LT Codes Decoding: Design and Analysis

Feng Lu
Electrical and Computer Engineering Department
University of California, San Diego
Email: f1lu@ucsd.edu

Chuan Heng Foh, Jianfei Cai and Liang-Tien Chia
School of Computer Engineering
Nanyang Technological University, Singapore 639798
Email: {aschfoh, asjfcai, asltchia}@ntu.edu.sg

***Abstract*—LT codes provide an efficient way to transfer information over erasure channels. Past research has illustrated that LT codes can perform well for a large number of input symbols. However, it is shown that LT codes have poor performance when the number of input symbols is small. We notice that the poor performance is due to the design of the LT decoding process. In this respect, we present a decoding algorithm called full rank decoding that extends the decodability of LT codes by using Wiedemann algorithm. We provide a detailed mathematical analysis on the rank of the random coefficient matrix to evaluate the probability of successful decoding for our proposed algorithm. Our studies show that our proposed method reduces the overhead significantly in the cases of small number of input symbols yet preserves the simplicity of the original LT decoding process.**

I. Introduction

The concept of digital fountain codes was first introduced by Byers *et al.* [1] in 1998 for information distribution. It is stimulated on an analogy to fountain spraying water drops, which are collected into a bucket. When a bucket is being filled from a fountain, which droplet falling into the bucket is not as important, as long as the bucket is being filled. Once the bucket is full, collection process ends, and further processing on decoding the content in the bucket will take place. With this idea, servers are spraying stochastically encoded packets of data, which clients then collect. Once enough packets are obtained, the original data can be reconstructed. Throughout the process, which packets are obtained should not matter. An ideal digital fountain code should exhibit the following properties. Given a message that would require k packets of length l for transmission, this code would allow the source to generate an infinite supply of encoded packets of length l. The client can then reconstruct the original message once any k encoded packets of the infinite supply of encoded packets are received. Moreover, the complexity of the reconstruction process should be linear in k.

Approximations to a digital fountain has existed for some time, such as the Low-Density-Priority Code (LDPC) code [2] and the Reed-Solomon (RS) code [3], which are generated by loosening the requirements in various ways. LT codes [4] are the first realization of a true digital fountain code. The symbol length for the code can be arbitrary, from one-bit binary symbols to general l-bit symbols. If an original file can be described by k symbols, then each coded packet can be independently generated, and the content of the original file can be recovered from any $k+O(\sqrt{k}\ln^2(k/\delta))$ packets with the probability of $1-\delta$ by an average of $O(k\ln(k/\delta))$ symbol operations. Following [4], Raptor codes were developed by Shokrollahi [5] as rateless fountain codes with even smaller encoding and decoding complexities than LT codes. Raptor codes make use of a pre-coder before the typical LT process.

It has been shown in [5] that LT codes perform very well for large values of k. By optimizing the degree distribution, the packet overhead ϵ can be as small as 3% for $k > 100000$. In reality, small numbers of symbols are often encountered. For example, in video streaming applications, k can be the number of frames in a GOP, which is typically in the range of 10 to 100. Under this configuration, high packet overhead is observed [7], [8]. Hyytia *et al.* [7] optimize the configuration parameters of the degree distribution for LT codes for small numbers of symbols. They proposed two approaches, namely Markov chain and combination methods. However, as indicated in the paper, the methods are not scalable, and can only handle symbols $k \leq 10$ (numerically up to 30 for the combination method). The authors in [8] define a new degree distribution function and apply the Gaussian elimination method [9] for decoding. While the packet overhead ϵ is reduced, the decoding complexity increases significantly.

Addressing the shortcoming of the LT codes and other approaches mentioned above, we propose a new decoding process called full rank decoding algorithm. While retaining the original LT encoding and decoding process in maximal possible extent to preserve the low complexity benefit of LT codes, our proposed decoding process introduces a method that extends the decodability of LT decoding process preventing LT decoding from terminating prematurely. We present the proposed full rank decoding algorithm in the next section. In Section III, a detailed mathematical analysis on the rank of the random coefficient matrix is given to evaluate success decoding probability for our proposed algorithm. Numerical results are shown in Section IV to demonstrate the effectiveness of our method, with important conclusions drawn in Section V.

II. Full Rank LT Decoding Process

A. LT Decoding Drawbacks

In the LT decoding process, a symbol is taken off from the ripple each time and the related packets are updated accordingly. The process terminates when there is no more symbol left in the ripple, which is said to be successful if all symbols are recovered. Hence, in the process, it is important not to let the symbols in the ripple run out before all symbols can be recovered. Since the arrival of new symbols into the ripple also depends on the degrees of the received packets, the degree distribution used in the encoding is also a crucial factor to the final success of the decoding process.

Our further investigation on the LT decoding process reveals that when LT decoding process terminates and reports a failure, often the undecodable packets can be decoded to recover all symbols within the packets by other means, such as Gaussian elimination [10]. In other words, when viewing a packet as an equation formed by combining linearly a number of variables (or symbols) in $GF(2)$, the set of available equations (or packets) may give a full rank such that a numerical solver (or decoder) can determine all variables (or symbols). Attributing to the design of the LT decoding process, the method recovers only partial but not all symbols.

B. Full Rank Decoding

Knowing that the LT decoding process terminates early due to lack of symbols in the ripple, in this paper, we propose a method, called *full rank decoding* to extend the decodability of LT codes. Specifically, given a set of packets of full rank, whenever the ripple is empty causing an early termination, a particular symbol is *borrowed* and decoded through some other method, and then placed into the ripple for the LT decoding process to continue. This procedure is repeated until the LT decoding process terminates with a success. Note that in the case of full rank, any picked borrowed symbol can be decoded with a suitable method to allow continuation of the LT decoding process. Unlike the previous approach [8], [10] where the decoding procedure completely or partially relies on Gaussian elimination, our proposed decoding algorithm mainly uses LT decoding to recover symbols, and triggers Wiedemann algorithm when LT decoding fails in order to recover a borrowed symbol, then returns back to LT decoding to recover subsequent symbols.

We use the following criteria to decide which symbol should be borrowed. Considering decoding k symbols with n packets, given the above analogy between a packet and a linear equation, let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$ denote the coefficient vectors with the entries in $GF(2)$ at the intermediate decoding process when the ripple is empty. Let b_j denote the index of the borrowed symbol, and b_j is decided by

$$b_j = \arg\max_j \left(V(j) | \mathbf{V} := \textstyle\sum_{i=1}^{n} \mathbf{v_i}\right). \tag{1}$$

The above sum operation is defined in real number field, and $V(j)$ gives the value at index j of the vector $\mathbf{V}$. The basic idea here is to choose the symbol that is carried by most packets. Incorporating borrowed symbols into the LT decoding, our proposed full rank LT decoding is shown in Algorithm 1.

C. Recovering the Borrowed Symbol

One of the critical designs in our proposed idea is the recovery of a borrowed symbol. We need to seek for a suitable method that can recover only a single symbol using a low computational cost. Common techniques such as Gaussian elimination are inadequate as they are designed to recover all symbols and often require high computing power. We shall present our method in the following.

Algorithm 1 Full rank LT decoding

```
1: rank(M):=k, ripple: $r$, borrowed symbol buffer: $bu$
2: while not all source symbols are recovered
3:    release all degree one symbols to the ripple
4:    if ripple is empty
5:       compute the index of borrowed symbol $bj$ as in (1)
6:       put index $bj$ into buffer $bu$
7:       recover the borrowed symbol
8:       release the borrowed symbol into ripple
9:    end if
10:   remove a symbol $r_l$ from the ripple $r$
11:   for all encoding symbols $c_i$
12:      if $c_i$ includes $r_l$
13:         $c_i := c_i \oplus r_l$
14:         update $\mathbf{v_i}(j)$:=0 and reduce degree of $c_i$ by one
15:      end if
16:   end for
17: end while
```

Let $\mathbf{M}$ denote the coefficient matrix. $\mathbf{M}$ is of size $n \times k$ for decoding k symbols with n packets. To ease our discussion, we let $n = k$ for the time being. The case of $n > k$ will be discussed later. The task of decoding is essentially to solve

$$\mathbf{Mx} = \mathbf{y} \tag{2}$$

for $\mathbf{x}$ where $\mathbf{x}$ is the $k \times l$ vector of the symbols, $\mathbf{y}$ is the $n \times l$ vector of received packets, l is the length of a symbol, and $\mathbf{M}$ is defined over $GF(2)$.

Rather than the entire symbol set, our decoding procedure only needs to solve for a particular symbol, i.e. the borrowed symbol. This allows us to use a solver that deals with solving for only a single symbol with much lesser computational cost. The first task is to identify a collection of row vectors from $\mathbf{M}$ such that the row vectors eliminates all other symbols except the borrowed symbol. Let $\mathbf{x}'$ be a $k \times 1$ vector over $GF(2)$ describes the selection of row vectors, where $x'_j = 1$ indicates that the j^{th} row vector is selected and $x'_j = 0$ indicates otherwise. Let $\mathbf{u_i}$ be a $k \times 1$ unit vector where the unique 1 locates at the index i which is also the index of the borrowed symbol. We then need to solve the following

$$\mathbf{M_t x'} = \mathbf{u_i} \tag{3}$$

for $\mathbf{x}'$ where $\mathbf{M_t} = \mathbf{M^T}$. The fact that $\mathbf{M}$ is of full rank guarantees the existence of a solution. Finally, the inner product of $(\mathbf{x}', \mathbf{y})$ gives the borrowed symbol. We use the efficient Wiedemann algorithm [11] to solve (3). The vector $\mathbf{u_i}$ is first used to generate the so-called Krylov sequence

$$\mathbf{u_i}, \mathbf{M_t u_i}, \mathbf{M_t^2 u_i}, ..., \mathbf{M_t^k u_i}, ... \quad .$$

Let S be the space spanned by this sequence, where $\mathbf{M_t}$ acts in a nonsingular way on S. Let M_s be the operator $\mathbf{M_t}$ restricted to S, and define the minimal polynomial of M_s to be

$f(z)$, where the coefficient of $f(z) \in GF(2)$. Let $d = deg(f)$ and $f[i]$ be the coefficient of z^i in $f(z)$. Provided that $f(z)$ can be found, the solution to (3) can be quickly obtained by

$$\mathbf{x}' = \sum_{i=1}^{d} f[i]\mathbf{M_t^{i-1}u_i}. \tag{4}$$

Algorithm 2 shows how the borrowed symbol is obtained.

Algorithm 2 Solving the borrowed symbol

```
1: compute $\mathbf{M_t^i u_i}$ for $i = 0, 1, \ldots, 2k-1$
2: set $s = 0$ and $g_0(z) = 1$
3: set $\mathbf{u_{s+1}}$ to be the $(s+1)^{st}$ unit vector
4: extract from the results of step 1 the sequence $(\mathbf{u_{s+1}}, \mathbf{M_t^i u_i})_{i=0}^{2k-1}$(inner product)
5: apply $g_s(z)$ to this sequence
6: set $f_{s+1}$ to be the minimal polynomial of the sequence produced in step 5
7: update $g_{s+1} = f_{s+1} g_s$
8: set s = s + 1; if $deg(g_s) < k$ and $s < k$, go to step 3
9: use $f = g_s$ and compute $\mathbf{x}'$ as in (4)
10: the borrowed symbol is the inner product of $< \mathbf{x}', \mathbf{y} >$
```

The first few steps presented in Algorithm 2 are self explanatory. For step 5, given a polynomial $g(z)$ of degree d and a sequence $\{s_i\}_{i=0}^{l}$, let $s(z) = \sum_{i=0}^{l} s_i z^i$ and by definition the result of applying $g(z)$ to the sequence $\{s_i\}_{i=0}^{l}$ is $\{(g(z^{-1})s(z))[i]\}_{i=0}^{l-d}$. We apply the BM algorithm [12], [13] in step 6 to find the minimal polynomial. The readers can refer to [14] for a practical algorithm particularly designed for sequences defined in $GF(2)$.

D. Non-square Case

In practice, it is likely that more than k packets have to be received in order to ensure the complete recovery of all symbols. As a result, the coefficient matrix $\mathbf{M}$ will be non-square, i.e., the matrix is of size $n \times k$ with rank equal to k and $n > k$.

Our approach is to find a $n \times k$ matrix $\mathbf{M_c}$ such that $\mathbf{M_t M_c}$ will be of full rank. Please note that M_t should be of full rank, otherwise, all decoding methods fail. One way to obtain $\mathbf{M_c}$ is to randomly set an entry of row i in $\mathbf{M_c}$ to one with probability $p_i = \min\left(\frac{1}{2}, 2\log\frac{n}{k+1-i}\right)$. According to [12], with the probability of at least 12.5%, the $k \times k$ matrix $\mathbf{M_t M_c}$ is non-singular (i.e. full rank). In this way, with a few rounds of trials, we can find $\mathbf{M_c}$. As we now introduce $\mathbf{M_c}$, (3) becomes

$$\mathbf{M_t M_c x'} = \mathbf{u_i}. \tag{5}$$

Once $\mathbf{x}'$ is solved, the corresponding recovered symbol is obtained as $< \mathbf{M_c x'}, \mathbf{y} >$, where $< . >$ denotes the vector inner product.

III. Random Matrix Rank

It is understood that the probability of successful decoding for our proposed full rank algorithm is equal to the probability that the coefficient matrix $\mathbf{M}$ formed by the received packets is of full rank. In other words, as long as $\mathbf{M}$ is of full rank, our proposed algorithm guarantees the success of the decoding. Studying of the rank in a random matrix gives insight to the decoding performance of our proposed full rank algorithm.

A. Random matrix rank when $n = k$

We first consider the special case of $n = k$, where $\mathbf{M}$ becomes a $k \times k$ matrix over $GF(2)$. Let $\mathbf{v_i}$ be the i^{th} row vector of $\mathbf{M}$. The row vectors are linearly dependent if there exists a nonzero vector $(c_1, ..., c_k) \in GF(2)^k$ that satisfies $\sum_{i=1}^{k} c_i \mathbf{v_i} = \mathbf{0}$. If $\mathbf{M}$ is said to have a full rank, any linear combination of coefficient vectors $(\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k})$ will not produce $\mathbf{0}$. Consider a non-zero vector $\mathbf{c} = (c_1, \ldots, c_k) \in GF(2)^k$, with exactly q non-zero coordinates. Without loss of generality, let us assume that the first q coordinates of $\mathbf{c}$ are non-zero. Define P_q to be the probability that $\sum_{i=1}^{q} c_i \mathbf{v_i} = \mathbf{0}$. Clearly, it is the same as the probability that $\sum_{i=1}^{q-1} \mathbf{v_i} = \mathbf{v_q}$ because the operations are defined over $GF(2)$.

Suppose that summing the first q vectors resulting a vector with degree i, i.e. a vector that carries i number of 1s. We would like to analyze the conditions that its degree changes to j with one additional vector added. As illustrated in Fig. 1, we can compute the degree of the $(q+1)^{th}$ vector, $\mathbf{v_{q+1}}$, as

$$deg(\mathbf{v_{q+1}}) = a + b \tag{6}$$

where $b = j - i + a$ and $0 \le a \le i, 0 \le b \le (k-i)$. Thus, the probability that $\mathbf{v_{q+1}}$ of degree $(a+b)$ happens to be in the arrangement as revealed in Fig. 1 is $\frac{\binom{i}{a}\cdot\binom{k-i}{b}}{\binom{k}{a+b}}$.

Based on the above analysis, we can derive the state transition probability P_{ij}, i.e. the probability that the sum of $(q+1)$ vectors has degree j given that the sum of the first q vectors has degree i,

$$P_{ij} = \begin{cases} \displaystyle\sum_{\substack{0 \le a \le \min(k-j,i) \\ b=j-i+a}} \Omega_{a+b} \frac{\binom{i}{a}\binom{k-i}{b}}{\binom{k}{a+b}}, i < j \\ \displaystyle\sum_{\substack{1 \le a \le \min(k-j,i) \\ b=j-i+a}} \Omega_{a+b} \frac{\binom{i}{a}\binom{k-i}{b}}{\binom{k}{a+b}}, i = j \\ \displaystyle\sum_{\substack{i-j \le a \le \min(k-j,i) \\ b=j-i+a}} \Omega_{a+b} \frac{\binom{i}{a}\binom{k-i}{b}}{\binom{k}{a+b}}, i > j \end{cases} \tag{7}$$

where Ω_d is the probability that an encoded packet has a degree of d. This degree distribution is defined in LT codes governed by two other configuration parameters λ, δ [4].

The very last result describes the evolution of degree when an additional vector is added. This allows us to determine the degree distribution of the sum of any number of vectors. We shall define a transition matrix $\mathbf{Tr}$ with dimension $(k+1) \times (k+1)$, and each element $\mathbf{Tr}_{ij} = P_{(i-1)(j-1)}$. Let $\mathbf{\Omega^q}$ denotes

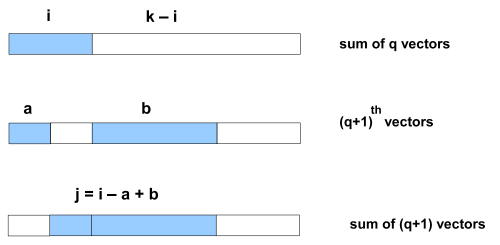


Fig. 1. An example of the degree changing from i to j with the $(q+1)^{th}$ vector added. Note that the dark region indicates the places of 1s.

the degree distribution of the sum of q vectors and $q \geq 1$, the following equation can be established

$$\mathbf{Tr}^{q-1} \cdot (\Omega_0, \Omega_1, \ldots, \Omega_k)^T = \mathbf{\Omega^q} \tag{8}$$

where $\mathbf{\Omega^q}$ can be determined numerically.

We now return our focus back on finding P_q, that is the probability that

$$\sum_{i=1}^{q-1} \mathbf{v_i} = \mathbf{v_q}. \tag{9}$$

In order to satisfy (9), the summation on the LHS must give a vector that has the same number of 1s as that of $\mathbf{v_q}$, and these 1s should be located at the same positions. Knowing that a vector of degree d can produce $\binom{k}{d}$ of possible forms, the probability for (9) to hold can be derived as

$$P_q = \sum_{d=0}^{k} \frac{\Omega_d \cdot \Omega_d^{q-1}}{\binom{k}{d}}. \tag{10}$$

If $\mathbf{M}$ is of full rank, then any combination of the coefficient vectors should not result in a linear dependency. Thus, the probability of full rank can be determined by

$$P_f(k) = \prod_{q=2}^{k} (1 - P_q)^{\binom{k}{q}} \tag{11}$$

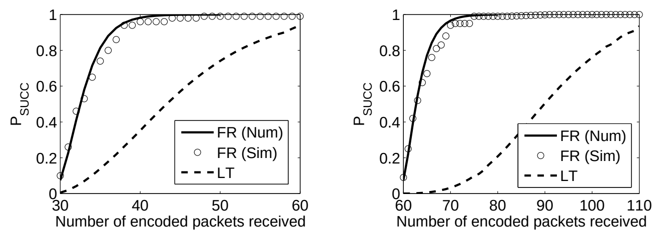
as there are $\binom{k}{q}$ of $\mathbf{c}$ with exactly q non-zero coordinates.

B. Random matrix rank when $n > k$

This subsection extends the case to the condition where $n > k$. Additional handling must be introduced since (11) is derived based on the condition that for a full rank matrix no linear dependency exists for any combination of the row vectors, which is not true for the case of $n > k$.

Let (q, r) denote the state of the coefficient matrix $\mathbf{M}$ where $\mathbf{M}$ consists of q row vectors with rank r, and $P(q, r)$ denote the corresponding state probability. Our objective is to derive the state probability of $P(n, k)$. The evolution of the rank of $\mathbf{M}$ can be described by the following recurrent relationship

$$P(q,r) = (1 - P_{or})P(q-1,r) + P_{o(r-1)}P(q-1,r-1)$$
$$1 < r \leq k, P(1,1) = 1, P_{ok} = 0$$
$$P(q,r) = 0, \text{ if } q < r \tag{12}$$



(a) $k = 30$ (b) $k = 60$

Fig. 2. The probability of success decoding for FR and LT with respect to the number of received packets.

where P_{or} is the probability that a random vector is linearly independent to r other linearly independent row vectors. It is easy to see that $P_{or} = \frac{P_f(r+1)}{P_f(r)}$.

With the establishment of the state recurrent relationship, the system state probabilities can then be determined by solving (13) (see page 5). Note that $(P(1,1), P(1,2), \ldots, P(1,k))^T$ is initialized to $(1, 0, 0, \ldots, 0)^T$. Since the transition matrix shown in (13) is of full rank, the methods like eigen decomposition or companion matrix and Jordan normal form [15] can be utilized to derive a close form expression for $P(q, r)$.

We set the number of input symbols to be $30, 60$, and present the numerical results for the probability of full rank versus the number of packets received for the setting $\lambda = 0.2, \delta = 0.1$ (λ and δ are the configuration parameters for encoding degree distribution). We further include simulation results to show the accuracy of our numerical computation. As can be seen from the results presented in Fig. 2, the probabilities of full rank climb quickly as soon as k packets are received. Comparing to the case of LT decoding algorithm [4], given a targeted success decoding probability, our full rank decoding algorithm uses much lesser received packets to decode the same number of symbols.

IV. Numerical Results and Discussion

Reference [6] and Section III provide the probabilities that all symbols can be successfully decoded after receiving a particular number of packets for LT and FR decoding algorithms respectively. Using these results, in Fig. 3, we compare the expected number of packets received for both algorithms. Four different set of configuration parameters are used, and the number of source symbols k ranges from 30 to 100. We compare with two sets of configuration parameters related to LT codes where both show consistent results.

From Fig. 3, it can be seen that our proposed full ranking decoding yields much better performance in terms of the expected number of packets needed for decoding. It can also be seen that the performance of our proposed algorithm is close to the ideal performance (i.e. $n = k$). Conversely, LT codes need additional packets to compensate for the imperfectness of received degree distribution to avoid the decoding process from terminating prematurely.

$$\begin{pmatrix} P(q,1) \\ P(q,2) \\ \vdots \\ P(q,k-1) \\ P(q,k) \end{pmatrix} = \begin{pmatrix} 1-P_{o1} & & & & \\ P_{o1} & 1-P_{o2} & & & \\ & P_{o2} & 1-P_{o3} & & \\ & & \ddots & \ddots & \\ & & & P_{o(k-1)} & 1-P_{ok} \end{pmatrix}^{(q-1)} \cdot \begin{pmatrix} P(1,1) \\ P(1,2) \\ \vdots \\ P(1,k-1) \\ P(1,k) \end{pmatrix} \tag{13}$$

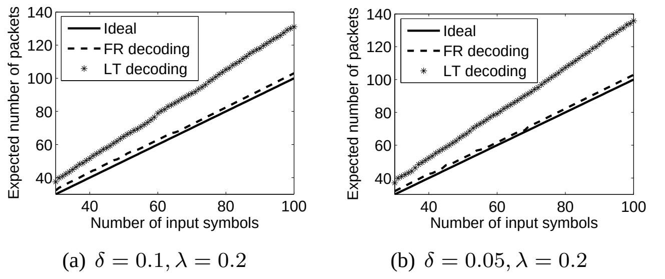


Fig. 3. The expected number of packets needed for recovering all input symbols.

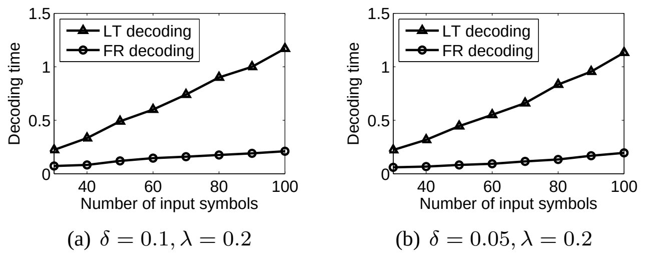


Fig. 4. Decoding runtime comparison between FR and LT decoding.

We further plot the benchmark the performance of both decoding schemes in Fig. 4. We develop both the decoding algorithms in Matlab and use tic/toc commands to record the runtime for decoding k symbols of 16 bits with 95% success probability. It can be seen our FR decoding requires much lower runtime for both configurations. While FR decoding requires additional steps to recover the borrowed symbols, it permits fewer input packets to decode which reduces the overall processing time. In addition, unlike traditional Wiedemann or Gaussian Elimination methods, the size of the coefficient matrix keeps on shrinking with each round of LT decoding passed.

V. Conclusion

In this paper, we first discussed the drawback of the LT codes when the number of symbols is small. We found that the existing methods addressing the drawback of the LT codes are either unscalable or computationally expensive for practical use. Our closer investigation on the LT decoding reveals that the LT decoding process often terminates prematurely even when other yet to be decoded symbols can be recovered by some other suitable decoding algorithm. We proposed a decoding algorithm named full rank decoding algorithm that is based on the LT decoding process with methods extending its decodability.

Our proposed algorithm ensures successful decoding when a full rank coefficient matrix is received. We analyzed the rank behavior of a random coefficient matrix which shows that a full rank can be achieved with low overheads in terms of packets. This result encouraged the design for decoding algorithms like ours that achieve successful decoding upon a full rank. In our design, we introduced the concept of borrowed symbols in the LT decoding process, where when LT decoding process terminate prematurely, a customized Wiedemann method is activated to recover a new symbol and place into the ripple so that the LT decoding process can continue. Our results have shown a lower expected number of packets needed for the recovery of all symbols and a lower runtime.

References

[1] J.W. Byers, M. Luby, M. Mitzenmachert, A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data,” *ACM SIGCOMM Computer Communication Review*, Vol. 28, no. 4, pp. 56-67, 1998.

[2] R.G. Gallager, “Low-Density Parity-Check Codes”. Cambridge, MA: MIT Press, 1963.

[3] I.S. Reed, G. Solomon, “Polynomial Codes over Certain Finite Fields,” *Journal of the Society for Industrial and Applied Mathematics*, Vol. 8, no. 2, pp. 300-304, Jun, 1960.

[4] M. Luby, “LT Codes,” *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 271-280, Nov, 2002.

[5] A. Shokrollahi, “Raptor Codes,” *IEEE Transactions on Information Theory*, Vol. 52, no. 6, pp. 2551-2567, 2006.

[6] R. Karp, M. Luby, A. Shokrollahi, “Finite length analysis of LT codes,” *The IEEE International Symposium on Information Theory*, 2004.

[7] E. Hyytia, T. Tirronen, J. Virtamo, “Optimal Degree Distribution for LT Codes with Small Message Length,” *The 26th IEEE International Conference on Computer Communications INFOCOM*, pp. 2576-2580, 2007.

[8] C. Studholme, I. Blake, “Windowed Erasure Codes,” *The IEEE International Symposium on Information Theory*, pp. 509-513, 2006.

[9] J. Gentle, “Numerical Linear Algebra for Application in Statistics,” pp. 87-91, Springer-Verlag, 1998.

[10] M. A. Shokrollahi, S. Lassen, and R. Karp, ”Systems and processes for decoding a chain reaction code through inactivation,” U.S. Patent 7,030,785. April 2006.

[11] D. Wiedemann, “Solving sparse linear equations over finite fields,” *IEEE Transactions on Information Theory*, Vol. 32, no. 1, pp. 54-62, 1986.

[12] E. Berlekamp, “Algebraic Coding Theory,” McGraw-Hill, New York, 1968.

[13] J. Massey, “Shift-register synthesis and BCH decoding,” *IEEE Transactions on Information Theory*, Vol. 15, no. 1, pp. 122-127, 1969.

[14] G. Gong, “Lecture Notes on Sequence Design,” Chp. 6, http://comsec.uwaterloo.ca/ ggong/CO739x/course.html.

[15] R.A. Horn, C.R. Johnson, “Matrix Analysis,” Cambridge University Press, 1985.

[16] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, “Wireless Sensor Networks: a Survey,” *Computer Networks*, Vol. 38, no. 4, pp. 393-422, 2002.