Efficient Incremental Raptor Decoding Over BEC for 3GPP MBMS and DVB IP-Datacast Services

Todor Mladenov, *Student Member, IEEE*, Saeid Nooshabadi, *Senior Member, IEEE*, and Kiseon Kim, *Senior Member, IEEE*

***Abstract*—The application layer forward error correction (FEC) schemes of the 3rd Generation Partnership Project (3GPP) Multimedia Broadcast/Multicast Service (MBMS) standard and IP Datacast over Digital Video Broadcasting (DVB) services employ systematic Raptor codes. Additionally, the 3GPP MBMS standard suggests the use of efficient enhanced Gaussian elimination (EGE) matrix inversion algorithm for the Raptor decoder. In this letter we propose a new incremental EGE (IEGE) algorithm, based on the specification of the EGE. We perform a detailed performance and complexity comparison with EGE, for various decoding scenarios. We show that IEGE performs up to 15.8 times better. Additionally, we investigate the choice of the decoding scheme, based on the symbol arrival rate.**

***Index Terms*—Codes, FEC, incremental decoding, raptor decoder, raptor multimedia broadcasting, 3GPP MBMS.**

I. Introduction

Raptor codes, since they were first introduced in [1], have become the most successful member of the fountain codes family [2], [3]. Their popularity is, primarily, to the fact that they achieve linear encoding and decoding times. Further, they are able to generate arbitrary number of encoding symbols as needed on the fly. This rateless property makes them a preferred choice when the communication channel conditions are unknown or vary extensively [1], [4]–[6]. Above all, Raptor codes have become the dominant technology for broadcast/multicast services, due to the practical elimination of the feedback channel. For that reason they have been chosen for the forward error correction (FEC) scheme at the application layer in the 3rd Generation Partnership Project (3GPP) Multimedia Broadcast/Multicast Service (MBMS) standard [7], [8], and Digital Video Broadcasting (DVB) Internet Protocol (IP) Datacast services [4], [9]–[12]. Furthermore, 3GPP MBMS standard suggests the use of an efficient Raptor decoding algorithm; enhanced Gaussian elimination (EGE) [13], for the most complex Raptor decoding operation – the pre-code matrix inversion. For most practical situations EGE is a preferred technique compared to the well-known Gaussian elimination (GE) [7].

Manuscript received May 13, 2010; revised November 23, 2010; accepted December 08, 2010. Date of publication February 04, 2011; date of current version May 25, 2011. This work was supported in part by the National IT Industry Promotion Agency of Korea and in part by the International Joint R&D Program funded by the Ministry of Knowledge Economy (MKE, Korea).

T. Mladenov and K. Kim are with the Department of Information and Communications, Gwangju Institute of Science and Technology, Gwangju 500-712, Korea (e-mail: todor@gist.ac.kr; kskim@gist.ac.kr).

S. Nooshabadi is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931 USA (e-mail: saeid@mtu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBC.2010.2102930

In this paper we investigate the Raptor decoding for binary erasure channel (BEC). We assume scenarios where the decoding may not be successful with the minimum required number of received symbols, due to errors in the channel. In such cases additional repair symbols are required.

The notion of incremental decoding for Raptor codes was first introduced for the GE technique in [14]. However, since EGE is a vastly superior technique, we propose a new algorithm, incremental EGE (IEGE), for the BEC. We show that IEGE outperforms EGE by up to 15.8 times.

The relative performance of IEGE compared to the incremental GE (IGE) in [14], is similar to that of EGE over GE. It has been found that EGE outperforms GE for typical symbol size values $T \geq 128$ bytes, for all block size values K [15].

This paper is organized as follows. Section II briefly introduces Raptor codes and their encoding and decoding processes. Section III outlines the details of the EGE algorithm. The newly proposed IEGE algorithm, together with its pseudo code, is described in Section IV. Section V presents the simulation results and analysis. Section VI concludes the paper.

II. Raptor Codes

The Raptor code specification in [7], uses a pre-code matrix $\mathbf{A}$ to encode a source symbol vector $\mathbf{k}$ of size K into an intermediate symbol vector $\mathbf{f}$ of size F. Vector $\mathbf{k}$ is padded with $S+H$ zeros $(F=K+S+H)$, prior to the pre-coding. The pre-code matrix $\mathbf{A}$ (of size $F \times F$) contains within it three sub matrices; a low density parity check code (LDPC) matrix (of size $S \times K$), a code with higher density parity check matrix (of size $H \times (K+S)$) and a Luby Transform (LT) code matrix (of size $K \times F$) [15], [16]. The later makes the Raptor code systematic.

The intermediate vector $\mathbf{f}$ is, further, encoded by an LT-code to an encoded vector $\mathbf{e}$. The LT-code provides the rateless property, i.e. vector $\mathbf{e}$ may be of arbitrary size. The first K symbols in $\mathbf{e}$ are the source symbols, while the rest are repair symbols. The pre-code relaxes the complexity of the LT-code, thus making it possible for Raptor codes to achieve linear encoding and decoding times.

The Raptor decoder in [7] applies the pre-code first, and then the LT-code. By doing so, it is able to decode a received encoded vector $\mathbf{e}'$ of size K padded with $S+H$ zeros, into the original vector $\mathbf{k}$, in the absence of errors in the channel. The most computationally intensive operation in the decoder is the inversion of the pre-code matrix $\mathbf{A}$ in;

$$\mathbf{f}^T = \mathbf{e}'^T \times \mathbf{A}^{-1} \tag{1}$$

where $(.)^T$ represents transpose of $(.)$

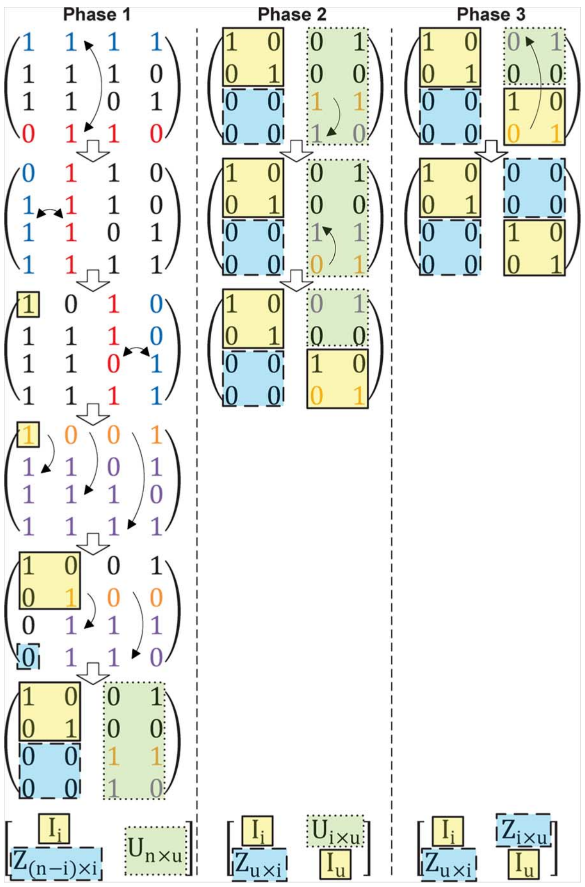

Fig. 1. Example of the operation of the EGE decoding algorithm. ↓ / ↑, ↕, and ↔ denote row exclusive-OR, row exchange, and column exchange, respectively.

III. EGE Decoding Algorithm

The EGE algorithm in [7] employs three main phases to invert the pre-code matrix $\mathbf{A}$. The operation of the algorithm is illustrated through the example in Fig. 1. The form of the matrix $\mathbf{A}$, after each inversion phase, is given at the bottom of Fig. 1. The form of each sub matrix within every step of the example is demonstrated through the use of boxes with different line patterns and colors. The signs ↓ / ↑, ↕ and ↔ denote row exclusive-OR (XOR), row exchange and column exchange operations, respectively. The text coloring in the example helps to identify the matrix elements that are subjected to ↓ / ↑, ↕, and ↔ operations.

To avoid the matrix multiplication in (1), the vector decoding is combined with the matrix inversion and is performed at the same time. When two rows of $\mathbf{A}$ are exchanged or XOR-ed, the same operation is performed on the corresponding symbols in the received vector $\mathbf{e}'$ in (1). At the end when $\mathbf{A}$ is inverted, vector $\mathbf{e}'$ is transformed to a form which is a permuted version of $\mathbf{f}$. Restoring the permuted version of $\mathbf{f}$ to the right order, requires application of the column exchange operations, which were performed on matrix $\mathbf{A}$, to corresponding symbols of $\mathbf{f}$.

Let the pre-code matrix $\mathbf{A}$ have n rows and m columns $(n \geq m)$. To achieve full rank, the dimensions of matrix $\mathbf{A}$ should be such that $n \geq F = m$. *Phase I* of EGE uses two column indices i and u, both initialized to zero. At each step, the algorithm goes through the matrix and selects row $j(j \geq i)$, which has the least (at least one) nonzero elements (r) between column i and column $m-u$. When found, row j and row i are exchanged. Next, if element $a_{ii} == 0$ and there is a nonzero element on row i at position k, columns k and i of matrix $\mathbf{A}$ are exchanged. If there are other nonzero elements on row i, the columns containing them are exchanged with columns $m-u$, and u is incremented with each exchange, $(r-1$ increments in total). Then, row i is XOR-ed with every row $k(n > k > i)$, which has a nonzero elements at position i. At the end of each step, i increments with one. The process repeats until *Phase I* completes successfully, when $i+u=m$. At the end of this phase, matrix $\mathbf{A}$ is reduced to the form shown in the bottom of leftmost column in Fig. 1. Sub-matrix $\mathbf{I}_i$ is an identity matrix, sub-matrix $\mathbf{Z}_{(n-i)\times i}$ is a zero matrix, and sub-matrix $\mathbf{U}_{n\times u}$ is a nonzero matrix, which will be processed in *Phases II* and *III*. If $i+u \neq m$ at the end of *Phase I*, new repair symbols (matrix rows) have to be acquired and the decoding starts from the beginning.

Phase II of the EGE algorithm converts the lower part of the sub-matrix $\mathbf{U}_{n\times u}$ to an identity matrix $\mathbf{I}_u$, through the standard GE process. The forward elimination of GE starts at row i and continues to row $m-1$. If GE is successful, the last $n-m$ rows of matrix $\mathbf{A}$ are discarded and the backward substitution is performed. If the forward elimination is not successful, additional repair symbols (matrix rows) have to be acquired and restart the decoding from the beginning. At the end of a successful *Phase II*, matrix $\mathbf{A}$ becomes square, as shown in the bottom of middle column in Fig. 1.

In *Phase III*, matrix $\mathbf{I}_u$ is used to zero matrix $\mathbf{U}_{i\times u}$. The final from of matrix $\mathbf{A}$ is shown in the bottom of rightmost column in Fig. 1. The details of the EGE algorithm can be found in [7] and [13].

IV. Incremental EGE Decoding Algorithm

The EGE algorithm may fail after *Phase I*, if $i+u<m$, or after *Phase II*, if there are no suitable pivot rows in the GE process [17]. In such cases, additional rows (symbols) have to be added to the matrix and start the decoding again from the beginning. The IEGE algorithm enhances *Phases I* and *II* of EGE with the ability to continue processing with the addition of new matrix rows (symbols), without the need to restart the decoding from the beginning. Algorithm I presents the pseudo code of IEGE.

Algorithm I: The proposed Incremental EGE algorithm

```
Require: $A \in \{0,1\}^{n\times m}; i = 0, u = 0;$
1: Phase I
2: for $(; i+u < m;)$ do
3:   $r = 1;$
4:   for $(; (r \leq m-u-i)\&(!row\_found); r++)$ do
5:     for $(s = i; (s < n)\&(!row\_found); s++)$ do
6:       if $(\mathbf{count\_ones\&store\_pos}(s) == r)$ then
7:         $row\_found = \text{true};$
8:   if $(r > m-u-i)$ then
```

```
9:  while $(\mathbf{count\_ones\&store\_pos}(s = v) == 0)$ do
10:     $\mathbf{add\_new\_rows}(v)$;
11:     $n = n + v$;
12:     $\mathbf{apply\_column\_exchange\_schedule}(v)$;
13: for $(s = n - v; s < n; s++)$ do
14:     for $(k = 0; k < i; k++)$ do
15:       if $a_{sk} == 1$ then
16:         for $l = 0; l < m; l++$ do
17:           $a_{sl} = a_{sl} \bigoplus a_{kl}$
18: if $(i \neq s)$ then
19:   $\mathbf{row\_exchange}(\vec{a}_i, \vec{a}_s)$;
20: if $(a_{ii} == 0)$ then
21:   $\mathbf{col\_exch\&build\_sched}(i, 1s\_col\_pos[0])$;
22: for $h = 1; h < r; h++$ do
23:   $\mathbf{col\_exch\&build\_sched}(m - u - 1, 1s\_col\_pos[h])$;
24:   $u = u + 1$;
25: for $(k = i + 1; k < n; k++)$ do
26:   if $a_{ki} == 1$ then
27:     for $(l = 0; l < m; l++)$ do
28:       $a_{kl} = a_{kl} \bigoplus a_{il}$
29: $i = i + 1$;
30: Phase II
31: for $(b = i, j = i; b < n; b++, j = b)$ do
32:  for $(; a_{jb} == 0; )$ do
33:    $j = j + 1$;
34:   if $j \geq n$ then
35:     $\mathbf{add\_new\_rows}(v)$;
36:     $n = n + v$;
37:     $\mathbf{apply\_column\_exchange\_schedule}(v)$;
38:     for $(s = n - v; s < n; s++)$ do
39:       for $(k = 0; k < b - 1; k++)$ do
40:         if $a_{sk} == 1$ then
41:           for $(l = 0; l < m; l++)$ do
42:             $a_{sl} = a_{sl} \bigoplus a_{kl}$
43:       if $(a_{sb} == 1)\&(j < n - v)$ then $j = s$;
44:  if $b \neq j$ then
45:    $\mathbf{row\_exchange}(\vec{a}_b, \vec{a}_j)$;
46:  for $(k = 0; (k < n)\&(k \neq b); k++)$ do
47:   if $a_{kb} == 1$ then
48:     for $(l = 0; l < m; l++)$ do
49:       $a_{kl} = a_{kl} \bigoplus a_{bl}$
50: Phase III
51: for $(k = 0; k < i; k++)$ do
52:  for $(s = i; s < m; s++)$ do
53:   if $a_{ks} == 1$ then
54:     $a_{ks} = a_{ks} \bigoplus a_{ss}$
```

The IEGE algorithm builds a column exchange schedule in its *Phase I*. After any two columns are exchanged, their position indexes are stored. Further, when the algorithm fails to find a suitable row for position i, it acquires a set of v additional rows (symbols) (line 10 in Algorithm I), which are added at the bottom of the matrix $\mathbf{A}$. The value v depends on the symbol arrival rate (Section V) and the number of available extra symbols. Those new rows have to be "updated" with the decoding progress up to that point. At first, IEGE applies the column exchange schedule to the new rows (symbols). Next, IEGE excludes the rows with index $k (0 \leq k < i)$ from the new rows, through XOR operations. Finally, the algorithm searches for a suitable row to replace the one at position i. If it finds one, IEGE continues its operation in the same way as EGE. If it fails, the process described above repeats.

The behavior of IEGE in *Phase II*, when a proper pivot element can not be found, is similar to the one just described for *Phase I*. A set of new rows are added (line 35 in Algorithm I), updated, and included into the decoding process. The IGE algorithm in *Phase II* is a reduced complexity version of the one reported in [14], achieving up to 7% better decoding time performance.[1] In the IEGE algorithm, IGE is mainly required for processing the diagonal elements that are situated in the lower right corner of the matrix, when significant repair symbols are available. *Phase II* of Algorithm I optimizes the use of those symbols, resulting in an IGE performance improvement better than that reported in [14].

Phase III of IEGE is the same as *Phase III* of EGE.

V. Simulation Results and Discussion

The decoding of a Raptor code usually starts as soon as at least K symbols have been received. This is the minimum number of symbols required to make the pre-code matrix $\mathbf{A}$ invertible and thus the decoding successful. Depending on the communication channel, additional repair symbols may be required for a successful decoding. If the decoding with K symbols fails, we can restart a new attempt after collecting a

[1]The performance measurements of various algorithms discussed in this paper are performed on a NIOS II embedded processor, running on an EP3SL150F1152 Altera Stratix III field programmable gate array (FPGA)

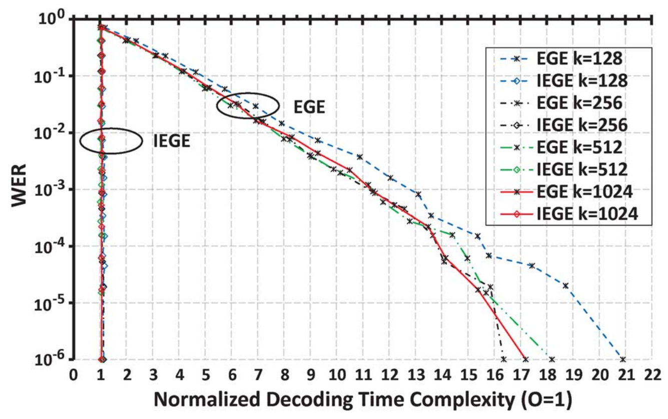

Fig. 2. WER versus normalized decoding time complexity for $O=1$.

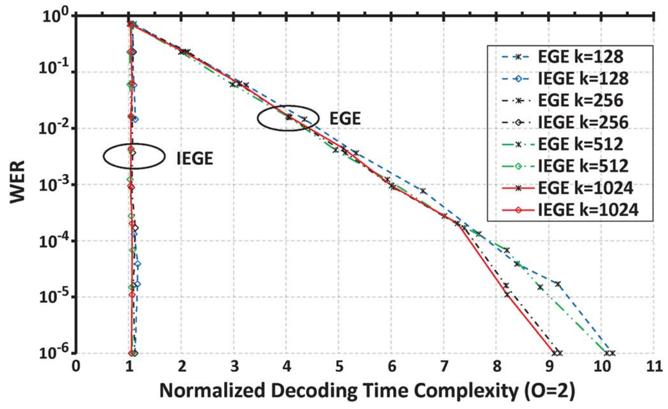

Fig. 3. WER versus normalized decoding time complexity for $O=2$.

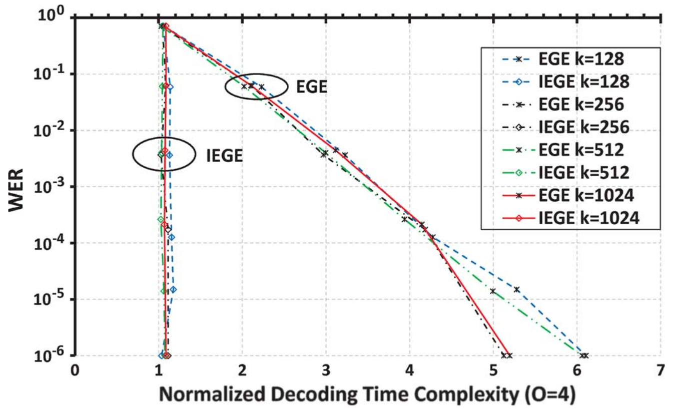

Fig. 4. WER versus normalized decoding time complexity for $O=4$.

group of additional $(O \geq 1)$ symbols. It should be obvious that the probability of successful decoding increases with O.

A. *Effect of Overhead Symbols*

Figs. 2–4 compare the performance of EGE and IEGE algorithms, for $O=1(O_1), 2(O_2)$, and $4(O_4)$ and $K=128$, 256, 512, and 1024. The plots for $O=8$, (not shown here), are similar. The word error rate (WER) probability is plotted against the normalized decoding time complexity, i.e. the time until a

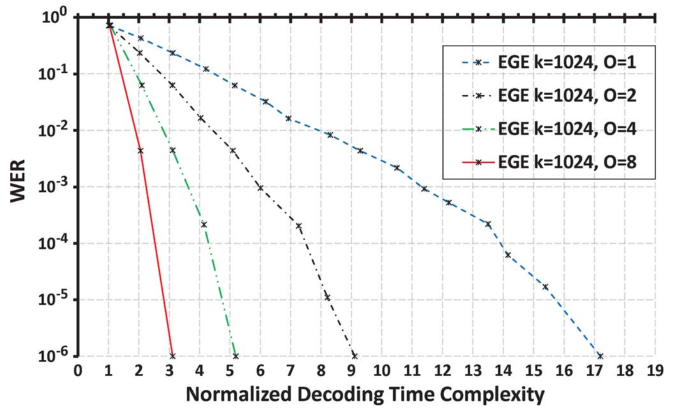

Fig. 5. WER versus normalized decoding time complexity for EGE for $K=1024$, for $O=1$, 2, 4, and 8.

successful decoding, normalized to the time for a single run of EGE with K symbols. The WER is the representation of probability of unsuccessful decoding with $K+n*O$ symbols, where $n=0,1,2\ldots$. In the plots of Figs. 2–4, various values of n for which the measurement are taken, are indicated with symbols $*$ and $\diamondsuit$ for EGE and IEGE, respectively. For each plot the simulation results are acquired using 1 million trials. For $K=512$ and $O=1$, IEGE reduces the complexity by a factor of 15.8, with respect to EGE, while providing the same WER performance. The relative complexity for the two algorithms, does not vary significantly for other values of K. With the increase in the step size O, the advantage of IEGE over EGE reduces, as expected.

We make an interesting observation that the slopes of the plots for EGE in Figs. 2–4, become sharper when the number of additional repair symbols becomes more than 12. This unexpected result is due to the fact that with more rows in the pre-code matrix, during the first phase of its operation, the EGE algorithm is able to find more rows with least number of "1" elements, which results in overall reduction in the decoding time.

One of the main advantages of Raptor codes is that they can guarantee a higher than 99% successful decoding with just 8 extra symbols, for all K values, and almost all erasure rates (channel conditions) [18]. The increase in complexity of the IEGE algorithm rapidly reaches to a minimum floor with the addition of new repair symbols. As a result the IEGE plots in Figs. 2–4 closely overlap each other, for all WER and K values.[2]

To highlight the effect of the step size O on the complexity of EGE decoding processes, we have combined the plots for $O=1$, 2 and 4, for $K=1024$ from Figs. 2–4. Additionally, Fig. 5 shows the plot for $O=8$, for the same block size. Comparing the plots we observe that for the same WER the complexity reduces inversely with O and at the limit reaches the performance of the IEGE. However, this requires a high code overhead reducing the coding and bandwidth efficiency.

B. *Effect of the Symbol Arrival Rate*

To further analyze the role of O in the EGE decoding, we need to include the rate at which the symbols arrive in the anal-

[2]Note that the normalization hides the fact that the linear increase in K leads to a logarithmic increase in the decoding time ($\mathbf{A}$ matrix inversion). This is well documented in [15].

ysis. The overall decoding time for $O = 1$, 2, 4, and 8 can be summarized as;

$$t_1 = \begin{cases} P_0 + \sum_{i=1} P_i(1+i\alpha), & t_s \le 1 \\ P_0 + \sum_{i=1} P_i(\alpha + it_s), & t_s > 1 \end{cases}$$
$$t_2 = \begin{cases} P_0 + \sum_{i=1}(P_{2i-1}+P_{2i})(1+i\alpha), & 2t_s \le 1 \\ P_0 + \sum_{i=1}(P_{2i-1}+P_{2i})(2it_s+\alpha), & 2t_s > 1 \end{cases}$$
$$t_4 = \begin{cases} P_0 + \sum_{i=1}\left(\sum_{j=1}^{4} P_{j+4(i-1)}\right)(1+i\alpha), & 4t_s \le 1 \\ P_0 + \sum_{i=1}\left(\sum_{j=1}^{4} P_{j+4(i-1)}\right)(4it_s+\alpha), & 4t_s > 1 \end{cases}$$
$$t_8 = \begin{cases} P_0 + \sum_{i=1}\left(\sum_{j=1}^{8} P_{j+8(i-1)}\right)(1+i\alpha), & 8t_s \le 1 \\ P_0 + \sum_{i=1}(\sum_{j=1}^{8} P_{j+8(i-1)})(8it_s+\alpha), & 8t_s > 1 \end{cases} \quad (2)$$

Here the decoding time is normalized to the time for a single run of EGE with K symbols. Parameter α is a relative complexity factor corresponding to subsequent restart decoding attempts. In our chosen platform the average value of $\alpha = 1.01$. The P_i parameter is the probability of the decoding success with the i^{th} additional symbol. t_s indicates the normalized symbol arrival time (with respect to a single run of EGE with K symbols).

Using (2), we can identify domains of superior performance for O_1 and O_2 as;

O_2 domain:

$$t_2 - t_1 = -\sum_{i=1}(P_{2i}+P_{2i+1})i\alpha,\ 2t_s \le 1$$

O_1 to O_2 cross over region:

$$t_2 - t_1 = \sum_{i=1}\left(P_{2i-1}(2it_s - 1 - (2i-2)\alpha) + P_{2i}(2it_s - 1 - (2i-1)\alpha)\right), \qquad 2t_s \ge 1 > t_s$$

O_1 domain:

$$t_2 - t_1 = t_s \sum_{i=1} P_i, \qquad 2t_s > t_s \ge 1 \quad (3)$$

We observe that for slow/fast rate of symbol arrival, the use of O_1/O_2 is advantageous. From (3), the t_s value for the cross over point from O_2 to O_1 lies in the region $2t_s > 1 > t_s$. The similar expressions for the cross over regions from O_4 to O_2, and from O_8 to O_4 are;

$$t_4 - t_2 = \sum_{i=1}\left(\sum_{j=1}^{2} P_{4(i-1)+j}(4it_s - 1 - (2i-2)\alpha) + \sum_{j=3}^{4} P_{4(i-1)+j}(4it_s - 1 - (2i-1)\alpha)\right), \quad 4t_s > 1 > 2t_s \quad (4)$$

$$t_8 - t_4 = \sum_{i=1}\left(\sum_{j=1}^{4} P_{8(i-1)+j}(8it_s - 1 - (2i-2)\alpha) + \sum_{j=5}^{8} P_{8(i-1)+j}(8it_s - 1 - (2i-1)\alpha)\right), \quad 8t_s > 1 > 4t_s \quad (5)$$

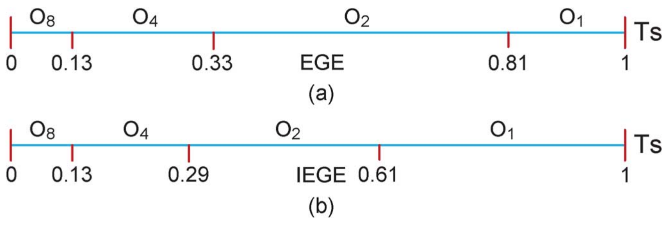

Fig. 6. Cross over points for t_s from O_2 to O_1, O_4 to O_2, and O_8 to O_4 for (a) EGE and (b) IEGE.

Fig. 6(a) shows the cross over points for EGE, for $\alpha = 1.05$ and P_i values for Raptor code obtained by simulations.

From the forgoing discussion, it is obvious that the choice of O depends on the rate of symbol arrival t_s relative to the decoding time. For example, for 2 Mbps CIF H.264 video transmission, with Raptor code symbol size of $T = 256$ bytes, the symbol arrival rate is about 1 ms. On a high end platform with negligible decoding time (less than $(1/0.81) = 1.23$ ms), choosing a step size of $O = 1$ is the best choice. However, on low end embedded system platform, where the decoding time is significant [15], higher values of O are more appropriate. For example, if the decoding time is more than $(1/0.81) = 1.23$ ms, $(1/0.33) = 3.1$ ms, and $(1/0.14) = 7.3$ ms, the step size of $O = 2$, $O = 4$, and $O = 8$, respectively, should be used.

For the case of IEGE where $\alpha = 0.01$ is only a small fraction of the EGE case, (3) modifies to;

O_2 domain:

$$t_2 - t_1 = t_s \sum_{i=1} P_{2i}, \qquad 1 \ge 2t_s > t_s > 0$$

O_1 to O_2 cross over region:

$$t_2 - t_1 = \sum_{i=1}\{P_{2i-1}(2t_s - 1) + P_{2i}(t_s - 1)\}, \qquad 2t_s > 1 \ge t_s > 0$$

O_1 domain:

$$t_2 - t_1 = t_s \sum_{i=1} P_{2i-1}, \qquad 2t_s > t_s \ge 1 \quad (6)$$

The expressions for the domains of O_2, O_4 and O_8 are similar.

With the IEGE algorithm, the cross over points for $O = 2$ to $O = 1$, $O = 4$ to $O = 2$ and $O = 8$ to $O = 4$ shift to 0.61, 0.29, and 0.13, respectively, as shown in Fig. 6(b). For a given symbol rate, the timing margin available for the decoding increases with IEGE. In other words, it shifts the balance in favor of the lower values of O. This is also beneficial in terms of bandwidth efficiency.

VI. Conclusion

This paper presented a new incremental enhanced Gaussian elimination (IEGE) decoding algorithm for Raptor codes, based on the 3GPP MBMS standard decoding algorithm (EGE). The performance of the proposed IEGE in BEC was investigated. Additionally, the two algorithms were compared based on their normalized decoding time complexity and in relation to the symbol arrival rate. It was demonstrated that IEGE shows a significant complexity reduction.

REFERENCES

[1] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2551–2567, June 2006.

[2] D. J. C. Mackay, "Fountain codes," *IEE Proc. Comm.*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[3] D. Sejdinovic, R. Piechocki, A. Doufexi, and M. Ismail, "Fountain code design for data multicast with side information," *IEEE Trans. Wireless Comm.*, vol. 8, no. 10, pp. 5155–5165, Oct. 2009.

[4] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, "Reliable multimedia download delivery in cellular broadcast networks," *IEEE Trans. Broadcasting*, vol. 53, no. 1, pp. 235–246, Mar. 2007.

[5] A. Abdulhussein, A. Oka, T. T. Nguyen, and L. Lampe, "Rateless coding for hybrid free-space optical and radio-frequency communication," *IEEE Trans. Wireless Comm.*, vol. 9, no. 3, pp. 907–913, Mar. 2010.

[6] J. Maisonneuve, M. Deschanel, J. Heiles, W. Li, H. Liu, R. Sharpe, and Y. Wu, "An overview of IPTV standards development," *IEEE Trans. Broadcasting*, vol. 55, no. 2, pp. 315–328, June 2009.

[7] *3GPP TS 26.346, Techn. Spec. Group Serv. and Sys. Aspects; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs*, 3GPP Techn. Spec., Rev. V7.4.1, June 2007.

[8] D. Gomez-Barquero, A. Fernandez-Aguilella, and N. Cardona, "Multicast delivery of file download services in evolved 3G mobile networks with HSDPA and MBMS," *IEEE Trans. Broadcasting*, vol. 55, no. 4, pp. 742–751, Dec. 2009.

[9] *Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols*, ETSI Techn. Spec., Rev. V1.2.1, 2006.

[10] D. Gomez-Barquero, D. Gozalvez, and N. Cardona, "Application layer FEC for mobile TV delivery in IP datacast over DVB-H systems," *IEEE Trans. Broadcasting*, vol. 55, no. 2, pp. 396–406, June 2009.

[11] A. Bouabdallah, M. Kieffer, J. Lacan, G. Sabeva, G. Scot, C. Bazile, and P. Duhamel, "Evaluation of cross-layer reliability mechanisms for satellite digital multimedia broadcast," *IEEE Trans. Broadcasting*, vol. 53, no. 1, pp. 391–404, Mar. 2007.

[12] F. Hartung, U. Horn, J. Huschke, M. Kampmann, T. Lohmar, and M. Lundevall, "Evaluation of cross-layer reliability mechanisms for satellite digital multimedia broadcast," *IEEE Trans. Broadcasting*, vol. 53, no. 1, pp. 188–199, Mar. 2007.

[13] *IETF RFC 5053 (2007): Raptor Forward Error Correction Scheme for Object Delivery*, , Sep. 2007, IETF Proposed Standard.

[14] S. Kim, K. Ko, and S.-Y. Chung, "Incremental Gaussian elimination decoding of raptor codes over BEC," *IEEE Communications Letters*, vol. 12, no. 4, pp. 307–309, April 2008.

[15] T. Mladenov, S. Nooshabadi, K. Kim, and A. Dassatti, "Parallel scalable hardware architecture for hard Raptor decoder," in *Proc., IEEE Int. Symp. on Cir. and Sys. (ISCAS)*, May 2010, pp. 3741–3744.

[16] M. Luby, "LT codes," in *Annual IEEE Symp. on Found. of Comp. Sci.*, Nov. 2002, pp. 271–280.

[17] D. S. Watkins*, Fundamentals of Matrix Computations*, 2nd ed. New York: CJohn Wiley & Sons, 2002.

[18] T. Mladenov, S. Nooshabadi, and K. Kim, "MBMS Raptor codes design trade-offs for IPTV," *IEEE Trans. Consumer Elect.*, vol. 56, no. 3, pp. 1264–1269, Aug 2010.