Design of LT Code Degree Distribution with Profiled Output Ripple Size

Kuo-Kuang Yen, Yen-Chin Liao, and Hsie-Chia Chang
Department of Electronics Engineering and Institute of Electronics,
National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
Email: hcchang@mail.nctu.edu.tw

***Abstract*—LT codes, a candidate for next-generation forward error correction (FEC) scheme, draws great attention due to its low encoding/decoding complexity and capacity-approaching properties. Especially for wireless applications in fast-changing environments, adaptive degree distribution is a possible solution to balance the trade-off between LT codes' efficiency and robustness. A fast mapping from attributes in the decoding process to degree distribution is required for realizing adaptive degree distribution. This study presents a reverse mapping from the expected ripple size to degree distributions. The basic idea is to minimize the error between a predetermined curve $\mu(\rho)$ and the expected ripple size $R(\Omega(x), \rho)$ of a target degree distribution $\Omega(x)$, where ρ is the number of decoded input symbols. By converting the code degree design into a pure optimization problem, it is possible to design LT codes with constraints from performance requirements to complexity issues. Applying sequential quadratic programming, a ripple-based distribution (RBD) is presented as a feasible design example. Simulation results show that, as compared to robust Soliton distribution (RSD) and previous studies, RBD is able to reduce the transmission overhead as well as the encoding and decoding complexity by at least 31.2% and 25.4%, respectively.** [1]

I. Introduction

LT code [1] is the first realization of fountain codes [2]. Low encoding and decoding complexity, as well as the capacity-approaching performance, LT codes has been drawing attention for being a candidate of application layer forward error correction (FEC) scheme. The performance of LT codes mainly depends on their degree distributions [3] [4] [5]. The state of the art robust Soliton distribution (RSD) [1] is designed for asymptotic optimality, and it performs well for long block lengths. *Ripple* is another important attribute in LT code decoding process. RSD is designed for a constant ripple size during decoding. However, in [6], it has been argued that a decreasing ripple is a more practical design considering efficiency and robustness. They suggest to design the degree distribution according to their deduced ripple evolution during the decoding process.

Inspired by this concept, we proposed to design degree distributions to fit some desired ripple evolution. Since the expected ripple size can be modeled as a function of degree distribution, it is possible to model the ripple evolution by any desired function $\mu(\rho)$ and infer the degree distribution by minimizing the error between the expected ripple size and $\mu(\rho)$. Degree distribution design can be converted to a pure optimization problem. Furthermore, LT codes are for sure candidates of next-generation communication FEC scheme. Especially in wireless applications where the channel conditions change so fast, a mapping from ripple evolution to degree distribution enables a quick adaptation of degree distribution. In this paper, a design example is introduced by minimizing the mean square error. Simulation results show the feasibility of this design concept. The rest of the paper is organized as follows. Section II introduces the background of LT codes. The proposed design with an example is presented in section III. Simulation results in Section IV compares the proposed design with RSD. Section V briefly concludes the work.

[1]This research is supported in part by Ministry of Science and Technology Taiwan, R.O.C. under grant NSC 101-2628-E-009 -013 -MY3 and MOST 104-2221-E-009-166-MY3, and in part by NCTU-MediaTek Research Center.

II. Background

A. LT Code Encoding and Decoding

An LT encoder is able to generate a potentially infinite sequence of output symbols from k input symbols. The degree of each output symbol is determined by the degree distribution $\Omega(x) = \sum_{d=1}^{k} \Omega_d x^d$, where d is the degree and Ω_d is the proportion of degree-d output symbols. During the encoding process, d input symbols are randomly selected, according to $\Omega(x)$, and bitwisely XORed to generate an output symbol.

Belief-Propagation (BP) algorithm is employed to decode input symbols after collecting $(1+\epsilon)k$ output symbols, where ϵ is the overhead. BP decoding iteratively performs these procedures:

- Identify all the degree-1 symbols and add their neighboring input symbols to a set termed *ripple*.
- Update the values of the input symbols in the ripple by their neighboring output symbols.
- Bitwisely XOR each output symbol with the connected input symbols in the ripple.
- Remove the edge connected to the ripple and subtract the output degree by 1.
- Repeat above procedures.

For each iteration, it requires at least one degree-1 output symbol for BP decoding to continue. When the ripple size decreases to zero, the decoding process terminates even though some input symbols remain undecoded. As a result, the key of designing degree distributions is to ensure the ripple size does not decrease to zero throughout the decoding process. Although a large ripple size reduce the probability of premature decoding termination, the probability of redundancy

978-1-4673-9604-2/15/.00 ©2015 IEEE

raises. That is, the input symbols associated to the degree-1 output symbols at each iteration are already in the ripple. As a result, the ripple size should be kept neither too large nor too small. Meanwhile, the average degree per output symbol must be small to achieve low encoding and decoding complexity.

B. Robust Soliton Distribution

The state of the art robust Soliton distribution (RSD) [1] can be represented by $\Omega^r(x) = \sum_{d=1}^{k} \Omega_d^r x^d$, where

$$\Omega_d^r = \frac{\zeta(d) + \Omega_d^i}{\sum_{d=1}^{k} \zeta(d) + \Omega_d^i},$$

$$\zeta(d) = \begin{cases} \frac{\eta}{kd} & \text{for } d = 1, 2, \cdots, \lfloor \frac{k}{\eta} \rfloor - 1 \\ \frac{\eta}{k} \ln(\frac{\eta}{\delta}) & \text{for } d = \lfloor \frac{k}{\eta} \rfloor \\ 0 & \text{else} \end{cases},$$

$$\Omega_d^i = \begin{cases} \frac{1}{k} & \text{for } d = 1 \\ \frac{1}{d(d-1)} & \text{for } d = 2, 3, \cdots, k \end{cases},$$

δ and c are design parameters and $\eta \equiv c \cdot \ln(\frac{k}{\delta})\sqrt{k}$.

III. Design of Degree Distributions

A. Optimization for Ripple-Based Distribution (RBD)

Although Luby's original work of LT code suggested a constant ripple size [1], Sorenson et al in [6] showed that decreasing ripple size could reduce necessary overhead. They modeled the ripple size evolution as a function of degree distribution, and proposed to design degree distribution by solving the least square solution to equations of the ripple size evolution.

Non-constant ripple size could be attractive from the perspectives of implementation and system resource allocation due to its flexibility. Following this concept, we propose a ripple-based distribution (RBD) with ripple size $\mu(\rho)$ as function of decoded input symbol number ρ. Assume there is a function $R(\mathbf{\Omega}, \rho)$ that models the expected ripple size, we can define a loss function by

$$L(\mathbf{\Omega}) = \frac{1}{k} \sum_{\rho=0}^{k-1} |R(\mathbf{\Omega}, \rho) - \mu(\rho)|^q + p(\mathbf{\Omega}) \tag{1}$$

where $p(\mathbf{\Omega})$ is a penalty function regularizing the solution that minimizes the loss function. As a result, degree distribution design can be converted to a pure optimization problem

$$\tilde{\mathbf{\Omega}} = arg \min_{\mathbf{\Omega}} L(\mathbf{\Omega}) \tag{2}$$

subjected to constraints on $\mathbf{\Omega}$. Further, q and $p(\mathbf{\Omega})$ can be chosen to meet requirements for different optimization schemes. For example, $q = 2$ and $p(\mathbf{\Omega}) = \lambda \sum_{d=1}^{k} |\Omega_d|^2$ ensure convexity of (1) required by many well developed optimization algorithms.

In addition, it is possible to twist the solutions by different combinations of q and $p(\mathbf{\Omega})$. For instance, if $p(\mathbf{\Omega}) = \lambda \sum_{d=1}^{k} |\Omega_k|$, with a carefully selection of λ a sparse solution can be found. That is, a large proportion of degrees are zero, leading to reduced encoding/decoding complexity. In this way, it is possible to design LT codes for different requirements and purposes by defining various loss function and target ripple size $\mu(\rho)$.

B. Reverse Mapping via Expected Output Ripple Evolution

In [7], Maatouk and Shokrollahi derived the expected ripple size as a function of frame length k, overhead ϵ, the number of decoded input symbols ρ, and the degree distribution $\Omega(x)$ as follows

$$\begin{aligned} R(\Omega(x), \rho) &= (1+\epsilon)(k-\rho)\left(\Omega'(\tfrac{\rho}{k}) + \tfrac{1}{1+\epsilon} \ln \tfrac{k-\rho}{k}\right) + C \\ &= (1+\epsilon)\left(\sum_{d=1}^{k} \Omega_d f(\rho, d) + \tfrac{k-\rho}{1+\epsilon} \ln \tfrac{k-\rho}{k}\right) + C, \end{aligned} \tag{3}$$

where the overhead $\epsilon = \frac{r-k}{k}$, r is the number of received output symbols, $f(\rho, d) = (k-\rho)d(\frac{\rho}{k})^{d-1}$ and C is a constant.

It is worth noting that the term ripple is defined as *the set of output symbols of degree one* in [7]. To distinguish it from the original definition of ripple in Luby's work [1], the set of degree-one output symbols is rephrased by *output ripple* hereafter. Although defined differently, there is a direct mapping from output ripple to ripple. The concept of degree distribution design by a reverse mapping is till applicable. Employing (3) as the ripple function $R(\mathbf{\Omega}, \rho)$ in the loss function (1), we can construct degree distribution $\Omega(x) = \sum_{d=1}^{k} \Omega_d x^d$ with this expected output ripple size approximating any given $\mu(\rho)$. Moreover, to preserve convexity and simplicity, q and $p(\mathbf{\Omega})$ are set to 2 and 0, respectively. Hence the loss function becomes

$$L(\Omega(x)) = \frac{1}{k} \sum_{\rho=0}^{k-1} (R(\Omega(x), \rho) - \mu(\rho))^2, \tag{4}$$

subject to

$$\Omega_d \geq 0 \text{ for } d = 1, 2, \cdots, k \tag{5}$$

$$\sum_{d=1}^{k} \Omega_d = 1 \tag{6}$$

$$R(\Omega(x), \rho) \geq \gamma \text{ for } \rho = 0, 1, \cdots, k-1, \tag{7}$$

where γ is a non-negative design parameter.

Constraints (5) and (6) ensure that all of the elements in Ω must be non-negative and they sum up to one, which are the basic properties of a probability distribution. In addition, minimizing the mean squared error does not guarantee that the expected ripple size of RBD never decrease to zero within the range of $0 \leq \rho \leq k-1$. Further, constraint (7) sets a tolerable minimum γ to the expected ripple size of RBD.

C. Solving the Solution

In (4), there are only k variables Ω_1, Ω_2, $\cdots$, Ω_k. Given k and ϵ, using (3), the expression of (4) can be simplified as

$$\begin{aligned} L(\Omega(x)) =& \frac{1}{k}\sum_{\rho=0}^{k-1}\left((1+\epsilon)\left(\sum_{d=1}^{k}\Omega_d f(\rho,d)+\frac{k-\rho}{1+\epsilon}\ln\frac{k-\rho}{k}\right)\right. \\ & \left.+C-\mu(\rho)\right)^2 \\ =& \frac{1}{k}\sum_{\rho=0}^{k-1}\left(\sum_{d=1}^{k}C_{\rho,d}\Omega_d+C'_\rho\right)^2 \end{aligned},$$

with constants $C_{\rho,d}=(1+\epsilon)\cdot f(\rho,d)$ and $C'_\rho=(k-\rho)\ln\frac{k-\rho}{k}+C-\mu(\rho)$.

Sequential quadratic programming (SQP) algorithm can be employed to search for the solution of the minimization problem (4). Similar to Newton-Raphson method, SQP iteratively derives the next minimizing point based on current results. Let us define $\mathbf{\Omega}=[\Omega_1\ \Omega_2\ \cdots\ \Omega_k]$, our minimization problem can be rewritten as

$$\min_{\mathbf{\Omega}} L(\mathbf{\Omega})$$

subject to

$$\begin{aligned} G(\mathbf{\Omega}) =& 0, \\ \mathbf{H}(\mathbf{\Omega}) \geq& \mathbf{0}, \end{aligned}$$

where $G(\mathbf{\Omega})=\sum_{d=1}^{k}\Omega_d-1$ corresponds to constraint (6), $\mathbf{0}$ is a $2k$-by-1 all zero vector and

$$\mathbf{H}(\mathbf{\Omega})=\begin{bmatrix}\Omega_1\\ \Omega_2\\ \vdots\\ \Omega_k\\ R(\mathbf{\Omega},0)-\gamma\\ R(\mathbf{\Omega},1)-\gamma\\ \vdots\\ R(\mathbf{\Omega},k-1)-\gamma\end{bmatrix}.$$

corresponds to constraints (5) and (7), which sum up to $2k$ inequalities. The corresponding Lagrangian function can be represented by

$$\mathcal{L}(\mathbf{\Omega},A,\mathbf{B})=L(\mathbf{\Omega})-aG(\mathbf{\Omega})-\mathbf{b}\mathbf{H}(\mathbf{\Omega}).$$

with Lagrange multipliers a and $\mathbf{b}=[b_1\ b_2\ \cdots\ b_{2k}]$. The Karush-Kuhn-Tucker (KKT) conditions for the solution are

$$\begin{aligned} \nabla_{\mathbf{\Omega}}\mathcal{L}(\mathbf{\Omega},a,\mathbf{b}) =& 0, \\ G(\mathbf{\Omega}) =& 0, \\ \mathbf{H}(\mathbf{\Omega}) \geq& \mathbf{0}, \\ \mathbf{b}^T \geq& \mathbf{0}, \\ \mathbf{b}\mathbf{H}(\mathbf{\Omega}) =& 0, \end{aligned}$$

and we can derive the quadratic programming subproblem

$$\begin{aligned} \min_{\Delta\mathbf{\Omega}^{(\ell)}} & \nabla L(\mathbf{\Omega}^{(\ell)})^T\Delta\mathbf{\Omega}^{(\ell)} \\ & +\frac{1}{2}(\Delta\mathbf{\Omega}^{(\ell)})^T\nabla^2_{\mathbf{\Omega}^{(\ell)}}\mathcal{L}(\mathbf{\Omega}^{(\ell)},a^{(\ell)},\mathbf{b}^{(\ell)})\Delta\mathbf{\Omega}^{(\ell)} \end{aligned}$$

subject to

$$\begin{aligned} G(\mathbf{\Omega}^{(\ell)})+\nabla G(\mathbf{\Omega}^{(\ell)})^T\Delta\mathbf{\Omega}^{(\ell)} =& 0, \\ \mathbf{H}(\mathbf{\Omega}^{(\ell)})+\nabla\mathbf{H}(\mathbf{\Omega}^{(\ell)})^T\Delta\mathbf{\Omega}^{(\ell)} \geq& 0, \end{aligned}$$

where ℓ is the iteration number. By solving this subproblem, we obtain the steps $\Delta\mathbf{\Omega}^{(\ell)}$, $\Delta a^{(\ell)}$ and $\Delta\mathbf{b}^{(\ell)}$. This algorithm proceeds with

$$\begin{aligned} \mathbf{\Omega}^{(i+1)} =& \mathbf{\Omega}^{(\ell)}+\Delta\mathbf{\Omega}^{(\ell)} \\ a^{(i+1)} =& a^{(\ell)}+\Delta a^{(\ell)} \\ \mathbf{b}^{(i+1)} =& \mathbf{b}^{(\ell)}+\Delta\mathbf{b}^{(\ell)}. \end{aligned}$$

and it ends at the j-th iteration when $L(\mathbf{\Omega}^{(j+1)})\geq L(\mathbf{\Omega}^{(j)})$.

D. Uniquity

Let us define two distinct points $\mathbf{Y}=[Y_1,Y_2,\cdots,Y_k]$ and $\mathbf{Z}=[Z_1,Z_2,\cdots,Z_k]$ in a k-dimensional space of degree distribution $\mathbf{\Omega}$. For all θ in the range $0<\theta<1$,

$$\begin{aligned} & \theta L(\mathbf{Y})-(1-\theta)L(\mathbf{Z})-L(\theta\mathbf{Y}-(1-\theta)\mathbf{Z}) \\ =& \frac{\theta(1-\theta)}{k}\sum_{\rho=0}^{k-1}\left(\sum_{d=1}^{k}C_{\rho,d}(Y_d-Z_d)\right)^2\geq 0 \end{aligned}$$

The equality holds when

$$\sum_{\rho=0}^{k-1}\left(\sum_{d=1}^{k}C_{\rho,d}(Y_d-Z_d)\right)^2=\sum_{\rho=0}^{k-1}\left(\mathbf{C}_\rho(\mathbf{Y}-\mathbf{Z})^T\right)^2=0,$$

for $\mathbf{C}_\rho=[C_{\rho,1},C_{\rho,1},\cdots,C_{\rho,k}]$. It can be verified that $[\mathbf{C}_1^T\mathbf{C}_2^T\cdots\mathbf{C}_k^T]$ forms a full rank matrix. Thus the above equality holds only if $\mathbf{Y}=\mathbf{Z}$, which contradicts the assumption that $\mathbf{Y}$ and $\mathbf{Z}$ are distinct points. As a result,

$$\theta L(\mathbf{Y})-(1-\theta)L(\mathbf{Z})-L(\theta\mathbf{Y}-(1-\theta)\mathbf{Z})>0.$$

The loss function in (3) is strict convex and the solution is unique.

IV. Design example

In [1], the ripple size evolution during BP decoding process is modeled as a random walk of length k. It has been proven that such a random walk process deviates from its mean by more than $\ln(\frac{k}{\delta})\sqrt{k}$ is at most δ. Robust Soliton distribution (RSD) is asymptotically designed with a constant ripple size around $c\cdot\ln(\frac{k}{\delta})\sqrt{k}$ throughout the decoding process. This guarantees the ripple size decreasing to zero has a probability no greater than δ. For finite code length where k is small, a smaller value of $\ln(\frac{k}{\delta})\sqrt{k}$ reveals a higher probability of decoding failure because the ripple size goes to zero more easily. Intuitively, larger ripple size at the early stage of decoding is desirable. However, large ripple implies higher proportion of degree-1 input symbols and potentially higher redundancy, especially at the late decoding stage. Thus for a constant-rippled distribution design, the value $c\cdot\ln(\frac{k}{\delta})\sqrt{k}$ should be neither too small nor too large, and the design parameters of RSD should be carefully chosen to balance the trade-off between overhead and robustness.

In fact, the length of the random walk process decreases to $k-\rho$ when ρ input symbols are decoded. A mean of

$\ln(\frac{k-\rho}{\delta})\sqrt{k-\rho}$ is sufficient to guarantee a random walk of length $k-\rho$ deviates from its mean with a probability no greater than δ. This coincides with the argument in [6] that decreasing ripple size during the decoding process is more practical considering overhead and robustness. Thus we can design the degree distribution to fit a target ripple function $\mu(\rho) \propto \ln(\frac{k-\rho}{\delta})\sqrt{k-\rho}$. Moreover, we add two design parameters α and β to increase the flexibility in adjusting the curve $\mu(\rho)$. That is,

$$\mu(\rho) := \alpha \ln(\frac{k-\rho}{\delta})\sqrt{k-\rho} + \beta. \quad (8)$$

The scaling factor α, the slop of $\mu(\rho)$, determines how quickly the ripple decrease with ρ. The offset β ensures a minimum ripple size throughout the decoding process.

The design parameters α and β may be derived empirically incorporated with γ in constraint (7). This example shows the feasibility of design an LT code with desired ripple function. Simulation results in next section will show, with carefully chosen design parameters, the proposed ripple-based distribution (RBD) performs well in terms of complexity and error rate. Notice that (8) is not the only format of $\mu(\rho)$. One can design any curve $\mu(\rho)$ to meet their own requirements.

V. Simulation Results

Ripple-based distribution (RBD) designed by (8) is compared with robust Soliton distribution (RSD) for k = 512, 1024, 3000 and 6000, respectively. Output symbols are transmitted through lossless channels. By fixing the parameter δ to 0.01, the design parameter c for RSD is determined empirically through exhaustively searching from 0 to 1 with a resolution of 0.005 for the value conducting to the least average overhead. $c = 0.015$ is used for $k = 512$, $k = 1024$, and $k = 3000$, and $c = 0.02$ is for $k = 6000$. The corresponding degree distributions are showns in Fig.1. Parameters for constructing RBDs are listed in Table TABLE I and the corresponding degree distribution are shown in TABLE II to TABLE V.

TABLE I. Parameters for constructing RBD

k	ϵ	δ	α	β	γ
512	0.12	1.0	0.14	5.81	1.5
1024	0.1	1.0	0.15	8.7	2
3000	0.07	1.0	0.11	22.5	2.6
6000	0.05	1.0	0.06	37.01	3.1

TABLE II. Degree distribution of RBD for $k = 512$

Degree	Proportion
1	0.0446
2	0.4575
3	0.1525
4	0.0882
5	0.0328
6	0.0172
7	0.0751
13	0.0369
14	0.0286
33	0.0395
113	0.027

Similar to RSD, RBD preserves three properties of a good degree distribution [8] [9] [10]. First, the degree-2 has the greatest proportion. Second, the degree-1 proportion is relatively small. Third, there is a spike at high degree that keeps the process alive at the end of BP decoding. However, RSD and RBD differ in the way of how the output ripple size evolves during the decoding process. Fig.2 to Fig.5 show the expected output ripple size versus the proportion of the decoded input symbols ρ/k , and the expected output ripple

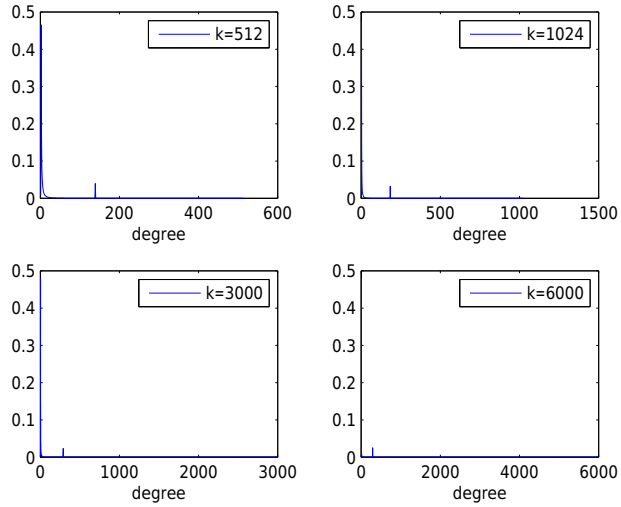

Fig. 1. Robust Soliton distribution for $k = 512$, $k = 1024$, $k = 3000$, and $k = 6000$. The degree-2 proportions are 0.4654,0.4718, 0.4799, and 0.4789, respectively. The high-degree spikes are at 139, 184, 289, and 291, respectively, and the corresponding proportions are 0.0393, 0.0321, 0.0229, and 0.025

TABLE III. Degree distribution of RBD for $k = 1024$

Degree	Proportion
1	0.0377
2	0.4637
3	0.1579
4	0.0698
5	0.0756
7	0.0105
8	0.0667
9	0.0036
13	0.0209
17	0.0366
36	0.03
86	0.0028
100	0.0166
301	0.0078

TABLE IV. Degree distribution of RBD for $k = 3000$

Degree	Proportion
1	0.022
2	0.4736
3	0.1595
4	0.0774
5	0.0638
7	0.0387
8	0.0459
14	0.0369
15	0.0192
32	0.0216
33	0.0124
87	0.0085
100	0.0128
397	0.0073

TABLE V. DEGREE DISTRIBUTION OF RBD FOR $k = 6000$

Degree	Proportion
1	0.0123
2	0.4806
3	0.1581
4	0.0922
5	0.0065
6	0.091
8	0.0277
12	0.0137
13	0.0484
29	0.0299
30	0.0074
87	0.0202
101	0.0016
483	0.0094

versus the number of undecoded input symbols $k-\rho$ for code lengths $k = 512$, $k = 1024$, $k = 3000$, $k = 6000$, respectively. The probability of BP decoding termination when ρ input symbols are decoded is denoted by $\Gamma(\rho)$, and the cumulative probability of BP decoding termination is $\prod(\rho) = \sum_{j=0}^{\rho} \Gamma(\rho)$. The corresponding cumulative probability$\prod(\rho)$ of BP decoding termination is depicted by averaging one million encoding/decoding. Apparently, BP decoding process is more likely to terminate where the expected output ripple size is relatively small. This is consistent with the results shown in [7]. On average, the expected output ripple size of RBD is greater than that of RSD, the reduced probability of premature decoding termination can be inferred. In particular, the zoom-in at $\rho - k$ close to 0 (on the right of each figure) show that RBD has smaller spikes in this region due to the smaller high-degree proportions. As a result, RBD relaxes a potentially troublesome implementation issue of boosting computation when $\rho - k$ is close to 0. Moreover, there remains noticeable spikes at high degree in RBD to prevent the expected output ripple size from decreasing to zero when most input symbols are decoded. This is observable from Fig.2 to Fig.5. For RBD, the cumulative probability $\prod(\rho)$ increases slightly when the number $k-\rho$ of undecoded input symbols is close to zero.

Assuming that the receiver is allowed to collect as many output symbols as needed to decode all the k input symbols. TABLE VII records the statistics of 10000 successful BP decoding: the average number of received symbols $(1+\bar{\epsilon})k$, the average symbol operations in encoding (normalized to k) and the average symbol operations in decoding (normalized to k). Compared to RSD, RBD reduces the overhead by at least $\frac{1.06-1.05}{1.05} = 1\%$, and improves the encoding and decoding complexity by at least $\frac{14.3-10.9}{10.9} = 31.2\%$ and $\frac{16.3-13}{13} = 24.5\%$, respectively. The percentage of improvement by RBD becomes more significant as k decreases.

Fig.6 shows the frame error rate of RSD and RBD versus overhead based on one million encoding/decoding. It is observed RBD outperforms RBD as code length k is small. However, as k grows, the FER decreases slowly in RBD in the high overhead region because the average degree per output symbol of RBD is lower than that of RSD. This can be explained by the smaller proportion of the high-degree spikes in RBD and the lower average degree shown in TABLE VI. Given the same amount of output symbols, the probability of all k input symbol being fully connected becomes lower when using RBD instead of RSD. Those unconnected input symbols result in decoding failure and increase the frame error

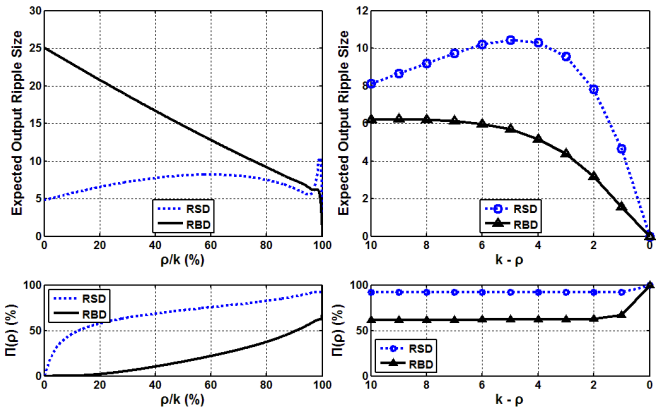

Fig. 2. Expected ripple size versus ratio ρ/k for $k = 512$ and overhead $\epsilon = 0.1$ and the corresponding occurrence probability $\Gamma(\rho)$ of BP decoding termination is depicted.

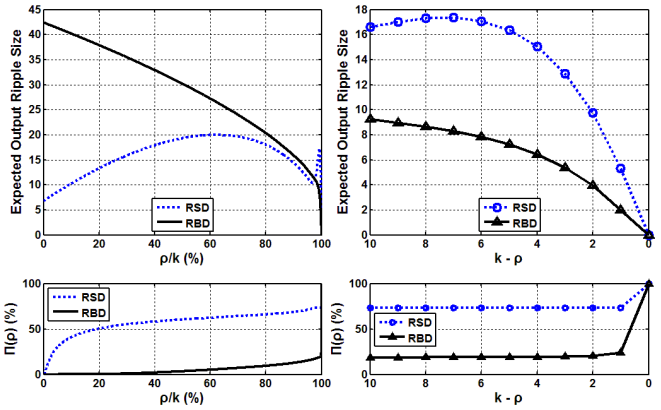

Fig. 3. Expected ripple size versus ratio ρ/k for $k = 1024$ and overhead $\epsilon = 0.1$ and the corresponding occurrence probability $\Gamma(\rho)$ of BP decoding termination is depicted.

rate of RBD. Most applications are targeted to tranmission with low-to-medium code lengths and low overheads (where the frame error rate of RBD is lower than that of RSD), though, RSD outperforms RBD in high-overhead regions for long code lengths. To remedy the FER degradation in the high-overhead regions, additional constraints could be added to ensure sufficient links between the input symbols and output symbols. For example, one additional constraint can be added to $\mathbf{H}(\mathbf{\Omega})$:

$$\sum_{d=1}^{k} d\Omega_d > e_0.$$

TABLE VI. EXPECTED OUTPUT DEGREE

Method	$k = 512$	$k = 1024$	$k = 3000$	$k = 6000$
RSD	12.672	13.876	15.790	17.087
RBD	7.797	8.958	9.537	10.959

VI. CONCLUSION

In this study, designing LT codes' degree distribution by reverse mapping from a desired ripple function $\mu(\rho)$ is presented. Given a ripple evolution function $R(\mathbf{\Omega}, \rho)$, the

TABLE VII. COMPLEXITY COMPARISON

Code length	$k = 512$		$k = 1024$		$k = 3000$		$k = 6000$	
Mothod	RSD	RBD	RSD	RBD	RSD	RBD	RSD	RBD
Average received symbol number $1 + \bar{\epsilon}$	1.20	1.12	1.13	1.09	1.09	1.06	1.06	1.05
Average symbol operations in encoding	11.5	7.8	12.4	8.9	13.6	9.5	14.3	10.9
Average symbol operations in decoding	13.9	9.8	15.1	11.1	15.9	11.6	16.3	13.0

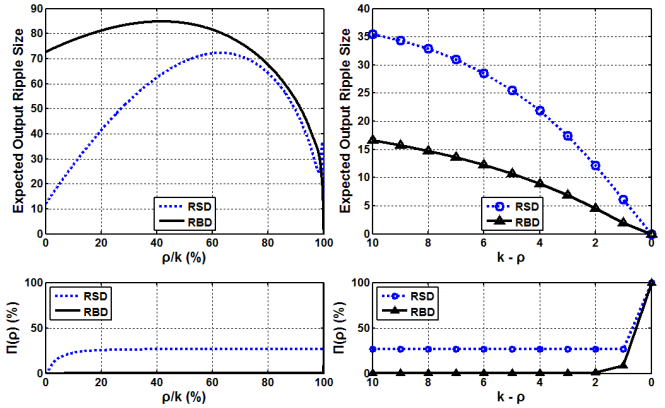

Fig. 4. Expected ripple size versus ratio ρ/k for $k = 3000$ and overhead $\epsilon = 0.1$ and the corresponding occurrence probability $\Gamma(\rho)$ of BP decoding termination is depicted.

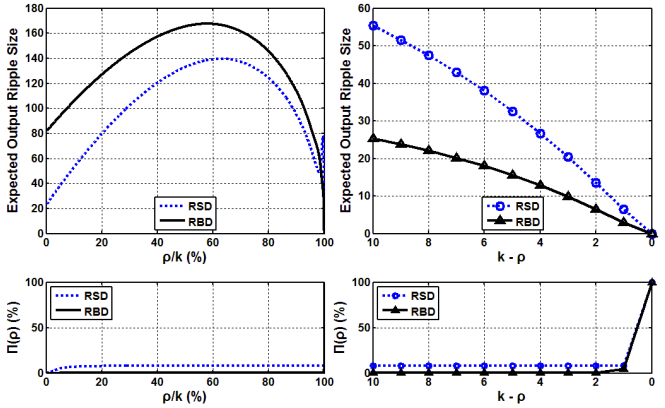

Fig. 5. Expected ripple size versus ratio ρ/k for $k = 6000$ and overhead $\epsilon = 0.1$ and the corresponding occurrence probability $\Gamma(\rho)$ of BP decoding termination is depicted.

corresponding degree $\mathbf{\Omega}$ can be inferred by minimizing the loss between $R(\mathbf{\Omega}, \rho)$ and $\mu(\rho)$. Degree distribution design is converted to a pure optimization problem. Various definition of the loss functions and optimization approaches conduct to different results, providing flexibility and possibilities of degree distribution designed with different constraints. The desired ripple function $\mu(\rho)$ can be modeled to meet specific desired properties as well. In our design example, a decreasing function $\mu(\rho)$ is introduced, and the simulation results show that the proposed design improves the efficiency and complexity, especially for short code lengths. Numerical results show the encoding and decoding complexity can be reduced to at least 31.2% and 25.4%, respectively.

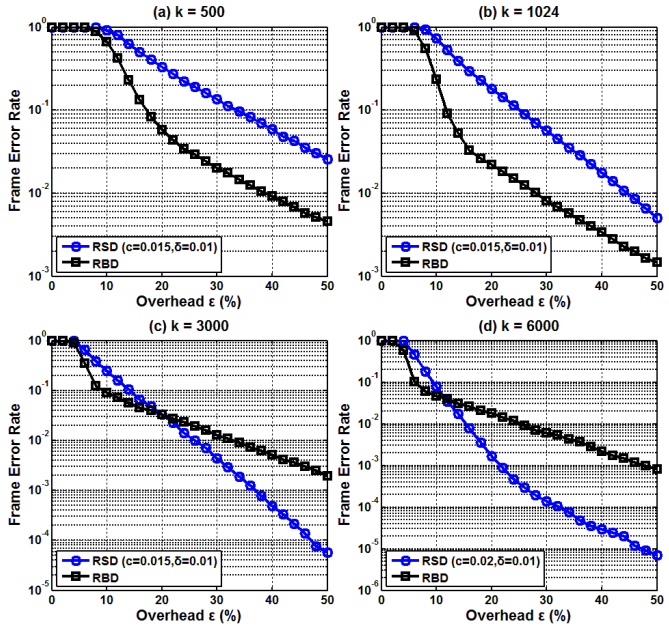

Fig. 6. Frame error rate versus overhead for $k = 512$, $k = 1024$, $k = 3000$, and $k = 6000$

REFERENCES

[1] M. Luby, "LT codes," in *Proceedings of The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271 – 280.

[2] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proceedings of the ACM SIGCOMM*, 1998, pp. 56 – 67.

[3] C.-M. Chen, Y.-P. Chen, T.-C. Shen, and J. K. Zao, "On the optimization of degree distributions in LT code with covariance matrix adaptation evolution strategy," in *IEEE Congress on Evolutionary Computation, CEC*, 2010, pp. 1–8.

[4] S. Jafarizadeh and A. Jamalipour, "An exact solution to degree distribution optimization in lt codes," in *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2014, pp. 764 – 768.

[5] H. Zeineddine, M. M. Mansour, and R. Puri, "Construction and hardware-efficient decoding of raptor codes," *IEEE Trans. Signal Process.*, vol. 59, pp. 2943 – 2960, 2011.

[6] J. H. Sorensen, P. Popovski, and J. Ostergaard, "Design and analysis of LT codes with decreasing ripple size," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3191–3197, 2012.

[7] G. Maatouk and A. Shokrollahi, "Analysis of the second moment of the LT decoder," *IEEE Trans. Inf. Theofy*, vol. 58, no. 5, pp. 2558–2569, 2012.

[8] D. J. C. MacKay, "Fountain codes," in *IEE Proceedings Communications*, vol. 152, Dec. 2005, pp. 1062 – 1068.

[9] A. Liau, S. Yousefi, and I.-M. Kim, "Binary Soliton-like rateless coding for the Y-network," *IEEE Transactions on Communications*, vol. 59, no. 12, pp. 3217–3222, 2011.

[10] K.-K. Yen, Y.-C. Liao, C.-L. Chen, and H.-C. Chang, "Modified robust soliton distribution (MRSD) with improved ripple size for LT codes," *IEEE Communications Letters*, vol. 17, no. 5, pp. 976–979, 2013.