Finite Length Analysis of LT Codes

Richard Karp
UC Berkeley
Berkeley, USA
karp@cs.berkeley.edu

Michael Luby
Digital Fountain, Inc.
Fremont, USA
luby@digitalfountain.com

Amin Shokrollahi
EPFL and Digital Fountain, Inc
Lausanne, Switzerland, and Fremont, USA
amin.shokrollahi@epfl.ch

Abstract — **We provide an efficient method for analyzing the error probability of the Belief Propagation (BP) Decoder applied to LT Codes.**

I. Introduction

Fountain Codes are a new class of codes originally designed for robust and scalable transmission of data over lossy computer networks. For a given vector of input symbols $(x_1, \ldots, x_k)$ a Fountain Code generates a potentially limitless stream of output symbols. Each output symbol is generated independently by sampling from a distribution $\mathcal{D}$ on $\mathbb{F}_2^k$, and adding the input symbols corresponding to the support of the sampled vector. The symbols generated by a Fountain Code are called output symbols.

Luby [1] was the first to invent an efficient class of Fountain Codes with fast encoding and decoding algorithms. The distribution in this subclass of Fountain Codes has the following shape: first, a distribution $\Omega = (\Omega_1, \ldots, \Omega_k)$ is chosen on the set $\{1, \ldots, k\}$. This distribution induces a distribution on $\mathbb{F}_2^k$ by assigning to the vector x the probability $\Omega_w / \binom{k}{w}$, where w is the weight of x. A Fountain Code with these properties is called an LT Code with parameters $(k, \Omega(x))$, where $\Omega(x) = \Omega_1 x + \Omega_2 x^2 + \cdots + \Omega_k x^k$ is the generating function of the distribution Ω.

LT Codes are decoded by applying the belief-propagation decoder to n of the output symbols of the encoder. The value of n depends on the output degree distribution of the LT Code and on the desired success probability of the decoder. In this paper we will give a method for calculating this success probability (or rather, the error probability) given the parameters of the LT Code, and the number of collected output symbols.

Our analysis is based on the observation that at each step the decoder is identified by a triple of natural numbers: the number u of undecoded input symbols, the number c of output symbols that are of reduced degree > 1 at that step, and the number r of output symbols that are of reduced degree one at that step. These parameters lead to a state generating function for the decoder, for which we will derive an explicit recursion.

II. Recursion for the State Generating Function

At each point in the decoding process of an LT Code the state of the decoder is described by three parameters: The number u of input symbols that are still unrecovered, the number r of output symbols of degree one at this point (the *output ripple*), and the number c of output symbols whose degree at this point is larger than 1 (the *cloud*). For fixed degree distribution, number k of input symbols, and number n of collected output symbols, the decoder will be in a state (c, r, u) with a certain probability $P_{c,r,u}$. Let $P_u(x, y) := \sum_{c,r,r \geq 1} P_{c,r,u} x^c y^{r-1}$ be the generating function of this probability distribution given that exactly u input symbols are undecoded. Then we have the following.

Theorem 1. *Suppose that the original code has k input symbols and that $n = k(1+\delta)$ output symbols have been collected for decoding. Further, denote by Ω_i, $i = 1, \ldots, D$, the probability that an output symbol is of degree i, where D is the maximum degree of an output symbol. Then we have for $u = k+1, k, \ldots, 1$*

$$P_{u-1}(x,y) = \frac{P_u\left(x(1-p_u) + yp_u, \frac{1}{u} + y\left(1 - \frac{1}{u}\right)\right)}{y} - \frac{P_u\left(x(1-p_u), \frac{1}{u}\right)}{y},$$

where for $u \leq k$,

$$p_u = \frac{\frac{1}{k}\frac{u-1}{k}\sum_{i=1}^{D} \Omega_i u(u-1) \frac{\left[\begin{array}{c} k-u \\ i-2 \end{array}\right]}{\left[\begin{array}{c} k \\ i \end{array}\right]}}{1 - \sum_{i=1}^{D} \Omega_i u \frac{\left[\begin{array}{c} k-u \\ i-1 \end{array}\right]}{\left[\begin{array}{c} k \\ i \end{array}\right]} - \sum_{i=1}^{D} \Omega_i \frac{\left[\begin{array}{c} k-u \\ i \end{array}\right]}{\left[\begin{array}{c} k \\ i \end{array}\right]}},$$

and $\left[\begin{array}{c} a \\ b \end{array}\right] := \binom{a}{b} b!$, and $p_{k+1} := \Omega_1$. Further, $P_{k+1}(x,y) := x^n$. A proof of this theorem can be found in the final version of this paper. The theorem is the basis for an efficient computation of the error probability of the BP Decoder applied to LT Codes. A detailed analysis shows that the running time of the algorithm to calculate the error probability of the LT Decoder has a bit-complexity of $O(n^3 \log^2(n) \log\log(n))$, where n is the number of collected output symbols of the LT Code.

Under the assumption that an output symbol can choose the same input symbol more than once as its neighbor, the recursion for the state generating function can also be used to derive recursions for the expected number of output symbols of degree one at each stage. If $R(u)$ denotes this expectation conditioned on u input symbols being undecoded, then we have

$$R(u-1) = p_u C(u) + \left(1 - \frac{1}{u}\right) R(u) - P_u(1,1) + P_u\left(1 - p_u, \frac{1}{u}\right).$$

Neglecting the "drift term" $1 - P_u(1,1) + P_u(1-p_u, 1/u)$ we obtain the exact expression $R(x) = x\Omega'(1-x) + x\ln(x)$, where $x = u/k$. This is exactly the same expression obtained from the tree analysis. This shows the interesting result that inexactness of the tree analysis comes solely from the drift term.

References

[1] M. Luby, "LT-codes," in *Proceedings of the ACM Symposium on Foundations of Computer Science (FOCS)*, 2002.