IET Communications

Research Article

Design of optimised multiple partial recovery LT codes

ISSN 1751-8628
Received on 3rd July 2015
Revised on 25th December 2015
Accepted on 15th February 2016
doi: 10.1049/iet-com.2015.0539
www.ietdl.org

Jianxin Liao[1,2] ✉, *Lei Zhang*[1,2], *Tonghong Li*[3], *Jingyu Wang*[1,2], *Qi Qi*[1,2]

[1]*State Key Laboratory of Network and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, People's Republic of China*
[2]*EBUPT Information Technology Co., Ltd, Beijing 100191, People's Republic of China*
[3]*Department of Computer Science, Technical University of Madrid, Madrid, Spain*
✉ *E-mail: jianxin.liao@sohu.com*

Abstract: Existing rateless codes have a very low intermediate symbol recovery rate. Therefore, a new analysis method named iterative and small degree first (I-SDF) is presented for the design of optimised partial recovery Luby transform codes (PR-LTC) in this study. On the basis of I-SDF, the required number of encoded symbols with degree d in each decoding step is calculated by an iterative optimisation algorithm. Under the proposed design, $R(R < k)$ input symbols can be recovered from as few encoded symbols as possible in PR-LTC with message length k. Furthermore, multiple PR-LTC (M-PR-LTC) is proposed to recover several partial recovery point (PRPs) efficiently. The analysis process is divided into multiple stages, and the required number of encoded symbols with degree d in each decoding step is calculated by a cross-stage iterative optimisation algorithm. In addition, the interaction of each stage is adjusted by introducing a weight for each PRP. The PR-LTC and M-PR-LTC are evaluated and compared with the existing schemes. The simulation results demonstrate that PR-LTC and M-PR-LTC outperform other existing schemes in terms of average overhead, average degree of encoded symbols, memory usage, bit error rate and energy consumption.

1 Introduction

Rateless codes proposed in [1] provide reliable and efficient information delivery without any knowledge of channel state information at transmitter. Luby transform (LT) codes were the first practical rateless codes [2], in which successful decoding is possible when $m = (1 + \varepsilon)k$ encoded symbols have been received, where ε approaches zero for increasing message length k. Rateless codes such as LT codes and Raptor codes [3] are capacity-achieving, however in intermediate ranges, i.e. $m < (1 + \varepsilon)k$, input symbols are barely recovered because most of the received encoded symbols are buffered for the later decoding. Therefore, these rateless codes have a low intermediate symbol recovery rate (ISRR). However, in some applications such as multimedia content delivery, partial recovery of the input symbols is still useful, which thus motivates researchers to design rateless codes with high ISRR.

Several works have tried to improve the intermediate performance of rateless codes by using feedback. In [4], the encoder gradually increases the degree of encoded symbols to maximise the instantaneous recovery probability of each arriving encoded symbol according to the number of recovered input symbols. In [5], shifted LT (SLT) codes based on feedback are designed, where the degree distribution is shifted to decrease the overhead and increase the ISRR. Lei *et al.* [6] extends the work in [5], where degree values are diversified. Contrary to [5, 6], the design proposed in [7] considers any feedback opportunity. In [8], iLTC-DRS-F (improved LT codes with decreasing ripple size and feedback) based on the generalised degree distribution algorithm are presented, where the accurate ripple size is evolved. However, they only increase the ISRR at the end of transmission since the degree distribution is dynamically adjusted according to the number of recovered input symbols and only a few input symbols are recovered at the beginning.

Growth codes are proposed in [9], which are originally designed for wireless sensor networks to maximise the data persistence by increasing the ISRR. The work in [10] is an extension of that presented in [9], where the asymptotic performance of growth codes is investigated by using the Wormald method. In growth codes, the perfect source setting, where the receiver receives encoded symbols exactly fitting a certain desired degree distribution, is not practical in unreliable transmission channels. A new design of rateless codes, which divides the intermediate range into three regions ([0, 1/2], (1/2, 2/3] and (2/3, 1]) for achieving the upper bound on the ISRR is studied in [11]. The optimum degree distributions are obtained by a random hyper-graphs analysis method. The codes designed in [11] are asymptotically optimal and may not be employed when k is finite. In [12], three overheads (0.5, 0.75, 1) are chosen and a multi-objective genetic algorithm is employed to design a near optimal degree distribution. However, the overhead is larger than 1 in some applications, hence it is not always feasible. The performance of LT codes has been improved in [13] to achieve the intermediate decoding performance, by following the evolutionary approaches. However, the parameters used in the optimisation process should be tuned for each specific scenario.

The contribution of this work is two-fold. First, we propose a novel analysis method named iterative and small degree first (I-SDF) to analyse the belief propagation BP) decoding process accurately. Second, we propose the design of an optimised partial recovery LT codes (PR-LTC) by using I-SDF. In I-SDF, only one input symbol is released in each step, and encoded symbols are processed in an ascending order with respect to their degrees since encoded symbols with small degrees have a higher release probability. The release probability of each encoded symbol and the required number of encoded symbols in each step are analysed by employing an iterative optimisation algorithm. In our optimised PR-LTC, number $R(R \leq k)$ is assigned a priori. R input symbols can be recovered when $R(1 + \varepsilon)$ encoded symbols are received. The degree distribution is calculated by normalising the required number of encoded symbols, which is similar to [14]. Furthermore, an optimised multi PR-LTC (M-PR-LTC) is proposed to obtain high ISRRs in several pre-specified partial recovery points (PRPs). To adjust the interaction of each PRP, a weighted M-PR-LTC is proposed. By using an optimum degree distribution, our codes perform better in terms of average overhead, average degree of encoded symbols, memory usage, BER (bit error rate) and energy consumption. This is verified through numerical evaluations and comparisons with state-of-the-art solutions.

The rest of the paper is organised as follows. In Section 2, the review of LT codes and analysis methods of BP decoding is given. In Section 3, optimised PR-LTC is presented after some preliminaries are demonstrated. In Section 4, optimised M-PR-LTC is presented. The performance of M-PR-LTC is analysed in Section 5. In Section 6, our experimental design is outlined, and the efficiency of PR-LTC and M-PR-LTC is illustrated by our experiment results. Finally, our work is summarised.

2 Review of LT codes and analysis methods of BP decoding

2.1 LT codes

Suppose that bulk data comprising of k input symbols need to be transmitted from transmitter to receiver. Let $\Omega(1), \ldots, \Omega(k)$ be the degree distribution, such that $\Omega(d)$ denotes the probability that degree d is chosen. An encoded symbol is generated as follows:

i. a degree (d) is chosen at random according to the distribution $\Omega(1), \ldots, \Omega(k)$;
ii. d distinct input symbols are chosen uniformly at random from k input symbols;
iii. an encoded symbol is generated by performing bitwise XOR operations on the selected d input symbols.

Any selected d input symbol is called *neighbour* of this encoded symbol. The BP decoding is widely used for rateless codes, which is performed through the reverse bitwise XOR operations. In the following, we describe two analysis methods of BP decoding: the And–Or tree and evolution of the ripple size (ERS).

2.2 And–or tree

An And–Or tree T_l is defined as follows. Let T_l be a tree of depth 2 l. The root of the tree is at depth 0, its children are at depth 1, and so forth. Each node at 0, 2, 4, …, $2l-2$ is called an OR-node, and each node at depth 1, 3, 5, …, $2l-1$ is called an AND-node. Suppose that each OR-node independently chooses to have i children with probability δ_i, where $\sum_i \delta_i = 1$. Similarly, each AND-node chooses to have i children with probability β_i, where $\sum_i \beta_i = 1$. OR-nodes with no children are assumed to have a value 0, whereas AND-nodes with no children are assumed to have a value 1. Let v_l denote the probability that an input symbol is not recovered after l decoding iterations, thus we have $v_l = \exp(-\alpha\omega(1-v_{l-1}))$, in which $v_0 = 1$, $\omega(y) = \beta'(y)/\beta'(1)$, and $\alpha = \delta'(1)$ [15]. Let m_z denote the number of released input symbols when m_r encoded symbols have been received. Two quantities $z = m_z/k$ and $r = m_r/k$ are defined to analyse the asymptotic performance of rateless codes. In [12], overhead is divided into three regions of $r \in [0, 0.5)$, $r \in [0.5, 0.75)$ and $r \in (0.75, 1]$. Let $z_{0.5,\Omega}$, $z_{0.75,\Omega}$ and $z_{1,\Omega}$ denote the value of z at three selected points representing three objective functions that aim to concurrently maximise and realise high ISRRs. Let V_r denote the probability that an input symbol is not recovered after r decoding iterations, hence $z_r = 1 - V_r$. Therefore, an optimal degree distribution is obtained by employing a multi-objective genetic algorithm. However, overhead r is larger than 1 in some applications, hence it is not always feasible.

2.3 Evolution of the ripple size

In ERS analysis method, initially all input symbols are unrecovered. All encoded symbols with degree one are first released to recover their unique neighbours. All recovered input symbols that have not been processed are called *ripple*. Symbols in the ripple are processed one by one until all input symbols are recovered. The processing of an input symbol in ripple is as follows:

i. it is removed from the ripple;
ii. it is removed as a neighbour from all encoded symbols that have it as a neighbour;
iii. for each encoded symbol with exactly one remaining neighbour, its remaining neighbour is released; this operation is called a symbol *release*; for each encoded symbol with zero remaining neighbour, its neighbours have been released before; this operation is called a symbol *sink*;
iv. new released input symbols previously unrecovered are added into the ripple.

Decoding is successful when all input symbols have been recovered. If the ripple size equals zero before the successful decoding, decoding fails. This hints that the well performing LT codes should ensure a high ripple size during the decoding process. However, when an encoded symbol is released, there is a risk that the neighbouring input symbol is already in the ripple, in which case the encoded symbol is redundant. Hence, to minimise the risk of redundancy, the ripple size should be kept low. This trade-off is the main argument for the design goal in ERS analysis method [2, 8, 14]. Luby sets forth a design goal where the constant ripple size should be above one at a reasonable level [2]. The work presented in [8, 14] is an extension of that in [2], which argues that the ripple size should decrease during the decoding process. The expected ripple size evolution can be achieved by Definition 2 in [8] or Formula (6) in [14]. Both [8, 14] assume that all releases in a single step are unique. This assumption is valid, since the expected number of releases in a single step is small. However, in the partial recovery decoding, the number of releases in a single step is more than that in the entire recovery decoding. For example, to recover fewer than $0.5k$ input symbols [11], all input symbols are released in the first step since all encoded symbols have degree one. Thus, in the partial recovery decoding, more encoded symbols are sunk in each step because there are more encoded symbols with small degrees compared with the entire recovery decoding. In addition, there is no suitable

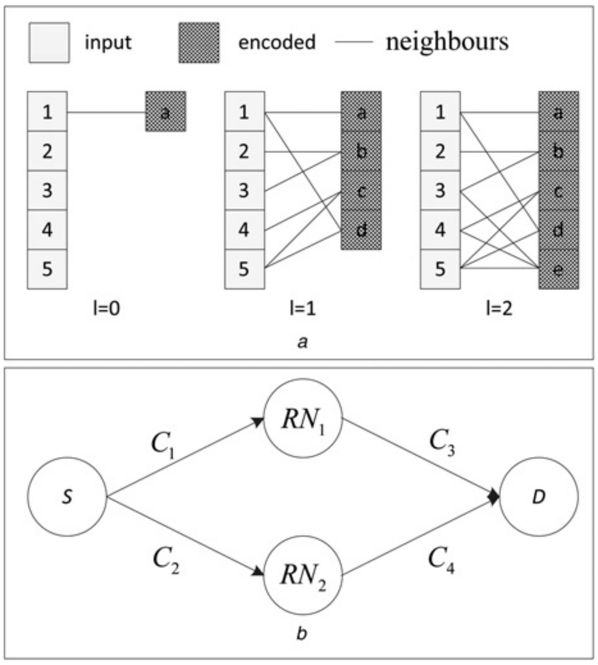


Fig. 1 *Required number of encoded symbols with degree d in step l and in each step, degree d and $\Delta m_{d,l}$ are calculated iteratively*

a I-SDF
b Delay tolerant network

algorithm to calculate the degree distribution of M-PR-LTC when ERS analysis method is employed.

3 Design of optimised partial recovery LT codes

Lemma 2 in [14] expresses the redundancy probability between ripple and the released input symbols in each step. However, it does not take into account the redundancy probability between released input symbols, i.e. the achieved input symbols are not unique. To solve this problem, a new analysis method named I-SDF is proposed in this paper, where only one input symbol is recovered in each step. Let l denote the number of input symbols that have been recovered. At the beginning of decoding, l is set as 0, and l is increased by one in each step. Thus, l also denotes the decoding step. Let $R(R \le k)$ denote the number of input symbols required to be recovered in PR-LTC. We first provide a brief introduction to I-SDF by an example with $k=5$ and $R=4$. Let $\Delta m_{\mathrm{d},l}$ denote the required number of encoded symbols with degree d in step l. In each step, degree d and $\Delta m_{\mathrm{d},l}$ are calculated iteratively. As shown in Fig. 1a, $d=1, \Delta m_{1,0}=1$ in step 0, $d=2, \Delta m_{2,1}=3$ in step 1, $d=3, \Delta m_{3,2}=1$ in step 2, and $\Delta m_{\mathrm{d},3}=0$ in step 3.

To calculate degree d and $\Delta m_{\mathrm{d},l}$ in each step, the release probability of encoded symbols should be analysed. Since the neighbours of an encoded symbol are selected from k input symbols in both entire recovery and partial recovery, we first analyse the decoding of entire recovery.

Lemma 1 (ideal release probability): If the interaction of release probability between encoded symbols is ignored, the ideal release probability of an encoded symbol with degree d in step l is shown as follows

$$I_{\mathrm{d},l} = \begin{cases} 1, & d=1, l=0 \\ \dfrac{(C_l^{d-1} - C_{l-1}^{d-1})C_{k-l}^1}{C_k^d}, & (1 < d \le k, d-1 < l) \\ 0, & \text{others} \end{cases} \tag{1}$$

Proof: The rateless process is a novel generalisation of the classical process of throwing balls randomly into bins. Since an encoded symbol chooses its neighbours independently of all other encoded symbols, the probability that this encoded symbol is released in step l is independent of the probability that any other encoded symbol is released. A well-known analysis of this classic process shows that $(C_l^{d-1} - C_{l-1}^{d-1})C_{k-l}^1$ combinations of encoded symbols are exactly released in each step l, and the total combinatorial number is C_k^d. Hence, Lemma 1 holds up. □

With a degree distribution $\Omega_1, \Omega_2, \ldots, \Omega_k$, we assume that m encoded symbols are sufficient to recover k input symbols. Let $m_{\mathrm{d}} = m \times \Omega_{\mathrm{d}}$ denote the number of encoded symbols with degree d. If $\sum_{d=1}^k m_{\mathrm{d}} \times I_{\mathrm{d},l} = 1$ for all l, the number of encoded symbols released in each step l is perfect and no encoded symbol is redundant. An ideal degree distribution can be achieved by formula (2), where m acts as the normalisation factor.

$$m \times \begin{bmatrix} I_{1,0} & I_{2,0} & \cdots & I_{k,0} \\ I_{1,1} & I_{2,1} & \cdots & I_{k,1} \\ \cdots & \cdots & \cdots & \cdots \\ I_{1,k-1} & I_{2,k-1} & \cdots & I_{k,k-1} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \cdots \\ \Omega_k \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix} \tag{2}$$

$m_{\mathrm{d}}(R < d \le k)$ is set as 0, since the encoded symbol with degree greater than R is a redundancy symbol. Furthermore, $m_1, m_2, \ldots, m_{\mathrm{R}}$ should be adjusted to minimise the risk of redundancy, since $\sum_{l=0}^{R-1} I_{\mathrm{d},l} \le \sum_{l=0}^{k-1} I_{\mathrm{d},l} = 1$. If $\sum_{d=1}^R m_{\mathrm{d}} \times I_{\mathrm{d},l} < 1$, the decoding process fails in step l. Therefore, $\sum_{d=1}^R m_{\mathrm{d}} \times I_{\mathrm{d},l} \ge 1$ should be satisfied. However, if $\sum_{d=1}^R m_{\mathrm{d}} \times I_{\mathrm{d},l} > 1$, it breaks the assumption that only one input symbol is released in each step. It is difficult to guarantee that $\sum_{d=1}^R m_{\mathrm{d}} \times I_{\mathrm{d},l} = 1$. To meet the above assumption, we use the term 'actual release probability' $a_{\mathrm{d},l}(a_{\mathrm{d},l} \le I_{\mathrm{d},l})$, which satisfies $\sum_{d=1}^R m_{\mathrm{d}} \times a_{\mathrm{d},l} = 1$ in each step. The release of other recoverable symbols is delayed if an input symbol has been released in step l. Thus, optimal $m_1, m_2, \ldots, m_{\mathrm{R}}$ is the solution to the following optimisation problem

$$\begin{aligned} \min \quad & \sum_{d=1}^R m_{\mathrm{d}} \\ \text{s.t.} \quad & \sum_{d=1}^R m_{\mathrm{d}} \times a_{\mathrm{d},l} = 1, \ \text{for} \quad l \in [0, R-1] \end{aligned} \tag{3}$$

The problem stated in formula (3) is not linear since $a_{\mathrm{d},l}$ depends on m_{d}. It is difficult to calculate $a_{\mathrm{d},l}$ directly, because $a_{\mathrm{d},l}$ and m_{d} interacts with each other. However, we can easily convert formula (3) into a new optimisation problem defined in formula (4). In formula (4), $\sum_{d=1}^R \sum_{l=0}^{R-1} m_{\mathrm{d}} \times a_{\mathrm{d},l}$ is the number of input symbols released by $\sum_{d=1}^R m_{\mathrm{d}}$ encoded symbols. Clearly, $\sum_{d=1}^R \sum_{l=0}^{R-1} m_{\mathrm{d}} \times a_{\mathrm{d},l} / \sum_{d=1}^R m_{\mathrm{d}}$ denotes the efficiency of our codes.

$$\begin{aligned} \max \quad & \frac{\sum_{d=1}^R \sum_{l=0}^{R-1} m_{\mathrm{d}} \times a_{\mathrm{d},l}}{\sum_{d=1}^R m_{\mathrm{d}}} \\ \text{s.t.} \quad & \sum_{d=1}^R m_{\mathrm{d}} \times a_{\mathrm{d},l} = 1, \ \text{for} \quad l \in [0, R-1] \end{aligned} \tag{4}$$

Let $m_{\mathrm{d},l} = \sum_{j=0}^{l-1} \Delta m_{\mathrm{d},j}$, which denotes the total required number of encoded symbols with degree d in the previous l-1 steps. It is obvious that $\sum_{d=1}^R m_{d,\mathrm{R}}$ encoded symbols are sufficient to recover R input symbols. Thus, m_{d} is calculated as follows

$$m_{\mathrm{d}} = m_{\mathrm{d,R}} = \sum_{l=0}^{R-1} \Delta m_{\mathrm{d},l} \tag{5}$$

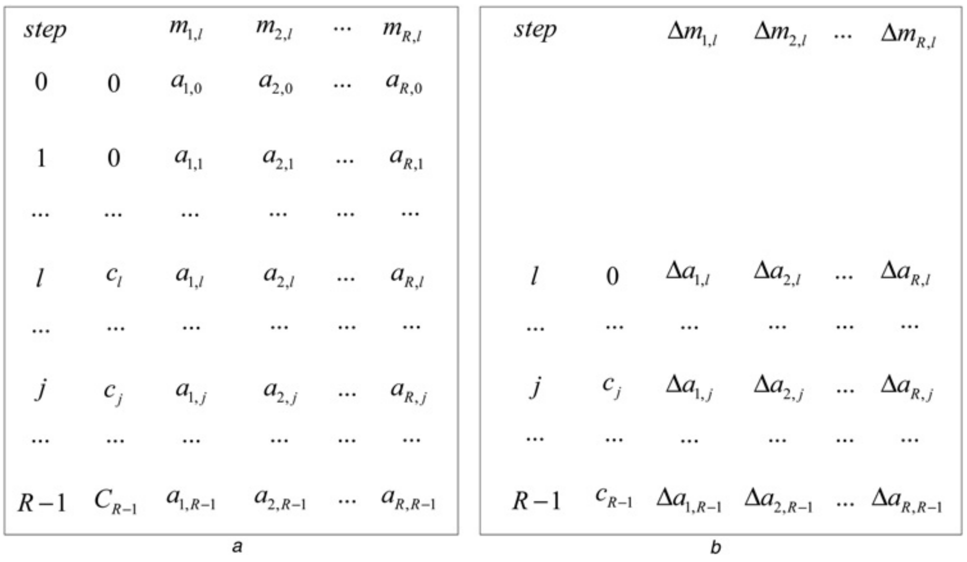
Unfortunately, the problem stated in formula (4) is not linear either. Therefore, we propose an iterative optimisation algorithm. We calculate $\Delta m_{\mathrm{d},l}$ step by step, which is derived by means of mathematical induction. In step 0, $\Delta m_{1,0}=1$, i.e. one input symbol is released by an encoded symbol with degree one. In step l, $\sum_{d=1}^R m_{\mathrm{d},l}$ encoded symbols have been calculated to recover l input symbols in the previous l-1 steps. To calculate $\Delta m_{\mathrm{d},l}$ in step l, the actual release probability $a_{\mathrm{d},l}$ of the encoded symbols with degree d in each step $j(0 \le j \le R-1)$ is calculated in the intermediate decoding process of LT codes, which will be introduced in detail.

3.1 Intermediate decoding of partial recovery LT codes

We analyse the intermediate decoding process when $\sum_{d=1}^R m_{\mathrm{d},l}$ encoded symbols have been received in step l. In each intermediate decoding step j, if $j < l$, one input symbol is released in this step; if $j \ge l$, some input symbols will be released by those encoded symbols in this step when the decoder receives more encoded symbols. Let $1 - c_j$ denote the number of input symbols that would be released by $\sum_{d=1}^R m_{\mathrm{d},l}$ encoded symbols in each step j. As shown in Fig. 2a, c_j and $a_{\mathrm{d},j}$ are calculated step by step.

We assume that encoded symbols with small degrees are released first. This is a reasonable assumption, because encoded symbols with small degrees have a higher release probability. By using this assumption, we can express the maximum number of input symbols released by the encoded symbols with degree d, and we have Lemma 2.

Lemma 2: In the intermediate decoding process of PR-LTC, the maximum number of input symbols that can be released by



$step$		$m_{1,l}$	$m_{2,l}$	$\cdots$	$m_{R,l}$
0	0	$a_{1,0}$	$a_{2,0}$	$\cdots$	$a_{R,0}$
1	0	$a_{1,1}$	$a_{2,1}$	$\cdots$	$a_{R,1}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
l	c_l	$a_{1,l}$	$a_{2,l}$	$\cdots$	$a_{R,l}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
j	c_j	$a_{1,j}$	$a_{2,j}$	$\cdots$	$a_{R,j}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$R-1$	C_{R-1}	$a_{1,R-1}$	$a_{2,R-1}$	$\cdots$	$a_{R,R-1}$

a

$step$		$\Delta m_{1,l}$	$\Delta m_{2,l}$	$\cdots$	$\Delta m_{R,l}$
l	0	$\Delta a_{1,l}$	$\Delta a_{2,l}$	$\cdots$	$\Delta a_{R,l}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
j	c_j	$\Delta a_{1,j}$	$\Delta a_{2,j}$	$\cdots$	$\Delta a_{R,j}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$R-1$	c_{R-1}	$\Delta a_{1,R-1}$	$\Delta a_{2,R-1}$	$\cdots$	$\Delta a_{R,R-1}$

b

Fig. 2 *c_j and $a_{d,j}$ are calculated step by step*
a Intermediate decoding process of PR-LTC
b Incremental decoding process of PR-LTC

encoded symbols with degree d in step j is shown as

$$c^{\max}_{\mathrm{d},j} = 1 - \sum_{i=1}^{d-1} m_{i,l} \times a_{i,j} \tag{6}$$

Proof: As we employ I-SDF analysis method, encoded symbols with small degrees are released first. In addition, we stipulate that only one input symbol be released in each step. Hence, Lemma 2 holds up. □

In Lemma 2, we assume that $a_{i,j}$ is known. However, $a_{i,j}$ interacts with $c^{\max}_{\mathrm{d},j}$, thus it cannot be calculated directly. In the following, we calculate it with a recursive solution. If $I_{i,j} > a_{i,j}$, the encoded symbols that should be released in step j can be postponed to be released in the following steps, which increases the release probability of these encoded symbols in the following steps. In I-SDF, $c^{\max}_{1,0}$ is defined as 1, we show the actual release probability in Lemma 3.

Lemma 3 (actual release probability): In the intermediate decoding process of PR-LTC, the actual release probability of an encoded symbol with degree d in step j is shown as

$$a_{\mathrm{d},j} = \min\left(\frac{c^{\max}_{\mathrm{d},j}}{m_{\mathrm{d},l}}, \frac{C_j^{d-1}C_{k-j}^1}{C_k^d} - \sum_{i=0}^{j-1} a_{\mathrm{d},i}\right) \tag{7}$$

Proof: If an encoded symbol with degree d has not been processed until step j, the release probability of this encoded symbol is $C_j^{d-1}C_{k-j}^1/C_k^d$. If this encoded symbol has been partially processed in the previous steps, the release probability of this encoded symbol is $C_j^{d-1}C_{k-j}^1/C_k^d - \sum_{i=0}^{j-1} a_{\mathrm{d},i}$. From Lemma 2, the maximum number of input symbols released by encoded symbols with degree d in step j is $c^{\max}_{\mathrm{d},j}$. Thus, the maximum release probability of this encoded symbol is $c^{\max}_{\mathrm{d},j}/m_{\mathrm{d},l}$. Hence, Lemma 3 holds up. □

Lemma 4: In the intermediate decoding process of PR-LTC, the number of input symbols remaining unreleased in step j is shown as

$$c_j = 1 - \sum_{d=1}^{l} m_{\mathrm{d},l} \times a_{\mathrm{d},j} \tag{8}$$

Proof: As we stipulate that only one input symbol be released in each step, Lemma 4 holds up. □

3.2 Incremental decoding of partial recovery LT codes

On the basis of the intermediate decoding of $\sum_{d=1}^{R} m_{\mathrm{d},l}$ encoded symbols, let $\Delta a_{\mathrm{d},l}$ denote the actual release probability of the incremental encoded symbols ($\Delta m_{\mathrm{d},l}$) with degree d in step l. As shown in Fig. 2*b*, input symbols remaining unreleased in step l will be released by $\sum_{d=1}^{R} \Delta m_{\mathrm{d},l}$ encoded symbols.

Lemma 5 (actual release probability): In the incremental decoding process of PR-LTC, the actual release probability of an encoded symbol with degree d in step l is shown as follows

$$\Delta a_{\mathrm{d},l} = \frac{C_l^{d-1}C_{k-l}^1}{C_k^d} \tag{9}$$

Proof: Since the incremental encoded symbols are new, they are not released in steps before l. Thus, Lemma 5 holds up. □

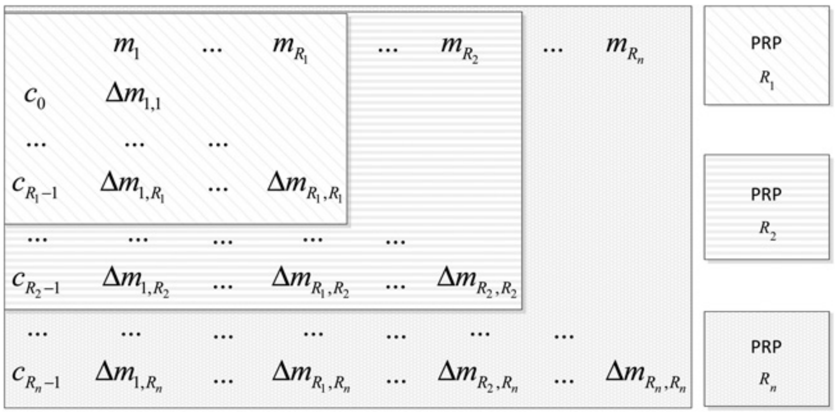
In the incremental decoding process, the number of input symbols remaining unreleased in each step can be obtained from Lemma 4. It is obvious that $\Delta a_{d,j}$ should be calculated step by step since it depends on $\Delta m_{1,l}$, $\Delta m_{2,l}$, …, $\Delta m_{\mathrm{R},l}$. The maximum number of symbols that can be released by the encoded symbols with degree d in step j is shown in formula (10). In addition, the actual release probability of encoded symbols with degree d in step j is shown in formula (11).

$$\Delta c^{\max}_{\mathrm{d},j} = c_j - \sum_{i=1}^{d-1} \Delta m_{i,l} \times \Delta a_{i,l} \tag{10}$$

$$\Delta a_{\mathrm{d},j} = \min\left(\frac{\Delta c^{\max}_{\mathrm{d},j}}{\Delta m_{\mathrm{d},l}}, \frac{C_j^{d-1}C_{k-j}^1}{C_k^d} - \sum_{i=l}^{j-1} \Delta a_{\mathrm{d},i}\right) \tag{11}$$

It is obvious that $\Delta m_{1,l}$, $\Delta m_{2,l}$, …, $\Delta m_{\mathrm{R},l}$ is the solution to the following optimisation problem

$$\begin{aligned} \max \quad & \frac{\sum_{\mathrm{d}=1}^{R}\sum_{j=l}^{R-1} \Delta m_{\mathrm{d},l} \times \Delta a_{\mathrm{d},j}}{\sum_{\mathrm{d}=1}^{R} \Delta m_{\mathrm{d},l}} \\ \text{s.t.} \quad & \sum_{d=1}^{R} \Delta m_{\mathrm{d},l} \times \Delta a_{\mathrm{d},l} = c_l \end{aligned} \tag{12}$$



m_1 … m_{R_1} … m_{R_2} … m_{R_n}
c_0 $\Delta m_{1,1}$
c_{R_1-1} $\Delta m_{1,R_1}$ … $\Delta m_{R_1,R_1}$
c_{R_2-1} $\Delta m_{1,R_2}$ … $\Delta m_{R_1,R_2}$ … $\Delta m_{R_2,R_2}$
c_{R_n-1} $\Delta m_{1,R_n}$ … $\Delta m_{R_1,R_n}$ … $\Delta m_{R_2,R_n}$ … $\Delta m_{R_n,R_n}$
PRP R_1
PRP R_2
PRP R_n

Fig. 3 *Multiple partial recovery LT codes*

It is assumed that c_l input symbols are only released by encoded symbols with specified degree t in each step. This is a valid assumption, since c_l is small and decreases as l increases. Thus, the above optimisation problem is converted into the following new one

$$\begin{aligned} \max \quad & \frac{\sum_{j=l}^{R-1} \Delta m_{t,l} \times \Delta a_{t,j}}{\Delta m_{t,j}} \\ \text{s.t.} \quad & \Delta m_{t,l} \times \Delta a_{t,l} = c_l \end{aligned} \tag{13}$$

The problem stated in formula (13) is linear since the constraint is linear. With the above lemma, we can obtain the specific degree t and $\Delta m_{t,l}$ in step l.

3.3 Partial recovery LT codes

Initially, l is set as 0. In step 0, $\Delta m_{1,0} = 1$ is calculated, and then l is set as 1. In step 1, the intermediate decoding of $\sum_{d=1}^{R} \sum_{j=0}^{0} \Delta m_{d,j}$ encoded symbols and the incremental decoding of $\Delta m_{1,1}$, $\Delta m_{2,1}$, …, $\Delta m_{d,1}$ encoded symbols are analysed. Then, $\Delta m_{1,1}$, $\Delta m_{2,1}$, …, $\Delta m_{d,1}$ is calculated, and l is set as 2. This process repeats until l is R.

Finally, $m_{1,R}$, $m_{2,R}$, …, $m_{R,R}$ are calculated. The achieved degree distribution is shown in the following Definition 1.

Definition 1: The degree distribution of PR-LTC is shown as

$$\Omega_d = \frac{m_{d,R}}{\sum_{i=1}^{R} m_{i,R},\ d \in [1, R]} \tag{14}$$

4 Design of optimised multiple partial recovery LT codes

In this section, we design a degree distribution of M-PR-LTC by employing multi-objective optimisation algorithms. To obtain high ISRR, the degree distribution should be tuned by considering all PRPs. In PR-LTC, an optimum degree distribution Ω has been proposed to minimise the number of encoded symbols required to recover partial input symbols. Let R_1, R_2, …, R_n denote the number of input symbols required to be recovered in each PRP. Without loss of generality, let us assume that $R_1 < R_2 < \ldots < R_n$ so that PRPs can be analysed step by step. Based on the desired PR-LTC, several optimum degree distributions Ω_1, Ω_2, …, Ω_n have been selected for PRP R_1, R_2, …, R_n, respectively. However, these degree distributions conflict with each other. Let $F_{R_1}(\Delta m_{t,l})$, $F_{R_2}(\Delta m_{t,l})$, …, $F_{R_n}(\Delta m_{t,l})$ denote the conflicting objective functions in step l. The optimal $F_{R_i}(\Delta m_{t,l})$ is the solution to the following optimisation problem

$$\begin{aligned} \max \quad & \frac{\sum_{j=l}^{R_i-1} \Delta m_{t,l} \times \Delta a_{t,j}}{\Delta m_{t,j}} \\ \text{s.t.} \quad & \Delta m_{t,l} \times \Delta a_{t,l} = c_l \end{aligned} \tag{15}$$

The problem is to find the decision vectors that maximise all objective functions. In the simple case with a single objective function, the problem is the conventional maximise problem. However, in the problem with multi-objective functions, we have to deal with multiple optimum answers called Pareto optimal.

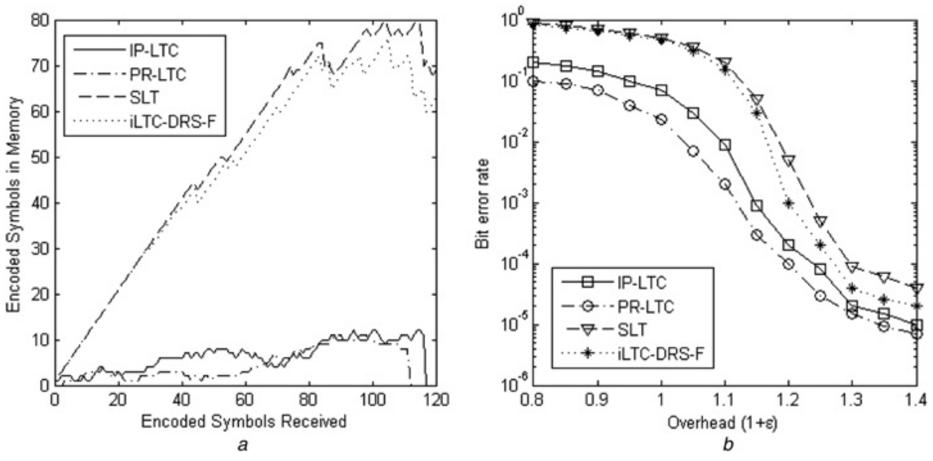


Fig. 4 *Number of encoded symbols cached in memory as a function of the number of encoded symbols received*

a Number of encoded symbols in memory as a function of the number of encoded symbols received
b BER as a function of overhead

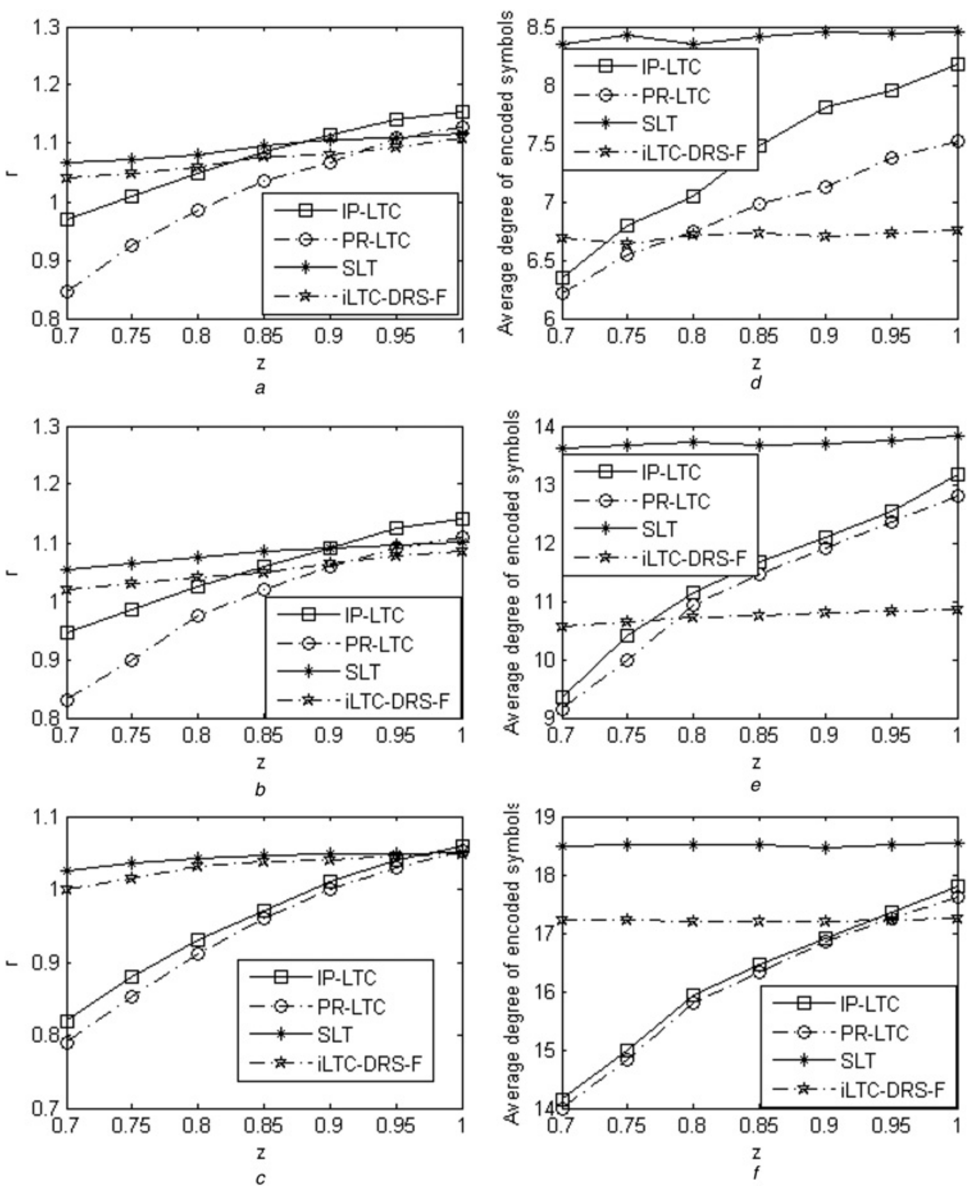


Fig. 5 *Ratio of the encoded symbols as a function of the ratio of input symbols at*
a $k = 100$
b $k = 1000$
c $k = 10000$; Average degree of encoded symbols as a function of the ratio of input symbols at
d $k = 100$
e $k = 1000$
f $k = 10000$

Therefore, we propose small partial recovery point first (SPRPF) to improve the ISRR of small PRPs. As shown in Fig. 3, in each step $l, l \in [0, R_1)$, specific degree t and $\Delta m_{t,l}$ are calculated by $F_{\mathrm{R}_1}(\Delta m_{t,l})$, and in each step $l, l \in [R_1, R_2)$, specific degree t and $\Delta m_{t,l}$ are calculated by $F_{\mathrm{R}_2}(\Delta m_{t,l})$, and so forth.

In SPRPF, the required number of encoded symbols in step $l, l \in [R_i, R_{i+1})$ impacts the recovery of the following PRPs. To adjust the impact between each PRP, we assign a weight to each PRP. Let $w_1, w_2, \ldots, w_n$ denote the weight of each PRP $R_1, R_2, \ldots, R_n$, respectively.

Lemma 6 (weighted M-PR-LTC): The weighted optimisation problem is shown as

$$\begin{aligned} \max \quad & \frac{\sum_{i=1}^{n} w_i \left(\sum_{j=l}^{R_i} \Delta m_{t,l} \times \Delta a_{t,j} \right)}{\Delta m_{t,j}} \\ \text{s.t.} \quad & \Delta m_{t,l} \times \Delta a_{t,l} = c_l \end{aligned} \tag{16}$$

Proof: In step $l, l \in [R_i, R_{i+1})$, $\Delta m_{t,L}$ contributes to the recovery of $R_i, R_{i+1}, \ldots, R_n$. The assigned weights adjust the objective function of the optimisation problem, i.e. they adjust the recovery of each PRP. Hence, Lemma 6 holds up. □

We evaluate the weighted M-PR-LTC and choose the degree with the highest weighted efficiency in each step l. The appropriate degree distribution for specific situation can be found by setting the appropriate weights.

5 Analysis of optimised multiple partial recovery LT codes

In this section, we theoretically analyse the properties of optimised M-PR-LTC. By using I-SDF, the required number of encoded symbols in each step is analysed accurately. Thus, an optimum degree distribution is calculated, which approximates to the ideal degree distribution. The following proposition shows that the

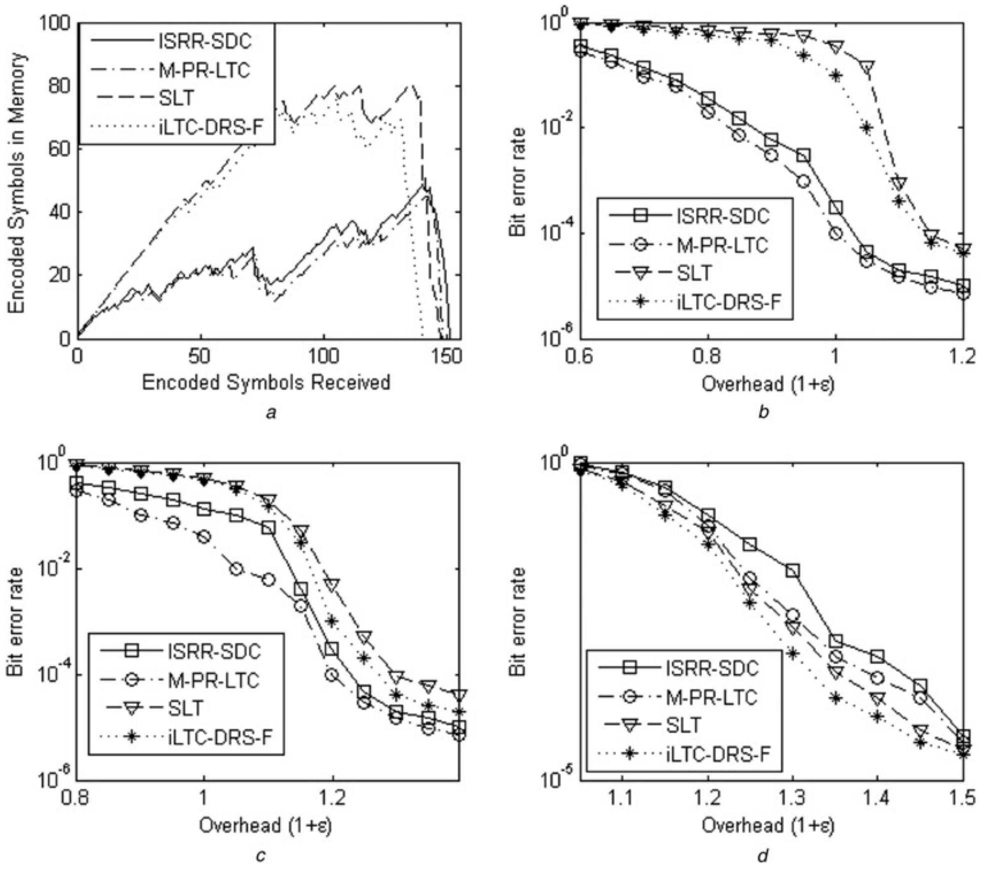


Fig. 6 *Proposed design exhibits a superior performance by using partial recovery*

a Number of encoded symbols in memory as a function of the number of encoded symbols received; BER as a function of overhead at
b $z = 0.5$
c $z = 2/3$
d $z = 1$

decoder requires only E_i encoded symbols to release R_i input symbols, and the average degree of encoded symbols is relatively low.

Lemma 7: A decoder requires E_i encoded symbols to release R_i input symbols.

$$E = \max\left\{\frac{\left(\sum_{i=1}^{R_n} m_{i,\mathrm{R}_n} \times m_{\mathrm{d},\mathrm{R}_i}\right)}{m_{\mathrm{d},\mathrm{R}_n},\ d \in [1, R_i]}\right\} \tag{17}$$

Proof: To recover R_i input symbols, $m_{\mathrm{d},\mathrm{R}_i}$ encoded symbols are required for each degree $d \in [1, R_i]$. The sum of encoded symbols with degree d is $m_{\mathrm{d},\mathrm{R}_n}$, and the sum of all encoded symbols is $\sum_{i=1}^{R_n} m_{i,\mathrm{R}_n}$. Thus, $\left(\sum_{i=1}^{R_n} m_{i,\mathrm{R}_n} \times m_{\mathrm{d},\mathrm{R}_i}\right)/m_{\mathrm{d},\mathrm{R}_n}$ encoded symbols are adequate to contain $m_{\mathrm{d},\mathrm{R}_i}$ encoded symbols for each degree $d \in [1, R_i]$. By choosing the maximum one, we yield the statement of Lemma 7. □

Lemma 8: The average degree of an encoded symbol under the proposed degree distribution of M-PRP-LTC is given as

$$\overline{d} = \frac{\left(\sum_{d=1}^{R_n} d \times m_{\mathrm{d},R_n}\right)}{\sum_{d=1}^{R_n} m_{\mathrm{d},R_n}} \tag{18}$$

Proof: The proof is obtained from the definition, as shown in (19). By substituting formula (14), we yield the statement of Lemma 8. □

$$\overline{d} = \sum_{d=1}^{R_n} d \times \Omega_{\mathrm{d}} \tag{19}$$

An encoded symbol that is not released in step l may be a redundancy symbol in the following steps, if it should be released in step l. We can see that $\left(C_l^{d-1}C_{k-l}^{1}/C_k^d\right) + \left(C_l^d/C_k^d\right) = \sum_{j=0}^{l}\left(\left(C_j^{d-1} - C_{j-1}^{d-1}\right)C_{k-j}^{1}/C_k^d\right)$, which means that the sunk probability of an encoded symbol is $\left(C_l^d/C_k^d\right)$ if it does not contribute to release input symbols in steps before l.

6 Numerical results

We have presented our PR-LTC and M-PR-LTC, and outlined its properties for the practical implementation. In this section, we evaluate its performance by comparing intermediate performance LT codes (IP-LTC) [11], SLT and iLTC-DRS-F with our PR-LTC, and comparing ISRR-SDC (ISRR of selected designed codes), SLT and iLTC-DRS-F with our M-PR-LTC through simulations. ISRR-SDC is constructed without rateless symbol sorting algorithm.

6.1 Comparison of PR-LTC, IP-LTC, SLT and iLTC-DRS-F

We compare PR-LTC against IP-LTC, SLT and iLTC-DRS-F in a single unicast stream. In each round of simulation, an encoded symbol is generated and transmitted, until R input symbols have been recovered by receiver. We assume that the channel between transmitter and receiver is an erasure one, and loss rate $r = 0.1$.

Fig. 4 shows two comparison at $k = 128$ and $R = 100$. Fig. 4*a* shows the number of encoded symbols cached in memory as a function of the number of encoded symbols received. We can see that both IP-LTC and PR-LTC outperform other methods since the degree distributions of IP-LTC and PR-LTC are designed for

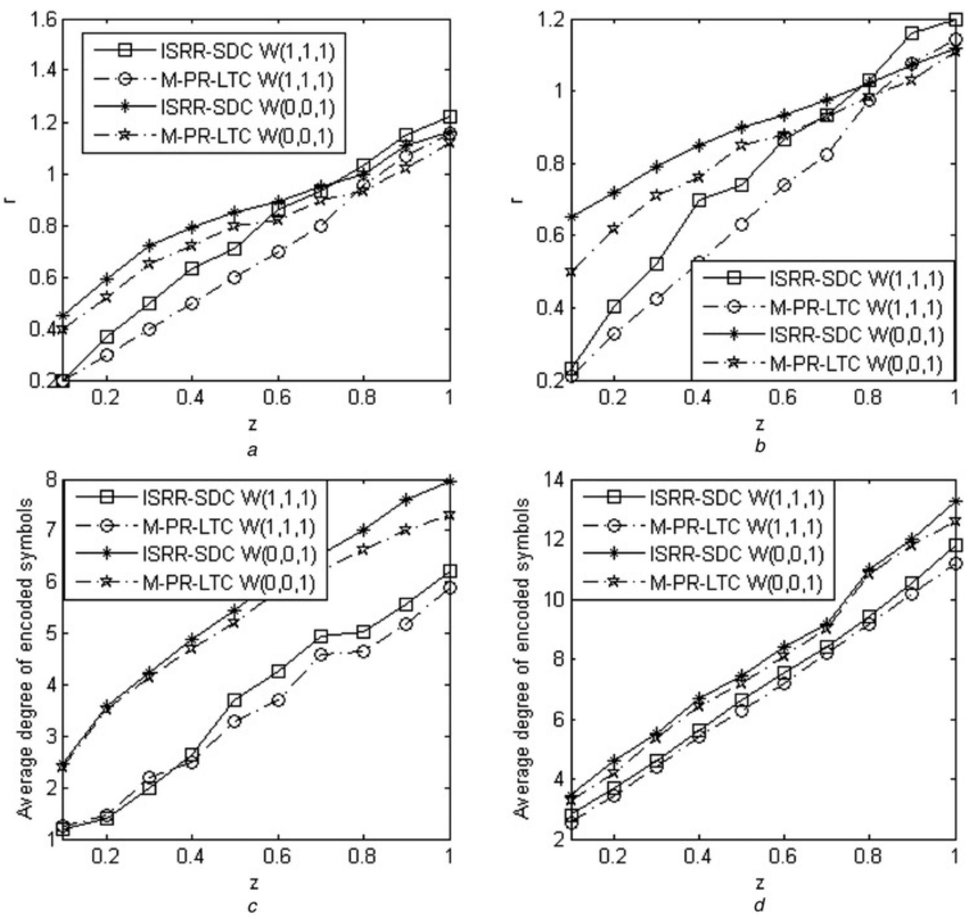


Fig. 7 *Ratio of the encoded symbols as a function of the ratio of input symbols when*
a $k = 100$
b $k = 1000$; Average degree of encoded symbols as a function of the ratio of input symbols when
c $k = 100$
d $k = 1000$

partial recovery. We can see that PR-LTC needs less memory than IP-LTC. This is due to the more accurate degree distribution used for finite k, which releases the encoded symbol with high probability. Fig. 4*b* shows the BER as a function of the overhead. The result shows that both IP-LTC and PR-LTC outperform other methods at a lower overhead. However all LT-like codes have an error floor, which is a well-known drawback.

The ratio of the encoded symbols as a function of the ratio of input symbols ($z \in [0.7, 1.0]$) is shown in Figs. 5*a–c* at $k = 100$, $k = 1000$, and $k = 10{,}000$, respectively. We can see that PR-LTC and IP-LTC require fewer encoded symbols than other schemes when z is small since the degree distributions of IP-LTC and PR-LTC are specially designed for each z. Due to the introduction of feedback, SLT and iLTC-DRS-F require fewer encoded symbols than other schemes when $z = 1$. Since degree distributions of IP-LTC are formulated for $k \to \infty$, as the number of input symbols decreases, performance of PR-LTC is better than IP-LTC. Figs. 5*d–f* show the average degree of encoded symbols as a function of the ratio of input symbols at $k = 100$, $k = 1000$, and $k = 10{,}000$, respectively. Note that the average degree of encoded symbols for PR-LTC increases more slowly than IP-LTC. This is due to the strategy of small degree first, which generates encoded symbols with small degrees as many as possible.

6.2 Comparison of M-PR-LTC, ISRR-SDC, SLT and iLTC-DRS-F

In this subsection, we compare M-PR-LTC against ISRR-SDC, SLT and iLTC-DRS-F in a single unicast stream. In each round of simulation an encoded symbol is generated and transmitted, until all input symbols are recovered by receiver. In the scheme with multiple PRPs, there is a weight for each PRP, following the configuration in a related literature [12]. As shown in Fig. 6*a*, we can see that there are three PRPs in ISRR-SDC and M-PR-LTC with $W(1,1,1)$. ISRR-SDC and M-PR-LTC outperform other methods except at the end of the encoding since ISRR-SDC and M-PR-LTC improve the partial recovery performance at the expense of the entire recovery. Figs. 6*b–d* show the BER as a function of the overhead at $z = 0.5$, $z = 2/3$ and $z = 1$, respectively. We can see that the proposed design exhibits a superior performance by using partial recovery.

The ratio of encoded symbols as a function of the ratio of input symbols is shown in Figs. 7*a* and *b* at $k = 100$ and $k = 1000$, respectively. We can see that the M-PR-LTC outperforms ISRR-SDC in both $W(1,1,1)$ and $W(0,0,1)$. In $W(0,0,1)$, the partial performance of both M-PR-LTC and ISRR-SDC is poorer, because both of them are constructed for recovering all input symbols. In $W(1,1,1)$, M-PR-LTC needs fewer encoded symbols than ISRR-SDC in each PRP. This is because the number of encoded symbols required for each step is calculated accurately by an optimisation algorithm, which releases the encoded symbols with high probability in each PRP. In addition, we can see that the difference between M-PR-LTC and ISRR-SDC decreases as k increases. Figs. 7*c* and *d* show the average degree of encoded symbols as a function of the ratio of input symbols at $k = 100$ and $k = 1000$, respectively. Note that the average degree of encoded symbols for M-PR-LTC increases more slowly than ISRR-SDC. This is due to the optimum degree distribution formulated by the iterative optimisation algorithm, which approximates to the ideal degree distribution in each region.

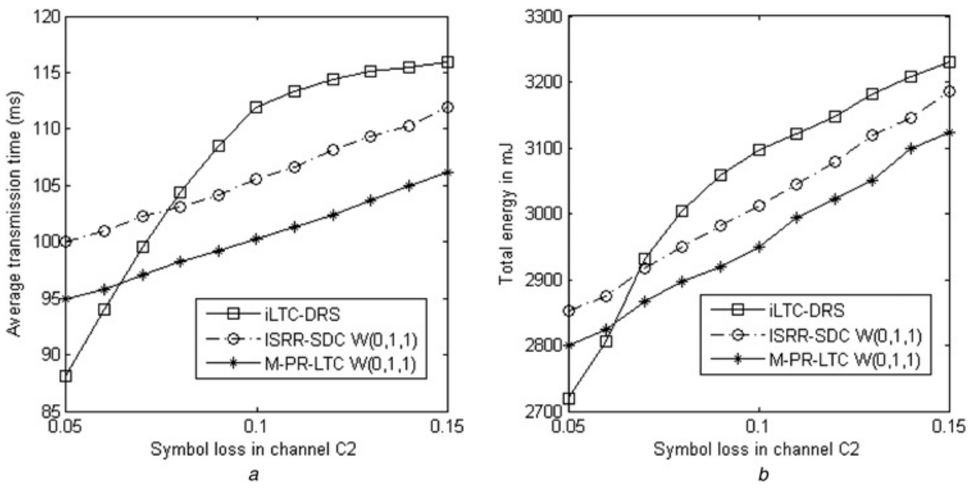


Fig. 8 *Average transmission time for different symbol losses in channel C_2 and energy consumption of different schemes by varying loss rates*

a Average transmission time as a function of symbol loss in channel C_2
b Total energy used by all nodes as a function of symbol loss in channel C_2

6.3 Communication of broadcast

In this subsection, we study the communication cost in the delay tolerant network (DTN). As shown in Fig. 1*b*, our experiment consists of four sensor nodes, in which one node named *S* serves as the broadcaster, two nodes named RN_1, RN_2 serve as relay nodes and one node named *D* serves as receiver. The loss rate of channel C_1 from *S* to the RN_1 is set as 0.05, and the loss rate of channel C_2 from *S* to the RN_2 is varied from 0.05 to 0.15. The loss rate of channel C_3, C_4 from RN_1, RN_2 to *D* are set as 0.05. First, *S* broadcasts encoded symbols to RN_1, RN_2 until one of RN_1, RN_2 has recovered all input symbols. The other one may only recover partial input symbols. Second, RN_1, RN_2 transmit encoded symbols to *D* until *D* has recovered all input symbols. As RN_1, RN_2 require partial recovery and the PRP is unknown, we employ three PRPs by following the configuration in a related literature [12]. As there is no much difference between the loss rate of channel C_1 and that of channel C_2, we design codes with weights $W(0,1,1)$. We compare M-PR-LTC against iLTC-DRS since it is difficult to use feedback in the DTN.

Fig. 8*a* shows the average transmission time for different symbol losses in channel C_2. It is obvious that the average transmission time of M-PR-LTC increases more slowly as the loss rate increases, compared with other two codes. However, iLTC-DRS performs better when the loss rate of channel C_2 is set as 0.05 and 0.06 since the number of encoded symbols received by RN_1 is almost similar to that received by RN_2. Fig. 8*b* shows the energy consumption of different schemes by varying loss rates. The energy consumption is measured by using the PowerTOSSIM simulator. Obviously, M-PR-LTC requires less energy than other two codes when loss rate of channel C_2 is set as more than 0.07. This is because iLTC-DRS need more energy to recover all input symbols in two relay nodes. Compared with M-PR-LTC, ISRR-SDC needs more encoded symbols to recover the same number of input symbols in relay nodes.

7 Conclusions

In this paper, we first propose optimised PR-LTC, in which a new degree distribution is designed based on I-SDF analysis method. When multiple PRPs are introduced, we note that the optimised degree distributions of different PRPs conflict with each other, which means that the multi-objective problem must be solved. After analysing how the required number of encoded symbols in each PRPs impacts the degree of encoded symbols, we propose a new degree distribution with weights to balance the impact of different PRPs.

Through simulations, we demonstrate that optimised PR-LTC outperforms IP-LTC, SLT and iLTC-DRS-F in terms of memory usage, BER, average overhead and average degree of encoded symbols for specified PRP. This is because optimised PR-LTC uses a new optimum degree distribution, which minimises the number of sunk symbols in each decoding step to reduce the redundancy of entire encoded symbols. We also note that optimised M-PR-LTC outperforms other methods in terms of memory usage, BER, average overhead, average degree of encoded symbols and energy consumption. Note that optimised PR-LTC can achieve arbitrary partial recovery in LT codes. Hence, different types of LT codes can be constructed for different scenarios. Our future work plans to apply this technology to the broadcast communication in wireless relay networks more accurately.

8 Acknowledgment

This work was jointly supported by: (1) the National Basic Research Program of China (grant no. 2013CB329102); (2) National Natural Science Foundation of China (grant nos. 61471063, 61421061, 61372120, 61271019, 61101119 and 61121001); (3) the Key (Keygrant) Project of Chinese Ministry of Education. (grant no. MCM20130310); (4) Beijing Municipal Natural Science Foundation (grant no. 4152039); (5) Beijing Higher Education Young Elite Teacher Project (grant no. YETP0473); (6) Spanish Research Council (grant no. TIN2014-46883); (7) Regional Government of Madrid (grant no. S2013/ICE-2894) cofunded by FSE & FEDER.

9 References

1 Byers, J., Luby, M., Mitzenmacher, M., *et al.*: 'A digital fountain approach to reliable distribution of bulk data'. Proc. of SIGCOMM, Vancouver, BC, CA, September 1998, pp. 56–67
2 Luby, M.: 'LT codes'. Proc. of the ACM Symp. on Foundations of Computer Science, Vancouver, BC, CA, November 2002, pp. 37–47
3 Shokrollahi, A.: 'Raptor codes', *Inf. Theory*, 2006, **52**, (6), pp. 2551–2567
4 Beimel, A., Dolev, S., Singer, N.: 'RT oblivious erasure correcting', *IEEE Trans. Netw.*, 2007, **15**, (6), pp. 1321–1332
5 Hagedorn, A., Agarwal, S., Starobinski, D., *et al.*: 'Rateless coding with feedback'. Proc. of INFOCOM, Rio de Janeiro, Brazil, April 2009
6 Lei, Z., Jianxin, L., Jingyu, W., *et al.*: 'Diversified SLT codes based on feedback for communication over wireless networks'. Proc. Int. Conf. Global Information Infrastructure Symp., Trento, Italy, October 2013, pp. 1–6

7 Sorensen, J.H., Koike-Akino, T., Orlik, P.: ‘Rateless feedback codes’. Proc. Int. Conf. Information Theory Proc., Cambridge, England, July 2012, pp. 1767–1771
8 Lei, Z., Jianxin, L., Jingyu, W., *et al.*: ‘Design of improved Luby transform codes with decreasing ripple size and feedback’, *IET Commun.*, 2014, **8**, (8), pp. 1409–1416
9 Kamra, A., Misra, V., Feldman, J., *et al.*: ‘Growth codes: maximizing sensor network data persistence’. Proc. of Applications, Technologies, Architectures, Protocols Computer Communications, 2006, pp. 255–266
10 Nikolaos, T., Rethnakaran, P., Pascal, F.: ‘Growth codes: Intermediate performance analysis and application to video’, *IEEE Trans. Commun.*, 2013, **61**, (11), pp. 4710–4721
11 Sanghavi, S.: ‘Intermediate performance of rateless codes’. Proc. 2007 IEEE Inf. Theory Workshop, pp. 478–482
12 Talari, A., Rahnavard, N.: ‘On the intermediate symbol recovery rate of rateless codes’, *IEEE Trans. Commun.*, 2012, **60**, (5), pp. 1237–1242
13 Kim, S., Lee, S.: ‘Improved intermediate performance of rateless codes’. Proc. of ICACT 2009, February 2009, pp. 1682–1686
14 Sorensen, J.H., Popovski, P., Ostergaard, J.: ‘Design and analysis of LT codes with decreasing ripple sizej’, *IEEE Trans. Commun.*, 2012, **60**, (11), pp. 1–7
15 Sejdinovic, D., Piechocki, R., Doufexi, A.: ‘AND-OR tree analysis of distributed LT codes’. Proc. of Information Theory Workshop on Networking and Information Theory, 2009, pp. 261–265