Improved Stopping Set Elimination by Parity-Check Matrix Extension of LDPC Codes

Saejoon Kim, *Member, IEEE*, Jun Heo, *Member, IEEE*, and Hyuncheol Park, *Member, IEEE*

***Abstract*—Stopping sets associated with a parity-check matrix of low-density parity-check codes limit the performance of iterative decoding over the binary erasure channel. In this letter, we propose a parity-check matrix extension scheme that eliminates stopping sets of small sizes. The results show that our proposed scheme provides significant performance improvement compared to previously known parity-check matrix extension schemes.**

***Index Terms*—LDPC codes, stopping sets, parity-check matrix extension.**

I. Introduction

IT is well known that the nemesis of the iterative decoding of low-density parity-check (LDPC) codes is stopping sets over erasure channels and trapping sets over general symmetric channels. High error floor behaviors of finite-length LDPC codes are attributed to these sets of small sizes, and therefore it is of great interest to eliminate or avoid these sets to improve code performance. In this letter, we consider the LDPC codes used over erasure channels and propose a scheme that eliminates stopping sets of small sizes.

Consider an $[n, k]$ linear code with an $(n-k) \times n$ parity-check matrix (PCM) H. For a subset $\mathcal{S}$ of $\{1, 2, \cdots, n\}$, let $H_{\mathcal{S}}$ denote the $(n-k) \times |\mathcal{S}|$ submatrix of H that consists of the columns indexed by $\mathcal{S}$. A set $\mathcal{S}$ is called a *stopping set* for H if all rows of $H_{\mathcal{S}}$ are not of weight one and at least one row of $H_{\mathcal{S}}$ is of positive weight. The size of the smallest nonempty stopping set is called the *stopping distance* for H. It is clear that if a set of erasures contains a nonempty stopping set then iterative decoding will not be successful. Therefore, an LDPC code's error floor is determined by nonempty stopping sets of small sizes, and for this reason, ways to prevent small stopping sets have been investigated; see [5] and references therein. In particular, it was shown in [5] that if enough redundant rows are added to the PCM of a code, then the stopping distance will exactly equal the minimum distance of the code, that is, the upper bound of stopping distance. However, this may require too many additional rows to be practically meaningful, and some codes are endowed with small minimum distances thereby have high error floors even with maximum-likelihood decoding.

Manuscript received November 1, 2010. The associate editor coordinating the review of this letter and approving it for publication was S. Yousefi.

S. Kim is with the Department of Computer Science and Engineering, Sogang University, Seoul, Korea (e-mail: saejoon@sogang.ac.kr).

J. Heo is with the School of Electrical Engineering, Korea University, Seoul, Korea.

H. Park is with the Department of Electrical Engineering, KAIST, Daejeon, Korea.

The work of S. Kim was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, Science and Technology (2010-0016802) and by the Sogang University Research Grant of 2010.

Digital Object Identifier 10.1109/LCOMM.2011.032811.102095

To this end, stopping set elimination schemes by PCM extension have recently been presented in [3] and [1] with success. In these schemes, a small number of new rows are added to the PCM of a code to eliminate stopping sets of small sizes. In this letter, we improve on these schemes by taking the effect of each stopping set's size on unsuccessful decoding into account for the new row generation. It will be demonstrated that our proposed scheme yields code performance, in terms of word-error-rate (WER), that is at least an order of magnitude better than that of the previous two schemes.

II. Stopping Set Elimination by Parity-Check Matrix Extension

Stopping set elimination by PCM extension refers to adding new rows to the PCM of a code so that small stopping sets can be eliminated. The new rows are either redundant or linearly independent; adding redundant rows has the nice feature of leaving the original code unchanged but can require many rows, and the stopping distance must be strictly less than the minimum distance of the code to be effective. Adding linearly independent rows changes the original code with a slight loss in code rate but typically generates a code with better performance using fewer new rows. We briefly review two PCM extension schemes involving the addition of linearly independent rows that were recently introduced in [3] and [1].

In [3], the locations of 1's in a new row are selected according to the following procedure. The column that is present in the most number of stopping sets is identified, and a 1 is placed in the new row at this column index. All stopping sets that contain this column will then become non-stopping sets since the new row is of weight one in the columns indexed by the stopping sets. The set of stopping sets is next updated, and this process is repeated iteratively until some pre-determined condition is met, *e.g.,* the weight of the new row reaches a maximum or no small stopping sets can be found. In [1], for each column, the numbers of stopping sets that will be eliminated and will be reactivated if a 1 were to be placed in the new row at this column index are determined. Here, reactivated stopping sets represent those that become non-stopping sets but revert back to stopping sets during the elimination process. The difference between the two numbers is computed for each column, and a 1 is placed in the new row at the column index with the largest difference. The sets of the remaining and eliminated stopping sets are updated, and this process is repeated until some pre-determined condition is met, *e.g.,* the difference becomes non-positive or the set of remaining stopping sets becomes empty. In both [3] and [1], additional new rows can be generated by repeated application of the respective processes. The scheme of [1] can be viewed as a refined version of that in [3] with the consideration of reactivated sets, and simulation results verify that the former yields better code performance than the latter. In these schemes, the new rows are generated by treating all stopping sets of varying sizes with equal importance. This approach can clearly be improved by applying different weights to stopping sets of different sizes, as was mentioned but not explicitly described in [1]. To this end, we describe, in the next section, how to properly weight each stopping set for the new row generation so that the code performance can be improved further.

III. Improved Stopping Set Elimination by Parity-Check Matrix Extension

The probability of an iterative decoding error depends on the code's PCM H and can be expressed by the following [6]

$$\Pr(error|H) = \sum_{i=s}^{n} D_i \epsilon^i (1-\epsilon)^{n-i} \tag{1}$$

where s is the stopping distance; D_i denotes the number of sets of size i that contain a stopping set; and ϵ is the channel erasure probability. Thus, each stopping set $\mathcal{S}$ contributes to the decoding error probability by $\geq (1-\epsilon)^n \cdot \left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathcal{S}|}$ with equality for $|\mathcal{S}| = s$. Our proposed stopping set elimination scheme captures this relationship between the size of the stopping sets and the decoding error probability by approximating each stopping set's contribution to the decoding error with $(1-\epsilon)^n \cdot \left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathcal{S}|}$ for all $\mathcal{S}$ and then applying it to the scheme of [1]. Specifically, instead of simply calculating the difference between the *numbers* of stopping sets that will be eliminated and reactivated to find the index of a 1 in the new row, we do the same by calculating for the difference between the *sums of* $\left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathcal{S}|}$ of respective stopping sets. Note that our proposed scheme is defined for each ϵ due to the nature of the decoding error probability shown in Equation (1). The scheme is presented below, where p represents the column indices in which 1's will be placed in the new row and where V represents the set of column indices $\{1, 2, \cdots, n\}$. $\mathcal{S}$ is a stopping set, and S and $\bar{S}$ represent the sets of remaining and eliminated stopping sets, respectively, both of which can be found using the technique from [4]. The scheme below shows how to generate one new parity-check row of maximum weight w_{max} and can be used repeatedly to generate multiple new rows.

Algorithm 1 Improved Stopping Set Elimination Scheme

```
initialization: $p \leftarrow \emptyset$; $\bar{S} \leftarrow \emptyset$;
for $w = 1$ to $w_{max}$
for all columns indices $v \in V$, compute:
% set of stopping sets that will be eliminated by placing 1 at $v$:
$\bar{S}_v \leftarrow \{\mathcal{S} \in S : v \in \mathcal{S}\}$;
% set of stopping sets that will be reactivated by placing 1 at $v$:
$S_v \leftarrow \{\mathcal{S} \in \bar{S} : v \in \mathcal{S}\}$;
end of for
$\tilde{v} \leftarrow \arg\max_v \left( \sum_{\mathcal{S} \in \bar{S}_v} \left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathcal{S}|} - \sum_{\mathcal{S} \in S_v} \left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathcal{S}|} \right)$;
if $\left( \sum_{\mathcal{S} \in \bar{S}_v} \left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathcal{S}|} - \sum_{\mathcal{S} \in S_v} \left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathcal{S}|} \right) > 0$ then
  $V \leftarrow V \backslash \{\tilde{v}\}$;
  $p \leftarrow p \cup \{\tilde{v}\}$;
  $S \leftarrow (S \cup S_{\tilde{v}}) \backslash \bar{S}_{\tilde{v}}$;
  $\bar{S} \leftarrow (\bar{S} \cup \bar{S}_{\tilde{v}}) \backslash S_{\tilde{v}}$;
if $\left( \sum_{\mathcal{S} \in \bar{S}_v} \left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathcal{S}|} - \sum_{\mathcal{S} \in S_v} \left(\frac{\epsilon}{1-\epsilon}\right)^{|\mathcal{S}|} \right) \leq 0$ or $S = \emptyset$ then
  break;
end of for
return $p$;
```

IV. Results

In this section, we present the simulation results of PCM extension schemes applied to LDPC codes used over the binary erasure channel. For the original code, we used a rate-$\frac{1}{2}$, length 500 code that was generated from the progressive-edge growth (PEG) algorithm [2] using an optimized degree distribution. This code's stopping distance was equal to its minimum distance of 8. The PCM extension schemes of [3] and [1] described in Section II and our proposed scheme described in the previous section will be denoted by method 1, method 2 and our method, respectively. For each extension scheme, we tested the addition of one and two new rows to the PCM of the original code without *a priori* restriction on the weight of the new additional rows to examine the limit of performance improvement of the extension schemes with the least possible rate loss. The weights of the new rows for all extension schemes considered were between approximately 20 and 30, whereas the average weight of the rows in the PCM of the original code was approximately 8. The resulting codes' multiplicities of stopping sets, which we found using the technique from [4], are summarized in Table I. Table I clearly shows that method 1 and method 2 do reduce the multiplicity of stopping sets of the sizes shown with respect to the original code; however, the multiplicity of stopping sets of the smallest sizes was not reduced much, if at all, thereby having a limited effect in improving code performance. Our proposed method is designed to remedy this problem and eliminates stopping sets of smallest sizes. In particular, our method yields significantly higher stopping distances. The minimum distances of PCM-extended codes after including one (two) new rows by method 1, method 2 and our method were 8 (8), 9 (10) and 15 (17), respectively.

Figure 1 shows the WER of the original code and of the six types of PCM-extended codes with respect to the channel erasure probability. Our proposed PCM-extended code sufficiently outperformed the original code and the other PCM-extended codes by at least an order of magnitude. Note that both method 1 and method 2 do not benefit much by including two new rows instead of one, as can be seen from Table I. However, the results presented in Table I suggest that our method should benefit by including two new rows, which yields a stopping distance of 17 instead of 15. While the two corresponding curves do not show a sufficient gap in the figure, we observed the gap widening for channel erasure probabilities less than the values shown in the figure. We note that while our method requires the knowledge of ϵ, negligible differences in the stopping set distribution and the

TABLE I
MULTIPLICITY OF STOPPING SETS AFTER ADDING ONE/TWO NEW ROW(S)

stopping set size	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
original code	1	1	1	2	0	3	6	12	17	45	116	198	357	554	900
method 1	1/1	1/1	0/0	1/0	0/0	2/2	6/4	7/1	10/7	30/14	63/34	105/57	212/95	329/187	545/278
method 2	0/0	1/1	0/0	1/0	0/0	0/0	4/1	4/0	2/0	16/4	46/6	74/23	130/38	275/152	467/237
our method	0/0	0/0	0/0	0/0	0/0	0/0	0/0	2/0	1/0	15/1	52/10	104/35	190/73	335/178	564/323

TABLE II
MULTIPLICITY OF STOPPING SETS OF RATE-$\frac{1}{2}$ CODES AFTER REPLACING ONE/TWO ROW(S)

stopping set size	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
original code	1	1	1	2	0	3	6	12	17	45	116	198	357	554	900
method 1	1/1	1/1	1/1	2/2	1/2	3/1	5/5	14/13	17/15	50/51	94/108	179/162	318/241	494/426	712/573
method 2	1/1	1/0	1/1	1/1	0/1	1/1	4/2	9/3	13/3	30/12	76/46	122/77	195/105	441/312	620/436
our method	0/0	0/0	0/0	0/0	0/0	0/0	0/0	2/0	8/0	22/5	72/30	171/96	281/167	462/343	696/493

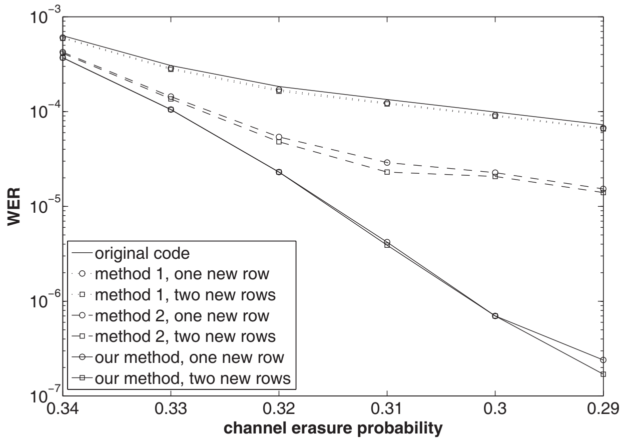

Fig. 1. WER performance of original and PCM-extended codes.

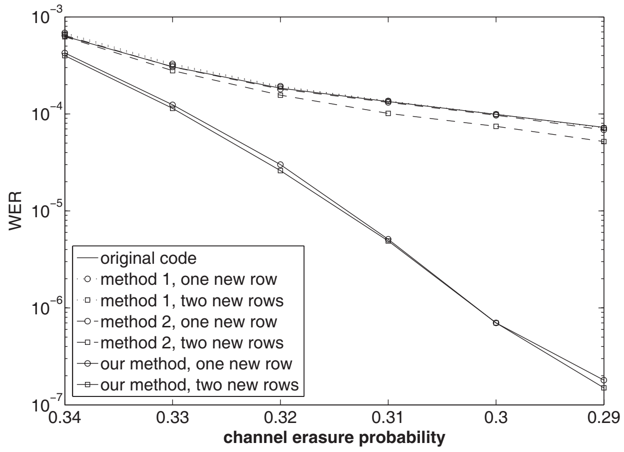

Fig. 2. WER performance of rate-$\frac{1}{2}$ original and PCM-extended codes.

WER performance were observed when ϵ was mis-estimated by up to 10%.

As in [3] and [1], we can incorporate the PCM extension scheme into the code construction so that no loss in the code rate can be attained. Specifically, instead of *adding* new rows as above, new rows, generated by PCM extension schemes, are *replaced* with the rows in the original PCM that were not used in the generation of the new rows. We tried replacing the bottom one or two rows of the PCM of the original code by one or two new rows, respectively. The resulting codes' multiplicities of stopping sets are summarized in Table II. The results presented in Table II indicate that PCM extension schemes reduce the multiplicity of stopping sets with respect to the original code; however, the distribution is worse than before. This result follows from the fact that the stopping set distribution of the PCM from which these new rows are generated is worse than that of the PCM of the original code. Note, however, that our proposed method still works pretty well and that the associated rate-$\frac{1}{2}$ codes have the same stopping distances, 15 and 17, as before shown in Table I. The minimum distances of rate-$\frac{1}{2}$ PCM-extended codes after replacing one (two) rows by method 1, method 2 and our method were 8 (8), 8 (8) and 16 (17), respectively.

Figure 2 shows the WER of the original code and of the six types of rate-$\frac{1}{2}$ PCM-extended codes. Codes constructed by method 1 and method 2 had essentially the same WER as the original code, while codes generated by our method substantially outperformed all other codes. In particular, the codes obtained by our method shown in Figs. 1 and 2 had essentially identical performance values, indicating that code rate loss was not necessarily required for performance improvement. We note that one PCM generated by the PEG algorithm was used to produce the results shown in Figs. 1 and 2.

V. CONCLUSION

In this letter, we proposed an improved stopping set elimination scheme using PCM extension for LDPC codes by exploiting the relationship between the size of a stopping set and the decoding error probability. Our results demonstrate that our proposed scheme provides performance improvement compared to previously developed PCM extension schemes.

REFERENCES

[1] O. Fainzilber, E. Sharon, and S. Litsyn, "Decreasing error floor in LDPC codes by PCM extensions," *IEEE Int'l Symp. Inform. Theory,* 2009.
[2] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory,* vol. 51, no. 1, pp. 386-398, Jan. 2005.
[3] X. Jiao, J. Mu, J. Song, and L. Zhou, "Eliminating small stopping sets in irregular low-density parity-check codes," *IEEE Commun. Lett.*, vol. 13, no. 6, pp. 435-437, June 2009.
[4] G. Richter, "Finding small stopping sets in the Tanner graphs of LDPC codes," *Proc. Int'l Symp. Turbo Codes and Related Topics,* Germany, 2006.
[5] M. Schwartz and A. Vardy, "On the stopping distance and the stopping redundancy of codes," *IEEE Trans. Inf. Theory,* vol. 52, no. 3, pp. 922-932, Mar. 2006.
[6] J. H. Weber and K. A. S. Abdel-Ghaffar, "Results on parity-check matrices with optimal stopping and/or dead-end set enumerators," *IEEE Trans. Inf. Theory,* vol. 54, no. 3, pp. 1368-1374, Mar. 2008.