Network and Communication Technologies; Vol. 2, No. 2; 2013
ISSN 1927-064X E-ISSN 1927-0658
Published by Canadian Center of Science and Education

# A Fast Raptor Codes Decoding Strategy for Real-Time Communication Systems

Yeqing Wu[1], Fei Hu[1], Qingquan Sun[1], Ke Bao[1] & Mengcheng Guo[1]

[1] Electrical and Computer Engineering, University of Alabama, Tuscaloosa, AL, USA

Correspondence: Yeqing Wu, Electrical and Computer Engineering, University of Alabama, Tuscaloosa, AL, USA. E-mail: ywu40@crimson.ua.edu

Received: September 17, 2013 Accepted: October 15, 2013 Online Published: November 18, 2013

doi:10.5539/nct.v2n2p29 URL: http://dx.doi.org/10.5539/nct.v2n2p29

**Abstract**

We propose an efficient algorithm for Raptor decoding, which reduces the computational complexity of the most time-consuming steps in systematic decoding. Our proposed algorithm includes two aspects: First, to handle the decoding failure of the Raptor decoding, we propose a scheme, which is called the No-Wrapup Failure Handling scheme. It can resume the decoding process from where it fails after receiving a pre-defined number of additional encoded symbols, and thus avoids the repetition of time-consuming steps in the decoding process. Second, in order to reduce the time of finding the row with the minimum degree in the precode, we propose a Fast Min-Degree Seeking (FMDS) scheme. FMDS automatically maintains and updates the row degrees of the precode when converting the precode into an identity matrix through Gaussian elimination and Belief-propagation. Experimental results show that, compared to other Raptor decoding schemes, the proposed scheme achieves a much shorter decoding time, and can greatly speed up the data recovery in real-time applications.

**Keywords:** Raptor codes, LT codes, Digital Fountain codes, FEC

## 1. Introduction

Recently, digital Fountain codes have attracted significant attention. Digital Fountain codes are rateless and provide forward error correction (FEC) to address the problem of losing packets in erasure channels under any channel condition. They can produce infinite number of encoded symbols from the source symbols without any knowledge of the channel, which makes them very suitable for wireless communications. Luby (2002) developed the first practical class of rateless codes, called Luby Transform (LT) codes. Shokrollahi (2006) further extended the LT codes to Raptor codes, which are a class of powerful rateless codes since they need very small overhead to completely recover the source data with linear encoding/decoding time.

Due to their high recovery rate and low time complexity, Raptor codes have been included in the Third Generation Partnership Project (3GPP) multimedia broadcast/multicast services (MBMS) (3GPP, 2007) and Digital Video Broadcast-Handheld (DVB-H) (DVB, 2004) standards. Detailed description of Raptor codes can be found in (Shokrollahi, 2006; 3GPP, 2007; DVB, 2004; IETF, 2007). In (Cataldi, Shatarski, Grangetto, & Magli, 2006), the performance of LT codes and Raptor codes was compared for multimedia applications. In (Hussein, Oka, & Lampe, 2008), a scheme for the early termination of decoding was proposed to avoid unnecessary message updates and stop a decoding attempt within a fraction of an iteration. The performance of Raptor codes on arbitrary binary input memoryless symmetric channels was investigated in (Etesami & Shokrollahi, 2006). Maximum-likelihood decoding of Raptor codes over the binary erasure channel was discussed in (Kim & Chung, 2008). The problem of Raptor codes design for binary-input additive white Gaussian noise was studied in (Cheng, Castura, & Mao, 2009). Alexiou, Bouras and Papazois (2010) presented a complete study of the impact of Raptor codes in mobile multicast transmission and Investigated the parameters that affect the optimal FEC code selection. In (Chen, Zhang, Lou, & Chen, 2012), an adaptive code symbol assignment scheme based on Raptor codes was proposed for the secondary user (SU) in a multichannel cognitive radio to maximize the throughput of SU. The permeable layer receiver (PLR) and the individual post-repair mechanism with Raptor codes, in order to enhance the system performance for the MBMS receiver, are studied in (Gasiba, Xu, & Stockhammer, 2008). In (Bouras, Kanakis, Kokkinos, & Papazois, 2012), the Raptor codes overhead requirements with different network conditions in 3GPP LTE MBMS streaming services were investigated.

Raptor decoding may fail even when the number of received encoded symbols is larger than the number of source symbols (Shokrollahi, 2006; Shokrollahi & Luby, 2011; Stockhammer, Shokrollahi, Watson, Luby, & Gasiba, 2008). The 3GPP MBMS standard (3GPP, 2007) proposes an efficient implementation of Raptor decoding. However, when the decoding fails, Raptor decoder needs to restart the decoding process after collecting enough encoded symbols. Recently, there have been a few other studies (Mladenov, Nooshabadi, & Kim, 2011; Shi, Yang, & Zhang, 2011) that provide implementation and improvements of the 3*GPP* decoding algorithm. In (Mladenov, Nooshabadi, & Kim, 2011), a hardware implementation of the Raptor decoding algorithm is discussed. In (Shi, Yang, & Zhang, 2011), an algorithm is proposed to handle the Raptor decoding failure in phase 2, and thus avoid the repetition of phase 1. In this paper, we consider the case when the decoding process fails and provide a solution to handle this problem.

### *1.1 Contributions of the Proposed Scheme*

The most time-consuming part of the Raptor decoding scheme in 3GPP MBMS standard is the transformation process for converting the precode into an identity matrix using Gaussian elimination (GE) and Belief-propagation (BP) (3GPP, 2007). The transformation process consists of four phases as discussed in Section 2.4. We propose the following two approaches to significantly reduce the computation time of phases 1 and 2 of the transformation process, in order to improve the Raptor decoder efficiency. *(i)* For fast recovery from the decoding failure in phases 1 and 2, we propose a no-wrapup failure handling (NWFH) scheme in Section 3.1. The NWFH scheme resumes the decoding process after receiving a pre-defined number of additional encoded symbols, instead of restarting the decoding process from scratch in (3GPP, 2007). Thus, our scheme significantly reduces the decoding time because phase 1 consumes about 90% of the entire decoding time (Mladenov, Nooshabadi, & Kim, 2011). Note that the decoding algorithm in (Shi, Yang, & Zhang, 2011) handles the decoding failure in phase 2, while assuming that the decoding process does not fail in phase 1. *(ii)* For second improvement, we propose a fast min-degree seeking (FMDS) scheme in Section 3.2 to maintain and update the row degrees of matrix in each iteration of phase 1 automatically, instead of recalculating it. The FMDS scheme thus saves time spent in recalculating them at each iteration.

The paper is organized as follows. Section 2 provides a brief discussion of the systematic Raptor codes used in our schemes. Section 3 describes our proposed NWFH and FMDS schemes for Raptor decoding. In Section 4, we compare the decoding performance of our proposed NWFH and FMDS schemes with other schemes, followed by conclusions in Section 5.

## 2. Principles of Systematic Raptor Codes

The detailed description of Raptor codes can be found in (Shokrollahi, 2006; Shokrollahi & Luby, 2011). In this section, we briefly describe the mathematical model of the systematic Raptor codec used in 3GPP MBMS (3GPP, 2007).

### *2.1 Systematic Raptor Codes*

Raptor codes encode one block of source symbols at a time, where the size of each source symbol $\ell$ is suggested to be within 1 to 1024 bytes in (3GPP, 2007). Different blocks may have different number (denoted as $K$) of source symbols. Raptor code consists of two parts: a precode $A$ as the outer code and the LT code as the inner code, where $A$ is a binary matrix consisting of a low-density parity check (LDPC) generation matrix ($G_{LDPC}$), a high-density parity check generation matrix ($G_{Half}$) and a LT generation matrix ($G_{LT}$). Table 1 summarizes the symbols used in this paper.

### *2.2 Raptor Encoder*

It includes the following two steps (3GPP, 2007):

**Step a)** The precode $[A]_{L\times L}$ has two functions. First, the constraint matrices of $A$, $G_{LDPC}$ and $G_{Half}$, ensure that each source symbol is encoded at least once, which enables the decoder to recover every source symbol. Second, $[A]_{L\times L}$ contains the first $K$ rows of the LT generation matrix $[G_{LT}]_{N\times L}$. As a result, the first $K$ output symbols of LT encoder correspond to the $K$ source symbols. This makes the overall Raptor code systematic.

The Raptor encoder first encodes the $K$ source symbols $S[j]$, $j = 0, \cdots, K-1$ into $L = K + S + H$ intermediate

Table 1. Main parameters of raptor codes

| Symbols | Definitions |
|---|---|
| $K$ | The number of source symbols |
| $S$ | The number of LDPC symbols |
| $H$ | The number of Half symbols |
| $L$ | The number of intermediate symbols: $L = K + S + H$ |
| $\varepsilon$ | Encoding overhead in percentage |
| $N$ | The number of encoded symbols: $N = K(1 + \varepsilon)$ |
| $N'$ | The number of received encoded symbols |
| $M$ | The number of rows in the precode $A$: $M = K + S + N'$ |
| $A$ | Pre-coding matrix with dimension $L \times L$ in Raptor encoder and $M \times L$ in Raptor decoder |
| $ESI$ | Encoding Symbol ID |
| $S[i]$ | The $i^{th}$ source symbol for $i = 1, \cdots, K$ |
| $C[i]$ | The $i^{th}$ intermediate symbol for $i = 1, \cdots, L$ |
| $E[i]$ | The $i^{th}$ encoded symbol for $i = 1, \cdots, N$ |
| $E'[i]$ | The $i^{th}$ encoded symbol for $i = 1, \cdots, N'$ |
| $D[i]$ | Zero-valued symbols for $i = 0, \cdots, S+H-1$, received encoded symbols for $i = S + H, \cdots, M$ |
| $c[i]$ | The original row index of current $C[i]$ before column exchange of $A$ |
| $d[i]$ | The original row index of current $D[i]$ before row exchange of $A$ |

symbols $C[i]$, $i = 0, \cdots, L-1$ by using the precode $A$:

$$\underbrace{\begin{bmatrix} C[0] \\ \vdots \\ C[L-1] \end{bmatrix}}_{[C]_{L\times 1}} = \underbrace{\begin{bmatrix} [G_{LDPC}]_{S\times K} & I_{S\times S} & \emptyset_{S\times H} \\ [G_{Half}]_{H\times(K+S)} & & I_{H\times H} \\ & [G_{LT}]_{K\times L} & \end{bmatrix}^{-1}}_{[A]_{L\times L}} \bullet \underbrace{\begin{bmatrix} \emptyset_{S\times 1} \\ \emptyset_{H\times 1} \\ S[0] \\ \vdots \\ S[K-1] \end{bmatrix}}_{[S_R]_{L\times 1}} \tag{1}$$

The intermediate symbols $C[i]$ generated by $G_{LDPC}$ and $G_{Half}$ are called redundant symbols.

**Step b)** LT encoder is used on the intermediate symbols to generate the $N \geq K$ encoded symbols $E[i]$, $i = 0, \cdots, N-1$, as follows. Each encoded symbol is associated with an encoding symbol ID $(ESI)_i$.

$$\underbrace{\begin{bmatrix} E[0] \\ \vdots \\ E[N-1] \end{bmatrix}}_{[E]_{N\times 1}} = \underbrace{\begin{bmatrix} g_{11} & \cdots & g_{1L} \\ \vdots & \ddots & \vdots \\ g_{N1} & \cdots & g_{NL} \end{bmatrix}}_{[G_{LT}]_{N\times L}} \bullet \underbrace{\begin{bmatrix} C[0] \\ \vdots \\ C[L-1] \end{bmatrix}}_{[C]_{L\times 1}} \tag{2}$$

The submatrix $[G_{LT}]_{K\times L}$ of $A$ is included as the first $K$ rows of $[G_{LT}]_{N\times L}$. This makes the first $K$ encoded symbols correspond to the $K$ source symbols, that is, $E[i] = S[i]$, for $i = 0, \cdots, K-1$.

*2.3 Raptor Decoder*

Raptor decoder includes the following two steps (3GPP, 2007):

**Step a)** After receiving the $N'$ encoded symbols ($N' \geq K$), a precode $[A]_{M\times L}$ is constructed, where $M = N' + S + H$, to recover the intermediate symbols $C[j]$, $j = 0, \cdots, L-1$ as shown in 3. The first $S + H$ rows in $[A]_{M\times L}$ are the same as in $A$ in 1, and $[G_{LT}]_{N'\times L}$ is generated using the ESI values of the $N'$ received encoded symbols. The rows

in $[G_{LT}]_{N\times L}$ corresponding to the lost encoded symbols are excluded from $[G_{LT}]_{N'\times L}$.

$$\underbrace{\begin{bmatrix} [G_{LDPC}]_{S\times K} & I_{S\times S} & \emptyset_{S\times H} \\ \left[G_{Half}\right]_{H\times(K+S)} & & I_{H\times H} \\ & [G_{LT}]_{N'\times L} & \end{bmatrix}}_{[A]_{M\times L}} \bullet \underbrace{\begin{bmatrix} C[0] \\ \vdots \\ C[L-1] \end{bmatrix}}_{[C]_{L\times 1}} = \underbrace{\begin{bmatrix} \emptyset_{S\times 1} \\ \emptyset_{H\times 1} \\ E'[0] \\ \vdots \\ E'[N'-1] \end{bmatrix}}_{[D]_{M\times 1}} \tag{3}$$

**Step b)** After obtaining the intermediate symbols, the $K$ source symbols $S[j]$ for $j = 0, \cdots, K-1$ are recovered using the $K \times L$ LT generator matrix:

$$\underbrace{\begin{bmatrix} S[0] \\ \vdots \\ S[K-1] \end{bmatrix}}_{[S]_{K\times 1}} = \underbrace{\begin{bmatrix} g_{11} & \cdots & g_{1L} \\ \vdots & \ddots & \vdots \\ g_{N1} & \cdots & g_{NL} \end{bmatrix}}_{[G_{LT}]_{K\times L}} \bullet \underbrace{\begin{bmatrix} C[0] \\ \vdots \\ C[L-1] \end{bmatrix}}_{[C]_{L\times 1}} \tag{4}$$

Source symbols are decoded directly from the corresponding received encoded symbols with ESI ¡ $K$. The missing source symbols are decoded from the recovered intermediate symbols $C$ using (4). To obtain $C$ from (3), the algorithm requires that $[A]_{M\times L}$ over $GF(2)$ has a full column rank (Shokrollahi, 2006; Shokrollahi & Luby, 2011). Note that all the source symbols may not be recovered even when ($N' \geq K$). When ($N' < K$), the systematic Raptor decoder cannot decode all the intermediate and source symbols. Instead, it directly recovers the source symbols from the corresponding received encoded symbols whose ESI ¡ $K$.

### *2.4 3GPP MBMS Decoding Algorithm*

The decoding algorithm of 3GPP MBMS uses GE and BP to transform $A$ to a $L \times L$ identity matrix after discarding its last $M - L$ rows. Since the computation efficiency of GE depends heavily on the sparsity of $A$, the BP operation is performed to maintain its sparsity. This transformation process can be summarized into four phases as shown in Figure 1 (3GPP, 2007).

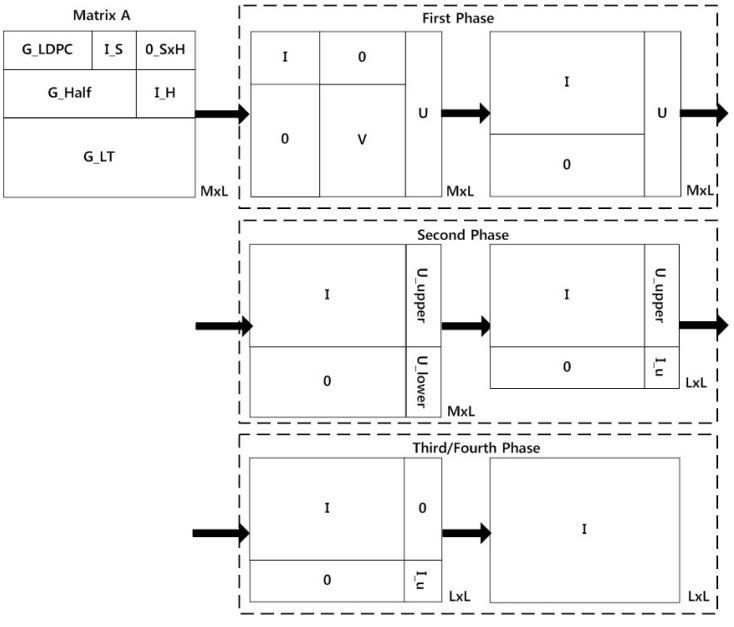



Figure 1. Illustration of four phases to transform matrix A into an identity matrix

In phase 1, the matrix $A$ is converted to three submatrices: $I$, zero submatrix, and $U$ as shown in Figure 1. In this figure, $V$ is an intermediate matrix. In the beginning of phase 1, $V = A$ and $I$ and $U$ are Null matrices. For each iteration, a row of $V$, with the minimum non-zero degree $r$, is chosen, where the degree of a row is defined as the number of elements '1' it has. Then, the chosen row is exchanged with the first row in $V$. All the columns of $V$ are reordered so that the first and the last $r-1$ elements of the first row are '1'. Then, the other rows which have ones in their first column are exclusive-ORed with the first row so that their columns become '0'. The rank of submatrix $I$ is increased by 1 and the number of columns in $U$ is increased by $r-1$. This process is repeated until the sum of columns of $I$ and $U$ is equal to $L$. Phase 1 fails when no non-zero row remains in $V$ before $V$ disappears, that is, the sum of columns of $I$ and $U$ is less than $L$ and elements in all rows of $V$ are zero.

In phase 2, the submatrix $U$ is partitioned into $U_{upper}$ and $U_{lower}$, where $U_{upper}$ consists of the first $i$ rows and $U_{lower}$ contains the remaining $M-i$ rows. If the rank of $U_{lower}$ is less than $u$, the decoding of phase 2 fails. Otherwise, GE is performed on $U_{lower}$ to transform it into an identity matrix with rank $u$. Then, the last $M-L$ rows of $[A]_{M\times L}$ are discarded.

In phase 3, a precomputation matrix $U'$ with $ceil(u/8) \times 255$ rows and $u$ columns, is generated based on $I_u$. In phase 4, the matrix $U'$ is used to zero out $U_{upper}$ and thus converts $A$ into the $L \times L$ identity matrix (please refer to (3GPP, 2007) and (IETF, 2007) for details).

Successful decoding depends on the transformation process discussed above. The intermediate symbols $C$ can be decoded in the meantime, based on the row operations and row/column exchanges occurring during the transformation process. Let $c[i]$ and $d[i]$, $i = 0, \cdots, L-1$, record the original row index of the $i^{th}$ row in $C$ and $D$, respectively. Initialize $c[0] = 0, \cdots, c[L-1] = L-1$, and $d[0] = 0, \cdots, d[M-1] = M-1$. The relationship between the transformation process and decoding of $C$ is summarized in Table 2:

Table 2. Relationship between the GE and decoding operations

| GE operation | Decoding operation |
|---|---|
| $A[i'] = A[i] \oplus A[i']$, where $A[i]$ represents the $i^{th}$ row of $A$ | $D[d[i']] = D[d[i]] \oplus D[d[i']]$ |
| Exchange $i^{th}$ and $i'^{th}$ rows of $A$ | Exchange the values of $d[i]$ and $d[i']$ |
| Exchange $i^{th}$ and $i'^{th}$ columns of $A$ | Exchange the values of $c[i]$ and $c[i']$ |

At the end of successful transformation and decoding processes, it is clear that $C[c[i]] = D[d[i]]$, for $i = 0, \cdots, L-1$, and thus the intermediate symbol vector $C$ is recovered.

## 3. Proposed Fast Raptor Decoding Scheme

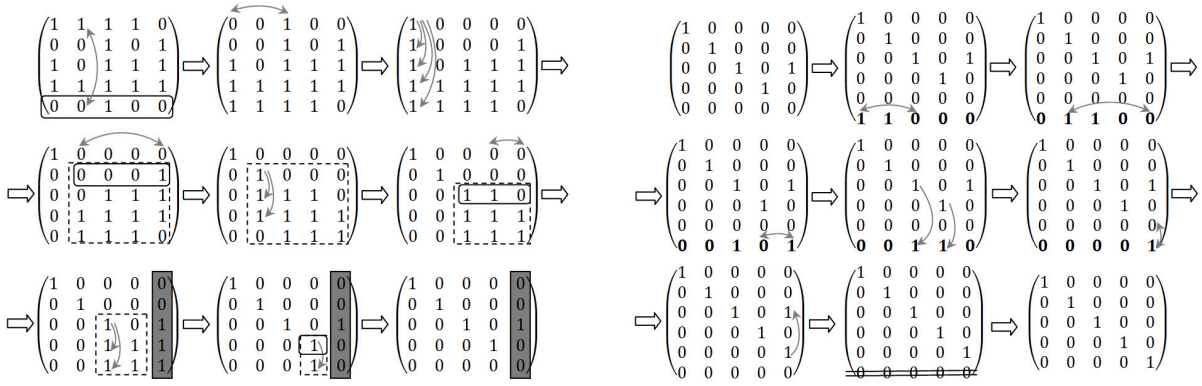
In this section, we describe our fast algorithm that makes two improvements to reduce the computation time in the phases 1 and 2 of the Raptor decoding algorithm used in 3GPP MBMS standard (3GPP, 2007). These improvements are: *(i)* The NWFH scheme for efficient handling of decoding failure in phases 1 and 2, and *(ii)* The FMDS scheme for efficient computing of the row degrees that are needed for each iteration in phase 1.

### *3.1 No-Wrapup Failure Handling Scheme*

As described in Section 2.4, the 3GPP MBMS decoding process fails *(i)* in phase 1, when there is no non-zero row to choose from before $V$ disappears, and *(ii)* in phase 2, when the rank of $U_{lower}$ ¡ $u$. When the decoding process fails in the first two phases, it needs to receive enough new encoded symbols to restart decoding from the beginning. However, phases 1 and 2 consume more than 90% decoding time of the entire Raptor decoding process. In order to significantly reduce the decoding time, we propose a new NWFH scheme, which stores the indices of the columns that are involved in the exchanges during the GE process. When the $i^{th}$ and $i'^{th}$ columns of $A$ are exchanged, we record $i$ and $i'$ as an index pair. When the decoding fails in phases 1 or 2, we append $n$ additional received encoded symbols to $D$ in (3), and append $n$ new rows of generation matrix $[G_{LT}]_{n\times L}$ (based on ESI values of those received encoded symbols) to the bottom of the matrix $A$. Our algorithm exchanges the columns based on the recorded indices for the submatrix $[G_{LT}]_{n\times L}$. After this, each row of the updated submatrix $[G_{LT}]_{n\times L}$ is *XORed* with each row of the identity submatrix $I$ in $A$. Therefore, we can achieve the same results of original GE and BP operations without restarting from the beginning. Finally, the standard GE and BP are performed to turn the

updated $A$ into an identity matrix from the location of decoding failure.

Figure 2(a) shows the basic ideas of GE and BP operations for a matrix. Figure 2(b) shows our proposed algorithm to handle the decoding failure in phase 2. We denote the row exchange as $\updownarrow$, column exchange as $\leftrightarrow$, and row XOR as $\downarrow\uparrow$, respectively. The rectangles with dash and solid borders represent the submatrix $V$ and the chosen row with the minimum degree, respectively. The rectangles with grey color denote the submatrix $U$. The text with bold font highlights the new appending row of the generation matrix $[G_{LT}]_{1\times 5}$ (based on ESI values of the received encoded symbols).



(a) (b)

Figure 2. (a) Example of GE and BP operations on matrix; (b) Proposed NWFH algorithm to handle the failure in phase 2

During the GE and BP operations, we store the indices of the columns that are involved in the exchanges in Figure 2(a). From the last matrix in Figure 2(a), we can see that the decoding process fails in phase 2 because the rank of $U_{lower}$, denoted as $rank(U_{lower})$, where $U_{lower}$ is the last row of submatrix $U$ in this example, is equal to 0 which is less than the number of columns in $U$ ($u = 1$). If the decoding fails, we append a new row (based on the ESI value of the encoded symbol) to the matrix. In Figure 2(b), the new added row is marked with bold font. We exchange the columns of the new appended row based on the recorded column indices in Figure 2(a). Then the rows of identity matrix $I$ is XORed with this new row to eliminate the ones of the new row in columns 0 to $i-1$. After these operations, we obtain the final identity matrix.

In Raptor codes, the probability $P_e(\cdot)$ of decoding failure for a source block (i.e., the probability of at least one source symbol in the source block not being recovered) can be estimated as a function of $K$ and $N$ (Stockhammer, Shokrollahi, Watson, Luby, & Gasiba, 2008):

$$P_e(\varepsilon(K)) = \begin{cases} 1 & \text{if } \varepsilon(K) < 0 \\ 0.85 \times 0.567^{\varepsilon(K)} & \text{if } \varepsilon(K) \geq 0 \end{cases} \tag{5}$$

The average extra overhead $\overline{R}_\varepsilon(K)$ needed for successful decoding after each failure is (Stockhammer, Shokrollahi, Watson, Luby, & Gasiba, 2008):

$$\begin{aligned} \overline{R}_\varepsilon(K) &= \frac{1}{K}\sum_{i=0}^{\infty} i \cdot (P_e(i-1) - P_e(i)) \\ &= \frac{0.85}{K}\sum_{i=0}^{\infty} i \cdot (0.567^{i-1} - 0.567^{i}) \\ &= \frac{0.85}{(1-0.567)K} \approx \frac{2}{K} \end{aligned} \tag{6}$$

We can see that the average number of additional encoded symbols required for a source block with size $K$ is $K \times \frac{2}{K} = 2$, which is independent of $K$. Due to this reason, when $N' \geq K$ and the decoding process fails, the proposed algorithm continues decoding after receiving $n = 2$ additional encoded symbols.

In other Raptor decoding schemes (3GPP, 2007; IETF, 2007; Mladenov, Nooshabadi, & Kim, 2011; Shi, Yang, & Zhang, 2011), the decoding process cannot begin until at least $N'$ encoded symbols ($N' \geq K$) are received. Instead, in our proposed NWFH algorithm, the decoding can begin when $N' < K$. Meanwhile, our algorithm continues to receive additional encoded symbols until $N' \geq K$. This reduces the decoding time.

*3.2 Fast Minimum Row Degree Seeking Scheme*

The computing load in phase 1 is high for a large matrix $A$ because the number of 1s are counted in each row in order to find out the row with the minimum degree $r$. After the minimum degree row is found, the row and column exchanges are executed to make the upper-left submatrix an identity matrix.

Assuming that $A$ is a $M \times L$ matrix and $A$ is represented by $V$, $L$ operations are needed to count the number of 1s in each row in the first iteration. If we ignore the row and column permutations, $(M-1) \times L$ operations will be needed to find out the row with the minimum degree of $V$ in the first iteration. In the next iteration, the same process will be repeated for the remaining $(M-1)$ rows and $(L-1)$ columns, which needs $(M-2) \times (L-1)$ operations. This process goes on until $V$ disappears. In Figures 1 and 2, the ones in the columns that are merged into submatrix $U$ are not considered when we calculate the row degree for subsequent iterations. Since the number of columns in $U$ is small compared to the column rank of matrix $A$ because of the sparsity nature of matrix $A$, we can ignore their contribution to the time complexity of phase 1. In the worse case, the time complexity of phase 1 is:

$$\begin{aligned} T &= (M-1) \times L + (M-2) \times (L-1) + \cdots + (M-L) \times 1 \\ &= O(L^2 M) \end{aligned} \tag{7}$$

In (7), the time complexity of GE and BP operations in phase 1 is a cubic of the dimension $(L, M)$. Note that phase 1 occupies more than 90% of the total decoding time.

In order to further reduce the decoding time of the systematic Raptor codes, we propose a new FMDS scheme to modify phase 1 so that we do not need to count 1s in each row for every iteration. In our proposed FMDS algorithm, we calculate and record the degree of each row in matrix $V$, and then iteratively update the degree of each row during the transformation process as discussed below. After the row with the minimum degree $r$ in submatrix $V$ is found and is exchanged into the first row of $V$, all the columns of $V$ are reordered so that the first and the last $r-1$ elements of the first row are '1'. Then, the other rows which have ones in their first column are exclusive-ORed with the first row so that their first columns become '0' and their degree is reduced by one. The last $(r-1)$ columns are merged into $U$, and the degree of rows whose elements in these columns are 1 are updated, instead of counting them as in (3GP, 2007). The row with the minimum degree is selected for the next iteration.

With our FMDS scheme, we only need to update the degree of a row in these two cases. It can easily maintain the correct degree of a row and do not need to count the ones in each row when finding the row with the minimum degree in each iteration.

In phase 1, the matrix $A$ is converted to three submatrices: $I$, zero submatrix, and $U$ as shown in Figure 1. In this figure, $V$ is an intermediate matrix. In the beginning of phase 1, $V = A$ and $I$ and $U$ are Null matrices. For each iteration, a row of $V$, with the minimum non-zero degree $r$, is chosen, where the degree of a row is defined as the number of elements '1' it has. Then, the chosen row is exchanged with the first row in $V$. All the columns of $V$ are reordered so that the first and the last $r-1$ elements of the first row are '1'. Then, the other rows which have ones in their first column are exclusive-ORed with the first row so that their first columns become '0'. The rank of submatrix $I$ is increased by 1 and the number of columns in $U$ is increased by $r-1$. This process is repeated until the sum of columns of $I$ and $U$ is equal to $L$. Phase 1 fails when no non-zero row remains in $V$ before $V$ disappears, that is, the sum of columns of $I$ and $U$ is less than $L$ and elements in all rows of $V$ are zero.

Figures 3(a) and 3(b) show scheme of (Mladenov, Nooshabadi, & Kim, 2011) and our FMDS scheme, respectively, to seek the minimum degree row. The digits with grey background in 'Degree' column indicate that this row already has the minimum degree and will not be considered in the next iteration. The symbol '?' in 'Degree' column in Figure 3(a) means that it needs to recount the number of ones in these rows to get the degree after the row/column operations. The meaning of the other symbols are the same as in Section 3.1.

In Figure 3(a), after finding the minimum-degree row and performing the row/column permutation, the method in (Mladenov, Nooshabadi, & Kim, 2011) needs to recount the number of ones in each row of submatrix $V$ for the next iteration. On the other hand, with our FMDS scheme in Figure 3(b), we only need to update the degree of a

row in the following two cases: 1) The row is *XORed* with the minimum-degree row; 2) The column is merged into the submatrix $U$ (marked with the rectangles with grey color) and has one in this column. This is because the one in this column will not be counted in the degree of the row for subsequent iterations.

### *3.3 Overall Algorithm*

The pseudocode of the proposed scheme is given in Algorithm 1. Here, we only show our improved phases 1 and 2 in the decoding process of $C$. The symbols used in the pseudocode have been explained in Tables 1 and 3, and Figure 1.

Table 3. Additional symbols defined in algorithm

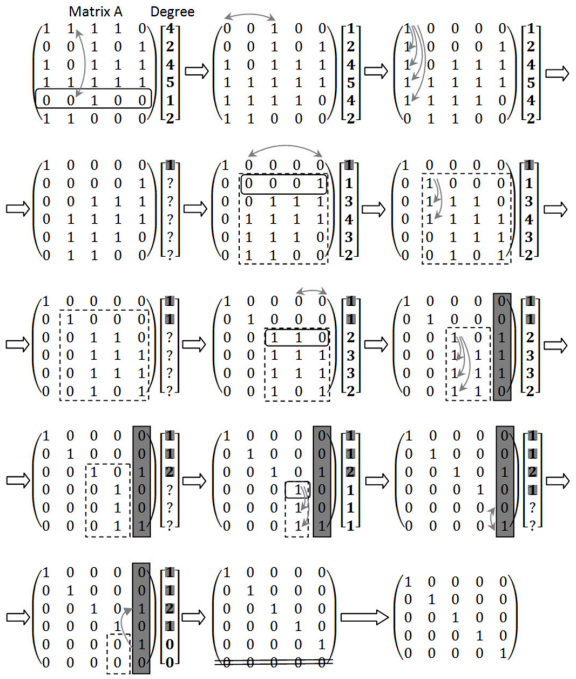
| Symbol | Definition |
|---|---|
| $\leftrightarrow$ | Operation that exchanges the values of two associated operands |
| $zeroRowNum$ | The number of zero-rows in $V$ |
| $i$ | The rank of identity matrix $I$ |
| $u$ | The number of columns in matrix $U$ |
| $A[m]$ | For $m = 0, \cdots, M-1$, the $m^{th}$ row in $A$ |
| $A(l)$ | For $h = 0, \cdots, L-1$, the $l^{th}$ column in $A$ |
| $degree[s]$ | For $s = i, \cdots, M-1$, records the degree in $V$ of the $i^{th}$ row in $A$ |
| $r$ | Minimum non-zero degree in $V$ of rows in $A$ |
| $R_r$ | The index of the selected row in $A$ with the minimum degree $r$ in $V$ |
| $colExPos$ | It records the indexes of the column exchanged during GE process |
| $onesInLastCols[k]$ | The number of ones in the last $r-1$ elements of $A[k]$. Here $k = i+1, \cdots, M-1$ |
| $addESI$ | $n \times 1$ vector that contains ESI values of the $n$ additional received encoded symbols |

## 4. Experimental Results

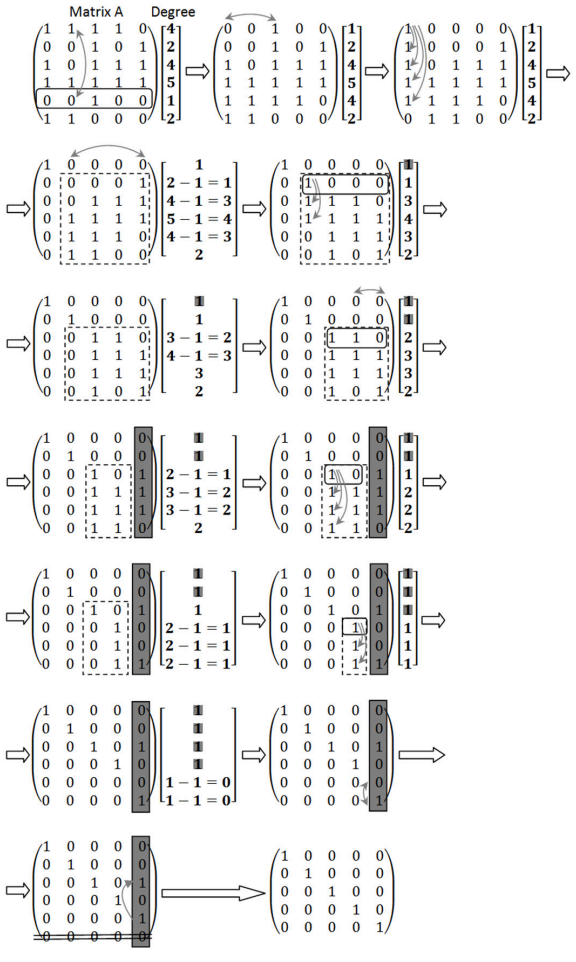
We compare the Raptor decoding computation time achieved by our proposed fast schemes with those in 3GPP MBMS (3GPP, 2007) and (Mladenov, Nooshabadi, & Kim, 2011; Shi, Yang, & Zhang, 2011). The Raptor encoder and decoder are implemented using Visual $C++$ on $Intel® Core^{TM}2$ Duo CPU $T6670@2.20GHz$, $3GB$ RAM. The decoding of the intermediate symbols occurs in the same time as the transformation process. That is, each time an operation in GE and BP occurs, the corresponding operation in the decoding process of the intermediate symbol vector $C$ also occurs.

### *4.1 Phase 1 Decoding Time*

As explained in the previous section, our algorithm differs from previous decoding algorithms in phases 1 and 2 of the transformation process, including the time consumed by the corresponding decoding operations of $C$. The computation times of phase 1 and the overall Raptor decoding are shown in Figure . We observe that phase 1 takes more than 90% of the total decoding time. Therefore, it is critical to handle the decoding failure in phases 1 or 2 in order to reduce the computation time of Raptor decoding.




(c)




(d)

Figure 3. (a) The method in (Mladenov & Nooshabadi & Kim, 2011) to seek the minimum-degree row; (b) Our proposed FMDS scheme

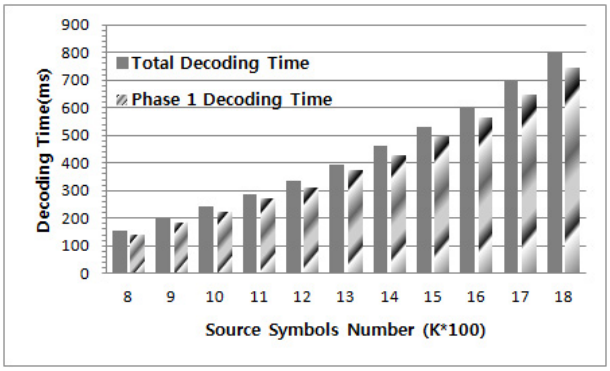



Figure 4. The total decoding time (i.e., decoding time of the transformation process of $A$ and decoding process of $C$) vs. the decoding time of phase 1 only for different number of source symbols ($K$). The symbol size $\ell$ = 128 bytes

**Algorithm 1** Pseudocode of the Proposed Algorithm for Raptor Decoding (code for only phases 1 and 2 is shown)

Initialization: $zeroRowNum = 0$; $i = 0$; $u = 0$;
$degree[s] = ctOnes(A[s])$ for $s = 0, \cdots, M-1$

Phase 1:

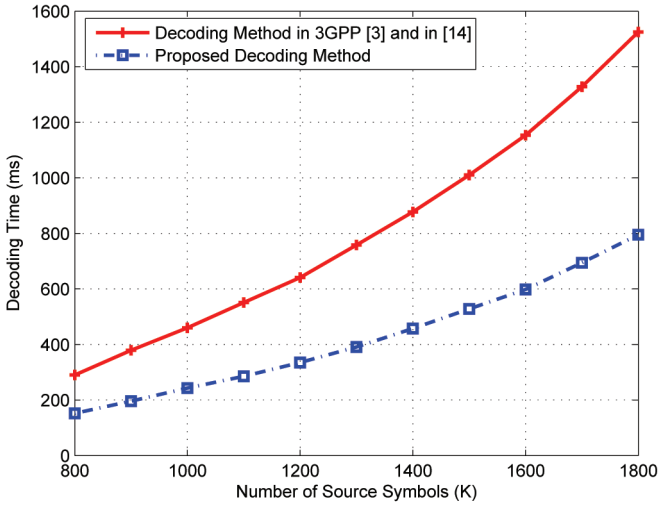
1. while $(i + u) < L$
2. $i = 0$; $u = 0$;
3. Find the minimum non-zero degree $r$ and the row $R_r$ in $A$ with degree $r$ in $V$: $[r \quad R_r] = Find(A)$;
4. $A[i] \leftrightarrow A[R_r]$;$d[i] \leftrightarrow d[R_r]$;$degree[i] \leftrightarrow degree[R_r]$
5. $[A \quad colExPos \quad c] = reordCol(A \quad colExPos \quad c)$; //Recorder columns in $A$ that intersect with $V$; record the column exchanges; $c[i] \leftrightarrow c[j]$ whenever $A(i) \leftrightarrow A(R_r)$ occurs
6. for $k = i + 1, \cdots, M-1$
7. $degree[k] = degree[k] - onesInLastCols[k]$;
8. end for 9. for $k = i + 1$ to $M - 1$
10. $A[k] = A[k] \oplus A[i]$;
11. $D[d[k]] = D[d[k]] \oplus D[d[i]]$
12. if $A[k][i] == 1$
13. $degree[k] = degree[k] - 1$;
14. end if 15. end for 16. $zeroRowNum = findZeros(degree)$;
17. if $zeroRowNum > (M - L)$ // Decoding fails in phase 1.
18. $[G_{LT}]_{n \times L} = genLTMatrix(addESI)$;
19. $[G_{LT}]_{n \times L} = colExchange([G_{LT}]_{n \times L}, colExPos)$;
20. for $s = M$ to $M + n - 1$
21. for $k = 0$ to $i - 1$
22. $A[s] = A[s] \oplus A[k]$;
23. $D[d[s]] = D[d[s]] \oplus D[d[k]]$;
24. end for
25. $degree[s] = ctOnes(A[s - i]) - ctOnes(U[s - i])$;
26. end for
27. $A = [A; [G_{LT}]_{n \times L}]$;
28. $M = M + n$;
29. end if
30. end while

Phase 2:

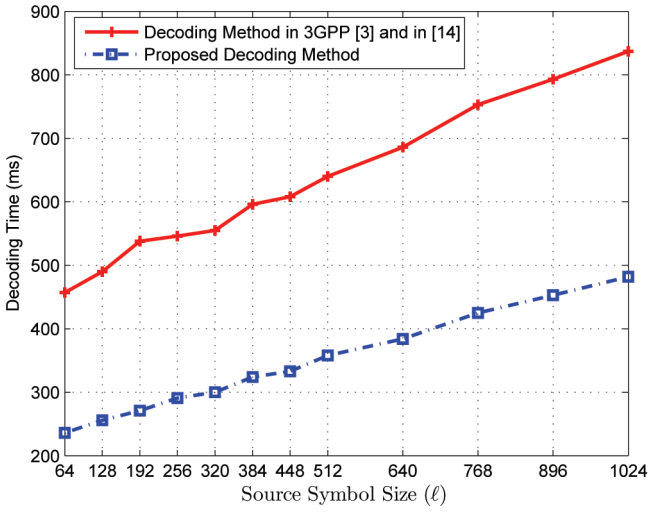
1. for $colInd = i$ to $M - 1$
2. $rowInd = findFirstOne(U_{lower})$//Find the index in $A$ of the first row in $U_{lower}$ that has one in column $colInd$ in $A$.
3. if $rowInd == 0$
4. $[G_{LT}]_{n \times L} = genLTMatrix(addESI)$;
5. $[G_{LT}]_{n \times L} = colExchange([G_{LT}]_{n \times L}, colExPos)$;
6. for $s = M$ to $M + n - 1$
7. for $k = 0$ to $i - 1$
8. $A[s] = A[s] \oplus A[k]$;
9. $D[d[s]] = D[d[s]] \oplus D[d[k]]$;
10. end for 11. end for 12. $A = [A; [G_{LT}]_{n \times L}]$;
13. $M = M + n$; 14. else
15. if $rowInd! = colInd$ // Decoding fails in phase 2.
16. $A[colInd] \leftrightarrow A[rowInd]$; $d[colInd] \leftrightarrow d[rowInd]$;
17. for $k = colInd + 1$ to $M - 1$
18. $A[k] = A[k] \oplus A[colInd]$;
19. $D[d[k]] = D[d[k]] \oplus D[d[colInd]]$;
20. end for
21. end if
22. end if
23. end for

*4.2 Performance of NWFH Scheme*

Figure 5 shows the decoding time using the schemes of (3GPP, 2007) and (Shi, Yang, & Zhang, 2011), and our proposed NWFH scheme when the Raptor decoding failure occurs in phase 1 of the transformation process. Figure 5(a) shows the performance when the size of source symbols ($\ell$) is fixed at 128 bytes and the number of source symbols ($K$) varies. Figure 5(b) shows the performance when $K$ is fixed at 1024 and $\ell$ varies. The NWFH algorithm achieves much lower decoding time than the schemes in (3GPP, 2007) and (Shi, Yang, & Zhang, 2011), because they both restart the decoding process, after receiving additional symbols, when decoding fails in phase 1, whereas our scheme continues decoding after receiving additional symbols without having to restart the phase 1.




(a)

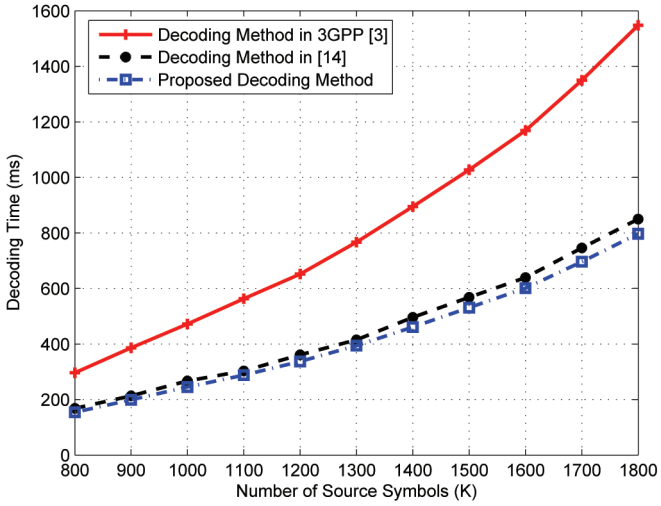



(b)

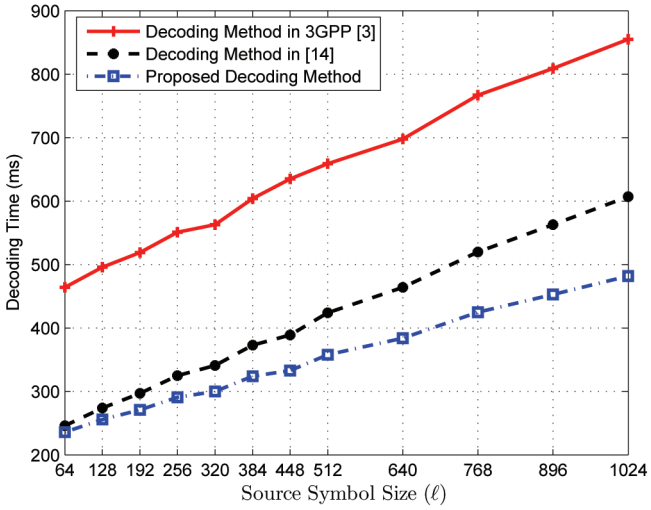
Figure 5. The decoding time when phase 1 fails with (a) different number of source symbols $K$ for a fixed symbol size $\ell = 128$ bytes; and (b) different source symbol sizes $\ell$ for a fixed number of source symbols $K = 1024$

Figure 6 shows the decoding time of the three decoding schemes when the Raptor decoding fails in phase 2 of the transformation process. When decoding fails in phase 2, both phases 1 and 2 are repeated in (3GPP, 2007). Therefore, the decoding time of NWFH scheme is much lower than that of the 3GPP MBMS scheme in (3GPP, 2007) in both Figures 6(a) and 6(b). Since the scheme in (Shi, Yang, & Zhang, 2011) also efficiently handles the decoding failure of phase 2, its performance is close to that of our NWFH scheme when values of $\ell$ are $K$ are small.

However, the scheme in (Shi, Yang, & Zhang, 2011) waits for the phases 3 and 4 to be completed before it can take action on phase 2 failure. When values of $\ell$ or $K$ increases, the time to finish phases 3 and 4 also increases. As a result, the decoding time of the scheme in (Shi, Yang, & Zhang, 2011) increases more than that of our NWFH schemes for larger values of $\ell$ and $K$ in Figures 6(a) and 6(b).




(a)



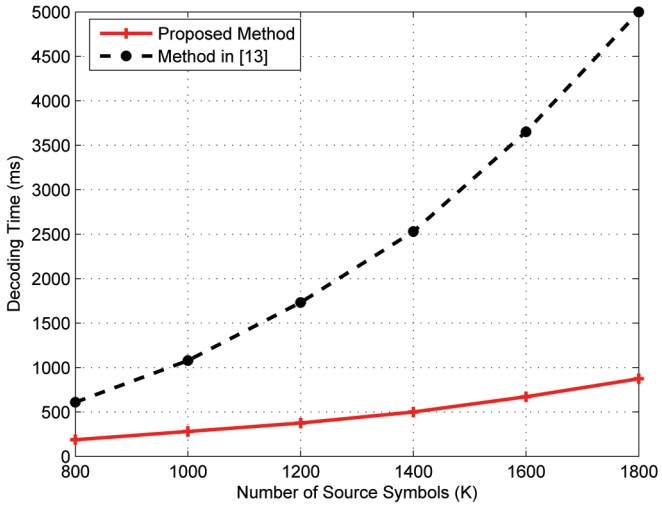

(b)

Figure 6. The decoding time when phase 2 fails with (a) different number of source symbols $K$ for a fixed symbol size $\ell = 128$ bytes; and (b) different source symbol sizes $\ell$ for a fixed number of source symbols $K = 1024$

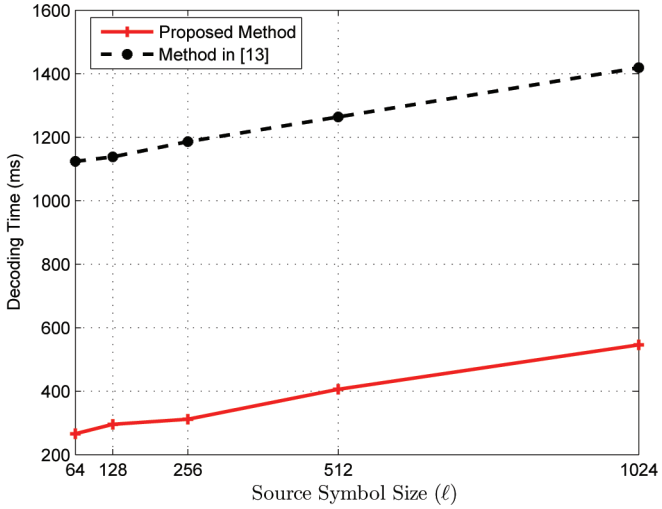
### *4.3 Performance of FMDS Scheme*

In this section, our goal is to study the time saved by our FMDS scheme for searching the row with minimum degree, when all the decoding phases succeed. Note that the decoding schemes in (3GPP, 2007) and (Shi, Yang, & Zhang, 2011) do not discuss how to find a row with minimum degree. We compare our FMDS scheme with the decoding schemes in (Mladenov, Nooshabadi, & Kim, 2011) as it gives the detail of how to find a row with minimum degree for the Raptor decoder. Figure 7(a) shows the decoding time of (Mladenov, Nooshabadi, & Kim, 2011) and our proposed FMDS scheme when $\ell = 128$ bytes and $K$ varies. In Figure 7(b), we fix $K$ at 1024 and vary $\ell$. Our proposed decoding scheme is much faster than (Mladenov, Nooshabadi, & Kim, 2011) because it does not need to count ones in each row when searching the row with the minimum degree in submatrix $V$. Instead, we

iteratively update the degree of each row during phase 1, based on the information from ESI, $G_{LDPC}$, $G_{Half}$ and $G_{LT}$. Our FMDS scheme thus saves much time of phase 1.




(a)




(b)

Figure 7. The decoding time of (Mladenov, Nooshabadi, & Kim, 2011) and our proposed scheme (a) different number of source symbols $K$ for a fixed symbol size $\ell = 128$ bytes; and (b) different source symbol sizes $\ell$ for a fixed number of source symbols $K = 1024$

## 5. Conclusions

In this paper, we have proposed two schemes to improve the decoding efficiency of Raptor decoding. First, the NWFH scheme can resume the decoding process from where the decoding failure occurs after receiving a pre-defined number of additional encoded symbols. This saves the decoding time by avoiding restarting the decoding process from scratch when decoding failure occurs. Second, the proposed FMDS scheme effectively seeks for the row with the minimum degree in matrix A without counting the ones of each row in each iteration. In our experiments, we compared the decoding time complexity of the proposed algorithm with that of other algorithms. our results showed that the proposed algorithm significantly improves the decoding time. We also extended the proposed scheme of failure handling in wireless channels with different erasure rates, to demonstrate the robustness

of improvement in decoding time. In the future work, we will adopt the proposed Raptor decoding schemes in video transmissions in Cognitive Radio Networks.

**References**

Alexiou, A., Bouras, C., & Papazois A. (2010). On the design of Raptor codes for binary-input Gaussian channels. *IEEE. Trans. Commun* (pp. 3269-3277).

Bouras, C., Kanakis, N., Kokkinos, V., & Papazois, A. (2012). Application layer forward error correction for multicast streaming over LTE networks. *International Journal of Communication Systems.*

Cataldi, P., Shatarski, M. P., Grangetto, M., & Magli, E. (2006). Implementation and performance evaluation of lt and raptor codes for multimedia applications. *Proc. IEEE Int. Conf. Intelligent Information Hiding and Multimedia Signal Processing*.

Chen, S., Zhang, Z., Lou, W., & Chen, X. (2012). Adaptive code symbol assignment in a rateless coded multichannel cognitive radio network. *International Journal of Communication Systems.* http://dx.doi.org/10.1002/dac.2463

Cheng, Z., Castura, J., & Mao, Y. (2009). On the design of Raptor codes for binary-input Gaussian channels. *IEEE. Trans. Commun* (pp. 3269-3277). http://dx.doi.org/10.1109/TCOMM.2009.11.070095

DVB. (2004). *Digital Video Broadcasting: Transmission System for Handheld Terminals (DVB-H).*

Etesami, O., & Shokrollahi, A. (2006). Raptor codes on binary memoryless symmetric channels. *IEEE Trans. Inf. Theory* (pp. 2033-2051). http://dx.doi.org/10.1109/TIT.2006.872855

Gasiba, T., Xu, W., & Stockhammer, T. (2008). Enhanced system design for download and streaming services using Raptor codes. *European Transactions on Telecommunications* (pp. 159-173).

Hussein, A., Oka, A., & Lampe, L. (2008). Decoding with early termination for Raptor codes. *IEEE Commun. Lett* (pp. 444-446).

IETF. (2007). IETF RFC 5053: Raptor Forward Error Correction Scheme for Object Delivery. *IETF Proposed Standard.*

Kim, S., & Chung, S. (2008). An efficient algorithm for ML decoding of Raptor codes over the binary erasure channel. *IEEE Commun. Lett.* (pp. 578-580).

Luby, M. (2002). LT codes. *Annual IEEE Symp. Found. of Comp. Sci.* (pp. 271-280).

Mladenov, T., Nooshabadi, S., & Kim, K. (2011). Implementation and evaluation of Raptor codes on embedded systems. *IEEE Trans. Comput.* (pp. 1678-1691). http://dx.doi.org/10.1109/TC.2010.210

Shi, D., Yang, Z., & Zhang W. (2011). A decoding algorithm of 3GPP MBMS Raptor codes. *IEEE Int. Conf. on Commun. Software and Networks.*

Shokrollahi, A. (2006). Raptor codes. *IEEE Trans. Inf. Theory* (pp. 2551-2567). http://dx.doi.org/10.1109/TIT.2006.874390

Shokrollahi, A., & Luby, M. (2011). Raptor codes - foundations and trends in communications and information theory. *Foundations and Trends in Comm. and Inf. Theory* (pp. 213-322).

Stockhammer, T., Shokrollahi, A., Watson, M., Luby, M., & Gasiba, T. (2008). Application layer forward error correction for mobile multimedia broadcasting. *Handbook of Mobile Broadcasting: DVB-H, DMB, ISDB-T and Media FLO* (pp. 239-280). http://dx.doi.org/10.1201/9781420053890.ch10

3GPP. (2007). *3GPP TS 26346 V741: Technical Spec Group Serv and Sys Aspects; Multimedia Broadcast/Multicast Service (MBMS); Protocols and Codecs. 3GPP Technical Spec.*

**Copyrights**