concentrate on binary fountain codes. Let k be a positive integer, and let $\mathcal{D}$ be a distribution on $\mathbb{F}_2^k$. A Fountain Code ensemble with parameters $(k, \mathcal{D})$ has as its domain the space $\mathbb{F}_2^k$ of binary strings of length k, and as its target space the set of all sequences over $\mathbb{F}_2$, denoted by $\mathbb{F}_2^{\mathbb{N}}$. Formally, a Fountain Code ensemble with parameters $(k, \mathcal{D})$ is a linear map $\mathbb{F}_2^k \to \mathbb{F}_2^{\mathbb{N}}$ represented by an $\infty \times k$ matrix where rows are chosen independently with identical distribution $\mathcal{D}$ over $\mathbb{F}_2^k$. The blocklength of a Fountain Code is potentially infinite, but in our data compression applications we will solely consider truncated Fountain Codes with a number of coordinates which is tailored to the source realization.

The symbols produced by a Fountain Code are called *output symbols*, and the k symbols from which these output symbols are calculated are called *input symbols*. The input and output symbols could be elements of $\mathbb{F}_2$, or more generally the elements of any finite dimensional vector space over $\mathbb{F}_2$. In this paper, we are primarily concerned with Fountain Codes over the field $\mathbb{F}_2$, in which symbols are bits.

A special class of Fountain Codes is furnished by LT-Codes [**10**]. In this class, the distribution $\mathcal{D}$ has a special form described in the following. Let $(\Omega_1, \ldots, \Omega_k)$ be a distribution on $\{1, \ldots, k\}$ so that Ω_i denotes the probability that the value i is chosen. Often we will denote this distribution by its generator polynomial $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$. The distribution $\Omega(x)$ induces a distribution on $\mathbb{F}_2^k$ in the following way: For any vector $v \in \mathbb{F}_2^k$ the probability of v is $\Omega_w / \binom{k}{w}$, where w is the Hamming weight of v. Abusing notation, we will denote this distribution in the following by $\Omega(x)$ again. An LT-code is a Fountain code with parameters $(k, \Omega(x))$.

The decoding graph of length n of a fountain code with parameters $(k, \Omega(x))$ is a bipartite graph with k nodes on one side (called *input nodes*, corresponding to the input symbols) and n nodes on the other side (called *output nodes*). The output nodes correspond to n output symbols collected at the output of the channel. The decoding graph is the *Tanner graph* of the linear encoder $\mathbb{F}_2^k \to \mathbb{F}_2^n$ obtained by

restricting the fountain encoder mapping to those n components actually observed at the output.

A raptor code [**12**] performs a pre-coding operation with a linear code (e.g., an LDPC code) prior to using an LT-code. Without a pre-coder the average degree of an LT-coder needs to grow at least logarithmically in the length of the input to guarantee a small error probability, since otherwise there would exist input symbols that are not present in any of the linear equations generating the output symbols. The pre-code lowers the error floor present in an LT-code with small average degree.

3. Belief Propagation Decoding

In this section we give a description of the BP algorithm that is used in the decoding process of LT codes. The algorithm proceeds in rounds. In every round messages are passed from input nodes to output nodes, and then from output nodes back to input nodes. The message sent from the input node ι to the output node ω in the ℓth round of the algorithm is denoted by $\mathtt{m}_{\iota,\omega}^{(\ell)}$, and similarly the message sent from an output node ω to an input node ι is denoted by $\mathtt{m}_{\omega,\iota}^{(\ell)}$. These messages are scalars in $\overline{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$. We will perform additions in this set according to the following rules: $a + \infty = \infty$ for all $a \neq -\infty$, and $a - \infty = -\infty$ for all $a \neq \infty$. The values of $\infty - \infty$ and $-\infty + \infty$ are undefined. Moreover, $\tanh(\infty/2) := 1$, and $\tanh(-\infty/2) := -1$.

Every value Y received from the channel has a log-likelihood ratio defined as $\ln(\Pr[Y = 0]/\Pr[Y = 1])$.

For every output node ω, we denote by Z_ω the corresponding log-likelihood ratio.

In round 0 of the BP algorithm, the output nodes send the value 0 to all their adjacent input nodes. Thereafter, the following update rules are used to obtain the messages passed at later rounds:

$$\tanh\left(\frac{\mathtt{m}_{\omega,\iota}^{(\ell)}}{2}\right) := \tanh\left(\frac{Z_\omega}{2}\right) \cdot \prod_{\iota' \neq \iota} \tanh\left(\frac{\mathtt{m}_{\iota',\omega}^{(\ell)}}{2}\right), \tag{3.1}$$

$$\mathtt{m}_{\iota,\omega}^{(\ell+1)} = \sum_{\omega' \neq \omega} \mathtt{m}_{\omega',\iota}^{(\ell)}, \tag{3.2}$$

where the product is over all input nodes adjacent to ω other than ι, and the sum is over all output nodess adjacent to ι other than ω, and $\ell \geq 0$.

After running the BP-algorithm for m rounds, the log-likelihood of each input symbol associated to node ι can be calculated as the sum $\sum_\omega \mathtt{m}_{\omega,\iota}^{(m)}$, where the sum is over all the output nodes ω adjacent to ι. In cases where the pre-code of the Raptor code is decoded separately from its LT-code, one may gather the log-likelihood of the input symbols, and run a decoding algorithm for the pre-code (which may itself be the BP algorithm). In that case, the prior log-likelihoods of the inputs are set to be equal to the calculated log-likelihoods according to (3.1).

In data compression applications, the CLID algorithm introduced in [**6**] runs BP at the encoder and supplies to the decoder the value of the lowest reliability symbol at certain iterations until successful decoding is achieved. Since the decoder runs an identical copy of the BP iterations, it knows the identity of the lowest reliability symbol, without the need to communicate this information explicitly. The symbols supplied by the CLID algorithm are referred to as doped symbols.

4. Data Compression with Fountain Codes

Fountain Codes can be applied to the problem of data compression using the methods described in [**2, 6, 7, 8**]. In this section we discuss three variants of this approach that use LT-Codes in the setting of nonuniversal compression of binary sources.

4.1. Raptor-code fountain approach. In this approach we use as data compressor the encoder of a raptor code, and as decompressor the decoder of a raptor code incorporating the statistics of the source. For a fountain application where a plurality of receivers obtain erased versions of the broadcast data with erasure rates that are unknown at the transmitter, this approach is able to carry out joint source-channel encoding-decoding effectively, both taking into account the redundancy of the source and offering protection against channel erasures. If the source has memory and its statistics are known to the receiver, it is easy to adapt the BWT-based method for fountain applications. In the universal case it is necessary to convey reliably the model (Section 6) to the receivers (using a raptor code for example). In applications where a systematic fountain code is used to send uncompressed redundant data, backward compatibility with receivers that do not do any data decompression is achievable using the method of [**14**].

In conventional (non-fountain) applications with one encoder and one decoder, the approach of this subsection can also be used. In this case, it is necessary to truncate the semiinfinite sequence of compressed bits at the compressor. In a pure compression application where no robustness against channel noise is required, the decompressor can be run at the encoder and the output sequence is truncated as soon as the BP converges to the true source realization. In those unlikely source realizations in which this method offers no compression, the source realization can be sent uncompressed. As it can be expected (and we will see below) this method is not competitive in terms of performance with the CLID-based approach explained in Section 4.4. An alternative to using a raptor code in this setting is to puncture a given code increasingly until the decoder is unable to reconstruct the original source sequence. This is the "decremental redundancy" approach of [**15**].

In noisy channel applications, the truncation is not carried out as soon as the BP converges. Instead, a number of extra parity check bits are produced in order to cope with the channel noise. For a given output rate, the interesting performance measure is the block error rate as a function of the channel noise level. If the code were a true linear random code and the decoder were maximum a posteriori, then asymptotically it would be possible to achieve an overall coding rate (channel bits/ source bits) equal to the source entropy divided by the channel capacity.

4.2. LT-CLID direct approach. For simplicity we limit here our discussion to the compression of a block of k binary symbols $(x_1, \ldots, x_k)$ independently drawn from a binary source with bias p.

As an alternative to the pre-code of the raptor code, we could envision reliance on CLID to provide good performance in conjunction with an LT code used in much the same way as the LDPC code used in [**2, 8**]: m output bits are generated by the encoder of a $(k, \Omega(x))$ LT code, and d bits are generated by the CLID algorithm. The number of bits generated by the CLID algorithm is sufficient to result in successful decoding of the k input symbols under the assumption that every symbol is 1 with probability p. This method resembles the method of [**6**] with the difference

that the check nodes of the LDPC code have a degree distribution equal to $\Omega(x)$, and the variable nodes have a binomial degree distribution. If the number of edges in the graph is large, this distribution is close to the Poisson distribution with mean $\alpha m/k$, where α is the average degree of the distribution $\Omega(x)$. In particular, the fraction of variable nodes not covered by any edge in the graph is close to $\mathrm{e}^{-\alpha m/k}$. This means that the number of doped symbols in this construction is on average more than $k\mathrm{e}^{-\alpha m/k}$, and that the number of output symbols of the compressor is on average more than $k\left(\mathrm{e}^{-\alpha m/k}+m/k\right)$[1]. A necessary condition to achieve an output length within distance δ from the Shannon limit is

$$m/k \leq h(p)+\delta-\mathrm{e}^{-\alpha m/k} \tag{4.1}$$

On the other hand, in order for decoding to be successful the subgraph containing all $k(1-\mathrm{e}^{-\alpha m/k})$ covered variable nodes and the m output nodes must correspond to a linear code of rate not smaller than $h(p)$. Hence, a second necessary condition is given by

$$m/k \geq h(p)(1-\mathrm{e}^{-\alpha m/k}) \tag{4.2}$$

Letting $m/k=z\in[0,1]$, the condition imposed on α and δ for given $h(p)$ by (4.1) and (4.2) is that the root of the equation $h(p)(1-\mathrm{e}^{-\alpha z})=z$ must be smaller than the largest root of the equation $h(p)+\delta-\mathrm{e}^{-\alpha z}=z$.

A potential shortcoming of this method (which motivates the consideration of the alternative method in Section 4.3) is that sparse-graph codes with check degree distribution $\Omega(x)$ and variable degree Poisson distribution are known to perform poorly under iterative decoding on the erasure channel and other symmetric channels.

4.3. Two-stage approach. In this subsection we describe the principle of a different method of using LT-codes that is much better suited for the data compression application. The detailed description of the algorithm can be found in Section 4.4.

We calculate from the input symbols $(x_1,\ldots,x_k)$ a vector of *intermediate symbols* $(y_1,\ldots,y_k)$ through a linear invertible $k\times k$ matrix $\mathbf{G}$:

$$(y_1,\ldots,y_k)^T=\mathbf{G}^{-1}(x_1,\ldots,x_k)^T \tag{4.3}$$

Next, using the intermediate symbols we calculate m output symbols $(x_{k+1},\ldots,x_{k+m})$ according to a distribution $\Omega(x)$.[2] These m output symbols, together with the doped symbols obtained from the closed-loop iterative doping algorithm constitute the output of the compressor; hence, their total number should be as close as possible to $kh(p)$.

The Tanner graph on which the BP is run takes the form shown in Figure 2: a bipartite graph with k *intermediate nodes* (corresponding to the intermediate symbols) on one side and k input and m output nodes on the other side, corresponding to the k input and m output symbols, respectively. The variable nodes in that graph are indicated by circles in Figure 2. The invertible matrix $\mathbf{G}$ is chosen as

[1] All these numbers are only estimates due to the approximation of the binomial distribution with the Poisson distribution; the real numbers are very close to these estimates.

[2] The choice of $\Omega(x)$ is crucial for performance and its specific expression is given at the end of the section.

a random realization so that the degree distribution of the input nodes is $\Omega(x)$.[3] In this way, the degree distribution of both the input and output nodes (i.e., all nodes on the left in the graph of Figure 2) is equal to $\Omega(x)$. Notice that the resulting graph can be *interpreted* as the decoding graph of a $(k, \Omega(x))$ LT code with input symbols $(y_1, \ldots, y_k)$ and output symbols $(x_1, \ldots, x_{k+m})$, observed through a non-stationary BSC with transition probability p over the first k components and 0 over the second m components. The initial reliabilities (absolute value of the initial LLRs) of $(y_1, \ldots, y_k)$, $(x_1, \ldots, x_k)$ and $(x_{k+1}, \ldots, x_{k+m})$ are 0, $|\log((1-p)/p)|$ and $+\infty$, respectively.

Even though the matrix $\mathbf{G}$ is sparse, its inverse is generally dense. This has two consequences: the intermediate symbols behave like random coin flips, and the computation of (4.3) is, in principle, quadratic in k. In order to compute (4.3) in linear time, $\mathbf{G}$ should be such that the linear system $\mathbf{G}(y_1, \ldots, y_k)^T = (x_1, \ldots, x_k)^T$ can be solved by direct elimination of one unknown at a time. Equivalently, the BEC message-passing decoding algorithm applied to the Tanner graph of the parity equation $\mathbf{G}(y_1, \ldots, y_k)^T + (x_1, \ldots, x_k)^T = 0$ must terminate (i.e., it recovers all intermediate symbols $(y_1, \ldots, y_k)$).

4.4. Compressor and Decompressor. The basic scheme of Figure 1 is used where the syndrome computing block consists of the scheme presented in Section 4.3. The modeling algorithm is described in Section 6.

For simplicity we first describe the special case of the algorithm for binary data.

(1) After block sorting the original k-data vector, we obtain a block of symbols $(x_1, \ldots, x_k)$.
(2) An intermediate block $(y_1, y_2, \ldots, y_k)$ of symbols is calculated by (4.3) (in practice, this is accomplished via message-passing).
(3) A vector of m symbols $(x_{k+1}, \ldots, x_{k+m})$ is generated from $(y_1, \ldots, y_k)$ through encoding with an LT-code with parameters $(k, \Omega(x))$. Together with the doped bits generated as indicated below, $(x_{k+1}, \ldots, x_{k+m})$ forms the payload of the compressed data.
(4) A bipartite graph (Figure 2) is set up between the nodes corresponding to $(y_1, \ldots, y_k)$, and the nodes corresponding to $(x_1, \ldots, x_{k+m})$. The edges in this graph are defined as follows: for all i, there is an edge from x_i to all the bits among $(y_1, \ldots, y_k)$ of which x_i is the addition.
(5) The BP algorithm is applied to the graph created in the previous step. The objective of this BP algorithm is to decode the symbols $(y_1, \ldots, y_k)$ using the full knowledge of the symbols $(x_{k+1}, \ldots, x_{k+m})$ and the a priori marginal source probabilities $\{P(x_i = 1) = p_i : i = 1, \ldots, k\}$. For $i = 1, \ldots k$, the LLR of the bit x_i (unavailable to the decoder) is set to $\log((1-p_i)/p_i)$. The marginal probabilities p_i are either known (in a non-universal setting) or estimated from the source sequence itself by the source modeler illustrated in Section 6. The nodes corresponding to $(y_1, \ldots, y_k)$ have initially zero reliability.
(6) During the BP-algorithm, the CLID algorithm of [**6, 2**] is applied. In every ℓ-th round of the iteration the intermediate bit y_{i_ℓ} with the smallest reliability is marked, included in the payload, and its log-likelihood is set

[3]For example, $\mathbf{G}$ can be constructed row-by-row, such that every row is sampled from $\Omega(x)$ subject to the constraint that all the rows created so far are linearly independent.