RC23472 (W0412-057) December 21, 2004
Electrical Engineering

IBM Research Report

A Highly Efficient Importance Sampling Method for LDPC Codes Using Trapping Sets

Enver Cavus, Charles Haymes
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A Highly Efficient Importance Sampling Method for LDPC Codes Using Trapping Sets

Enver Cavus, Charles Haymes

IBM T.J Watson Research Center
e-mail: {ecavus, haymes}@us.ibm.com

Abstract

We introduce an Importance Sampling (IS) method that successfully simulates the performance of Low density Parity Check (LDPC) Codes in an AWGN channel at very low bit error rates (BERs). The proposed technique is independent of block size of a given LDPC code and attains significant simulation gains. By effectively finding and biasing bit node combinations that are the dominant sources of error events, called trapping sets, the developed technique provokes more frequent decoder failures. Consequently, fewer simulation runs and higher simulation gains are achieved. The proposed method consists of two steps. In the first step, a novel trapping search algorithm is used to identify all trapping sets and then, the trapping sets are classified based on the total number of bits involved in each trap. In the second step, IS is applied to only one trapping set from each class and the total BER is estimated by adding the BER contribution of each class. Regardless of the block size of an LDPC code, only a few dominant trapping set classes cause decoder failures at low BER regions. Therefore, the proposed technique allows the performance evaluation for any size LDPC code at very low BER regions with remarkable simulation gains in the order of 10^{14} at BER of 10^{-20}.

I. INTRODUCTION

The performance evaluation of Low Density Parity Check (LDPC) codes at very low bit error rates (BERs) are of great interest. Low BER estimation of LDPC codes exposes the error floor regions that are beyond the reach of traditional simulation and helps the

feasibility studies of employing LDPC codes in applications requiring very low BERs. So far, no analytical tool is available to evaluate the performance of LDPC codes and a Monte Carlo (MC) simulation is the only way to assess the performance of these codes. However, at very low BER regions, traditional MC simulations cannot be performed due to prohibitive simulation times. In order to avoid long simulation times, a fast BER simulation technique, called Importance Sampling (IS), can be applied. IS methods are widely used in digital communications [11][12][13] and has been also effectively applied to FEC schemes [14][15][16][17]. If a proper biasing scheme can be developed for the required application, IS methods promise significant simulation gains while the performance is accurately predicted.

Recently, two independent IS approaches have been proposed to address the performance evaluation of the LDPC codes at very low BERs [1][2]. In [1], a parity check biasing scheme has been first developed for the single parity check (SPC) codes and then applied to LDPC codes. Performance of a (96, 48) LDPC code is demonstrated up to the BER of 10^{-14} and, compared to a standard MC technique, simulation gains in the order of 10^5 are reported. In a more recent paper [4], the same authors provided another modified IS technique that incorporates the biasing method in [1] with an error set partitioning method for word error rate (WER) estimation. However, the proposed method does not yield an optimum biasing scheme for LDPC codes and estimates the total BER by employing IS simulation to each bit separately. For example, for a (96, 48) LDPC code 96 independent IS simulations is required to calculate the complete BER. In [1], this drawback is referred as "divide by N" problem, as the simulation gain provided by IS needs to be divided by the code length to get actual gain achieved. Both limitations impair the efficiency of the IS method and hence, sufficient simulation gains cannot be obtained to estimate the performance of longer LDPC codes.

In the second approach, [2], a two phase adaptive importance sampling method, called Dual Adaptive Importance Sampling (DAIS), based on Multicanonical Monte Carlo (MMC) simulations is introduced. Unlike the scheme in [1], this method does not utilize a priori biasing scheme, instead first an unconstrained and then a constrained MMC algorithm adaptively iterates to find the optimum IS biasing scheme. Based on the DIAS algorithm, BER and WER curves for (96, 48) LDPC code are illustrated for as low as 10^{-19} with simulation gains of 10^{7}. Unfortunately, the complexity of the adaptive search algorithm depends on the block size of the code and the efficiency of the algorithm gets deteriorated for bigger size LDPC codes since a larger noise space has to be searched for an optimum bias.

In this work, we propose a highly efficient IS scheme for LDPC codes, which is independent of block size of a given LDPC code and achieves higher simulation gains. By effectively finding and biasing bit node combinations that are the dominant source of error events, the developed technique provokes more frequent decoder failures. Having more error events leads to statistically valid simulations in less time with higher simulation gains. It is well known that medium to long block size LDPC codes have large minimum distances and, the decoder failures are mainly due to near codewords [5][6][7]. In [3], the combinations of bit nodes that prevent the LDPC decoder from converging to the transmitted codeword are defined as trapping sets. Based on these trapping sets, a computational technique is developed to predict the error floor of a given LDPC code. In this paper, a two step algorithm is introduced to estimate low BER performance of LDPC codes. In the first step, a novel trapping search algorithm is used to identify all trapping sets and then, the trapping sets are classified based on the total number of bits involved in each trap. In the second step, IS is applied to only one trapping set from each class and the total BER of the code is estimated by adding the BER contribution of each trapping class. Regardless of the block

size of an LDPC code, only a few dominant trapping set classes cause decoder failures at low BER regions. Therefore, the proposed technique allows the performance evaluation for any size LDPC code at very low BER regions with remarkable simulation gains in the order of 10^{10}.

The remainder of this paper is organized as follows. In Section II, we briefly review IS method and describe how a biasing scheme is applied to LDPC codes. Section III discusses trapping sets and explains the proposed algorithm in detail. Then, the proposed IS approach is applied two different LDPC codes and BER performances are illustrated in Section IV. Finally, Section V concludes the paper.

II. IS FOR LDPC CODES

In this section, we briefly review IS and discuss its application to LDPC codes by reviewing the proposed biasing scheme in [1]. As a similar biasing scheme is developed in the next section, describing the simple parity check biasing scheme of [1] facilitates to understand the proposed IS algorithm using the trapping sets.

A. Importance Sampling

Importance sampling is a fast simulation technique that aims to increase the speed of simulation by sampling rare error events. If the regions of noise samples, which contribute to an error event, are known, then the noise distribution can be modified such that more samples are taken from these "important" regions. Variance scaling and mean translation (MT) are the two common ways of biasing a noise distribution. The most effective IS method used in ECC schemes is the MT method, as variance scaling has a dimensionality effect [13]. In mean translation, the mean of the original noise density, f(x), is biased such that the new noise density, f(x)*, **Figure** 1, moves the transmitted codeword towards its nearby codewords. Hence, more frequent decoder failures are encountered. Consequently,

compared to a traditional MC method, low BER performance is effectively simulated with significantly fewer runs. As the bias associated with each noise sample is known, the error count per simulation run can be unbiased and translated back to obtain true BER estimate.

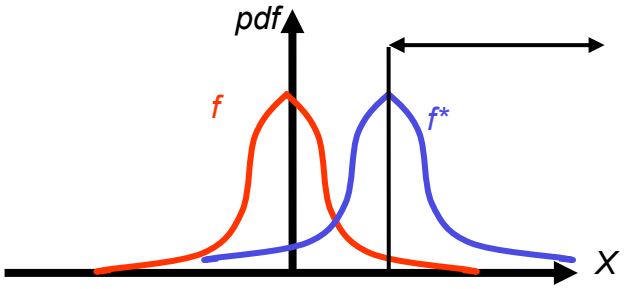

Figure 1. Mean translation method for Importance Sampling

In a Monte Carlo simulation, the word error rate of a coded system is the average number of error events, which occur during the transmission of N_s codewords. Given an error control code that is described by discrete set of codewords x_i ϵ £ , the MC estimator $\hat{P}_{MC}$ is given by

$$WER = \hat{P}_{MC} = \frac{1}{N_s}\sum_{i=1}^{N_s} \mathrm{I}_E(x_i) \tag{1}$$

where $\mathrm{I}_E(x_i)$ is the indicator function of the error set E as defined in (2).

$$I_E(x) = \begin{cases} 1 & if \quad x \in E \\ 0 & if \quad x \in \overline{E} \end{cases} \tag{2}$$

In a standard MC simulation, there is no bias on the nose distribution so the WER can be calculated directly from (1). In an IS simulation, the noise vector is sampled from a biased distribution, $f(x)^*$, and therefore the probability of the selected noise sample is increased by the bias

$$B(x) = \frac{f^*(x)}{f(x)} \tag{3}$$

In order to unbias the IS estimator, a weight function is introduced into (1) to adjust the error count. Equation (4) gives the weight function as the inverse of the bias function $B(x)$.

$$w(x) = \frac{1}{B(x)} = \frac{f(x)}{f^*(x)} \tag{4}$$

Applying equation (4) to equation (1) yields the IS WER estimator $\hat{P}_{IS}$ as

$$WER_{IS} = \hat{P}_{IS} = \frac{1}{N_s}\sum_{i=1}^{N_s} I_E(x_i)w(x_i) \tag{5}$$

Apparently, if the biased distribution is chosen as $f^*(x) = f(x)$, then the important sampling estimator reduces to the MC estimator as in (1). In order to evaluate the quality of a used estimator $\hat{P}$, the normalized error could be computed as

$$\varepsilon = \frac{\sqrt{\mathrm{var}(\hat{P})}}{\hat{P}} \tag{6}$$

In a standard MC simulation, the variance of the simulation is given by

$$\mathrm{var}(\hat{P}_{MC}) = \frac{\hat{P}_{MC} - \hat{P}_{MC}^2}{N_s} \;;\; \frac{\hat{P}_{MC}}{N_s} \tag{7}$$

and the MC normalized error estimator is estimated through equation (8).

$$\varepsilon_{MC} = \frac{\sqrt{\mathrm{var}(\hat{P}_{MC})}}{\hat{P}_{MC}} = \frac{1}{\sqrt{N_{err}}} \tag{8}$$

In an IS simulation, the estimator has the well-known variance formula

$$\mathrm{var}(\hat{P}_{IS}) = \sigma_{IS}^2 = \frac{1}{N_s}\left(\frac{1}{N_s}\sum_{i=1}^{N_s}\left[I_E(x_i)w(x_i)\right]^2 - \hat{P}_{IS}^{\,2} \right) \tag{9}$$

where $\hat{P}_{IS}$ is given in (5). To achieve the same level of confidence in IS simulations as in MC simulations, we follow the criteria given in [16], where IS simulation is continued until the condition in (10) is satisfied.

$$\varepsilon_{IS} \le \varepsilon_{MC} \quad \Leftrightarrow \quad \frac{1}{N_{s,IS}}\left(\frac{1}{N_{s,IS}}\sum_{i=1}^{N_{s,IS}}\left[I_E(x_i)w(x_i)\right]^2 - \hat{P}_{IS}^{\,2} \right) \le \frac{1}{\sqrt{N_{err,MC}}} \tag{10}$$

Finally, the speed-up gain of the IS simulation is defined as

$$G = \left(\frac{N_s^{MC}}{N_s^{IS}} \right)_{\varepsilon_{IS}=\varepsilon_{MC}} \tag{11}$$

where N_s^{MC} and N_s^{IS} are the number of codewords simulated by MC and IS simulations, respectively.

It can be concluded from the above discussions that the efficiency of the IS mean translation method depends on the choice of the biased distribution. In order to simplify the biasing of multi-dimensional systems, the error region is partitioned into simple sub-error regions and then MT is applied to each sub-error region separately to obtain error statistics. In the case of linear block codes, a practical choice of the biasing scheme for the optimal or near optimal decoders is to shift the mean of noise density at the boundary of the error region and transmitted codeword [16][17]. For LDPC codes, the employed soft iterative Sum-Product Decoder is non-optimal and the iterative process complicates defining an exact error region for the choice of a good biasing scheme.

B. Biasing of IS for LDPC Codes

In the first work applying IS to LDPC codes [1], the Sum-Product decoder is applied to a single parity check (n, n-1) code, and a biasing scheme, which considers the error probability of only one bit at a time, is developed. It is shown that the best bias for the mean of the noise density is given by

$$\mu = \left\{1, \quad \frac{1}{n-1}, \quad \frac{1}{n-1}, \quad \mathrm{K}\ , \quad \frac{1}{n-1}\right\} \tag{12}$$

where in this case bit position 0 is chosen for IS simulation and all-zero codeword transmission is considered. Figure 2 demonstrates an example of this biasing scheme for a (6, 5) single parity check code with associated bias values. Note that the total bias value for a parity check equals to 2, which causes the parity check to make mis-corrections. As the biasing scheme is applied to each bit node separately, the individual BER estimates are summed in order to obtain total BER value.

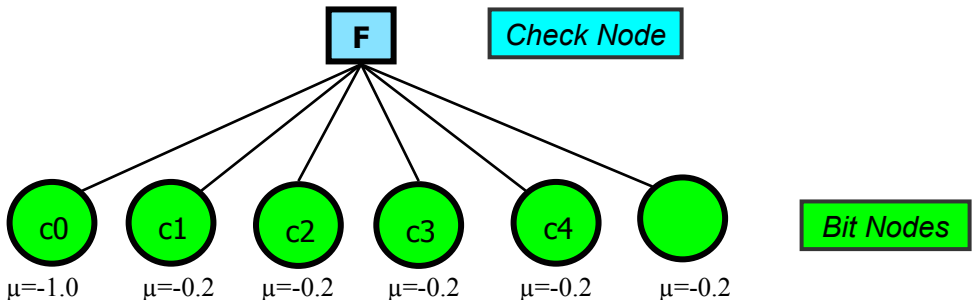

Figure 2. Graph for a single parity-check code and associated IS bias values for simulation of bit 0.

The resulting weight function for an all-zero codeword transmission, which corresponds to all-one channel vector, in AWGN channel is given by

$$w(r) = \frac{f(r)}{f^*(r)} = \frac{\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=0}^{N-1}(r_i - 1)^2\right\}}{\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=0}^{N-1}(r_i - (1+\mu_i))^2\right\}} \tag{13}$$

where N is the length of the code and r is the received channel vector.

This technique is then extended to LDPC codes by modeling the LDPC code as a 2-layer tree, which consists of concatenation of multiple single-parity check codes. **Figure 3** shows the construction of a 2-layer tree from a simple (7, 4) code from the bi-partite graph, where bit position ν_0 is chosen for simulation. The bias technique introduced for SPC codes are then adapted for each of two parity checks that ν_0 participates and associated bias values for each bit is shown in **Figure 3**.

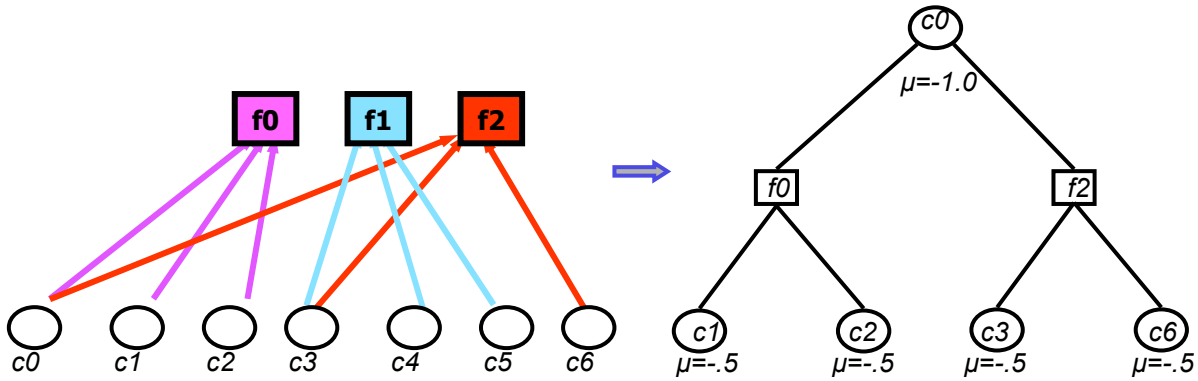

Figure 3. Two layer tree construction from LDPC bipartite graph and associated bias values.

Although the proposed IS MT method works successfully for the SPC codes, this scheme does not yield an optimum biasing for LDPC codes. LDPC codes forms much deeper decoder tree and the bit nodes at the bottom of the two-layer representation also contains multiple parity checks that improves their reliability while iterating. The weak bias values at the bottom bit nodes are easily reverted by the third level parity checks and decoder failures cannot be generated frequently. As a result, this impairs the efficiency of the IS method and

leads to reduced simulation gains. In addition, "divide by N" problem as mentioned earlier further decreases the gain obtained using IS approach and prevents adoption of this technique for the evaluation of medium to long block size LDPC codes.

As it will be more apparent from the discussions in the next section, applying mean translation to trapping set bits at the same time eliminates the limitations of the technique in [1] and result in a highly efficient IS estimator.

III. PROPOSED APPROACH: IS WITH TRAPPING SETS

In order to develop an efficient algorithm, it is necessary to accurately define the error regions for an LDPC code. In this work, unlike the previous approaches in [1] and [2], an error set partitioning based on trapping set bits is considered. As mentioned earlier, medium to long length LDPC codes have good minimum distance properties, and decoding errors are caused only when the decoder fails to converge the transmitted codeword after maximum number of iterations. In other words, the error space of LDPC codes consists of trapping regions or states, and once captured the decoding algorithm is never able to recover from these states. In our approach, error space is partitioned based on these trapping regions and IS simulation is tuned for these specific regions by biasing the corresponding trapping set bits. Consequently, a block size independent IS simulation technique with higher simulation gains is developed.

A. Definition of the Trapping Sets

The combination of error bits that leads to a decoder failure are defined as trapping sets. We label the trapping sets as (x, y) trapping sets, where x is the number of error bits in a failure and y represents the number of unsatisfied checks caused by x error bits. The significant characteristic of these trapping sets is that they contain multiple "mis-satisfied" parity check nodes. A check node is called a "mis-satisfied check" if the parity check contains even

number of error bits. In a mis-satisfied check node, all extrinsic outputs are unreliable, which makes the decoder difficult to recover all the error bits. When specific mis-satisfied checks occur at the same time, a trapping set is formed and the decoder reinforces the error bits, which finally causes decoder failures. Figure 4 illustrates a (4, 2) trapping set for (96, 48) LDPC code in a bipartite graph, where error bits are shown in red and only the edge connections of the error bits are included. The blue nodes are the mis-satisfied parity checks that prevent the decoder escape from the error state.

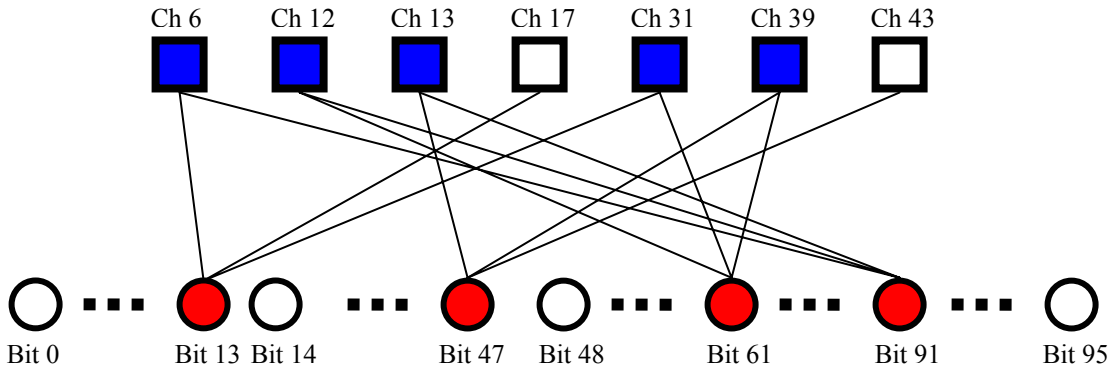

Figure 4. Illustration of a (4, 2) trapping set for (96, 48) LDPC code in a bipartite graph.

Another way of viewing these trapping sets is the parity check equations. Equation (14) exemplifies the same (4, 2) trapping set. As seen in Equation (14), only the parity check nodes 17 and 43 are not satisfied and these two nodes are the only reliable parity checks that are trying to correct b_{13} and b_{47} while all the other check nodes resist the correction operation.

$$
\begin{aligned}
&b_5 \oplus b_{13} \oplus b_{33} \oplus b_{48} \oplus b_{91} \otimes b_{92} = 0 \quad (check\ node\ 6)\\
&b_{10} \oplus b_{13} \oplus b_{43} \oplus b_{62} \oplus b_{80} \otimes b_{81} = 1 \quad (check\ node\ 17)\\
&b_{13} \oplus b_{14} \oplus b_{34} \oplus b_{56} \oplus b_{61} \otimes b_{82} = 0 \quad (check\ node\ 31)\\
&b_{25} \oplus b_{27} \oplus b_{47} \oplus b_{83} \oplus b_{91} \otimes b_{93} = 0 \quad (check\ node\ 13)\\
&b_{21} \oplus b_{41} \oplus b_{47} \oplus b_{68} \oplus b_{71} \otimes b_{87} = 1 \quad (check\ node\ 43)\\
&b_7 \oplus b_{39} \oplus b_{47} \oplus b_{52} \oplus b_{58} \otimes b_{61} = 0 \quad (check\ node\ 39)\\
&b_{21} \oplus b_{23} \oplus b_{24} \oplus b_{61} \oplus b_{74} \otimes b_{91} = 0 \quad (check\ node\ 12)
\end{aligned}
\tag{14}
$$

Note that the strength of the trapping set becomes stronger, as x is increasing and y is decreasing. For instance, given an LDPC code a (5,1) trapping sets are more powerful traps than (4,2) sets. In a (5, 1) trapping set, 1 parity check strives to correct the error bits against 5 mis-satisfied check nodes, whereas 2 parity checks attempt to improve reliability of the messages against 4 mis-satisfied checks in a (4,2) trap.

B. Trapping Set Search

The set of bits that leads to a decoder failure can be identified for each error event and one simple method of locating the LDPC code's trapping sets is described in [3]. In [3], the trapping set bits are simply searched by running the decoder at a particular SNR until a decoder failure occurs. Afterward, the sum-product decoder is run an additional 20 iterations and the bits that are in error during these 20 iterations are identified as a trapping set. Table 1 presents the trapping sets for the (96, 48) LDPC code observed at Eb/N0 from 6 to 7.5 by using the proposed technique in [3]. This particular code has a minimum distance of 6 and there are two codewords at the minimum distance, which is referred as (6, 0) in Table 1.

Trap	Bits	Unsatisfied Check Node	Trap	Bits	Unsatisfied Check Node
(6,0)	16 26 64 78 88 90	None	(5,1)	19 29 49 60 90	32
(6,0)	10 35 43 44 64 90	None	(5,1)	19 32 49 51 90	18
(5,1)	0 20 31 54 66	23	(5,1)	21 27 86 91 92	3
(5,1)	1 2 18 22 37	36	(5,1)	29 72 73 85 94	8
(5,1)	4 54 65 80 81	35	(4,2)	13 47 61 91	17, 43
(5,1)	11 17 37 55 90	44	(7,1)	12 16 17 45 50 52 58	16
(5,1)	13 14 43 48 78	41	(7,1)	12 47 50 58 59 87 93	35

Table 1. Trapping sets of a (96,48) LDPC code identified using MC simulations.

Although, using a MC simulation approach as in [3] is a simple way of locating the trapping sets, it is not a very efficient method to find out all the trapping sets at low BERs. The dominant trapping sets changes as the SNR increases and in order to identify the effective trapping sets at low BERs, MC simulations need to be performed in the proximity of the low BER region of interest, which requires very long simulation times. Table 2 presents the

percentage of trapping sets at different SNRs and required MC simulation times for the (96, 48) code.

SNR	(4,2)	(5,1)	(6,0)	(7,1)	All other (x,y) traps	Simulation Time (min)
4	--	--	15%	--	85%	1
5	5%	5%	30%	10%	50%	5
6	---	60%	20%	10%	10%	46
7	10%	50%	30%	10%	--	778

Table 2. Percentage of trapping sets at different SNRs and required simulation times.

As shown in Table 2, it took 13 hours to reach 20 error events at 7 dB. Even though this simulation time might be reasonable for short block size LDPC codes, it will be very impractical for longer codes. Furthermore, as observed in Table 2, the dominant trapping sets changes as SNR increases and a MC simulation technique is unable to search the trapping sets at very low BER regions of interest.

In order to facilitate trapping set search problem at very low BER regions such as 10^{-15}, we propose a novel search technique that effectively finds the dominant trapping sets from the bipartite graph of a given LDPC code. Note that an additional trapping set search in a LDPC code graph has been mentioned in [3] but this graph search technique is not discussed in detail. If the trapping sets of Table 1 are investigated carefully in the bipartite graph of the (96, 48) LDPC code, it is easy to see that these trapping sets formed by overlapping short cycles existing in the bipartite graph. For example, the (4, 2) trapping set of Figure 4 consists of 2 different cycle-6s and these cycle-6s are shown in Figure 5 and Figure 6, respectively.

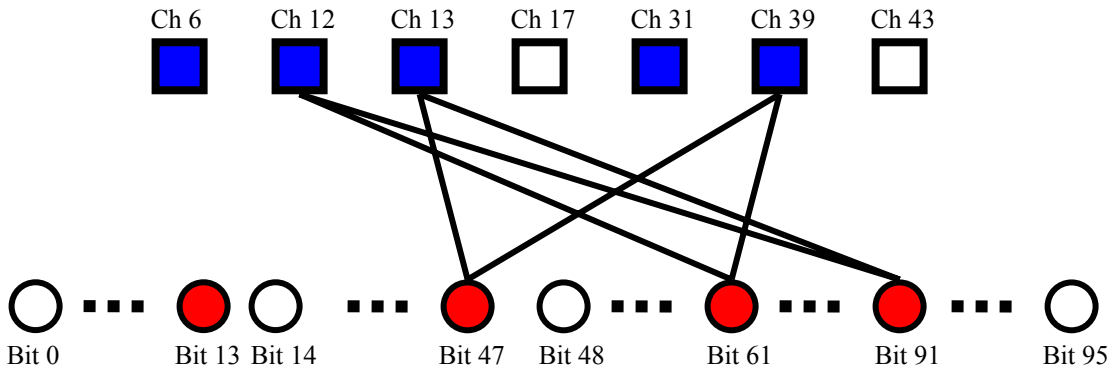

Figure 5. The first cycle 6 in a (4,2) trapping set of (48, 96) LDPC code.

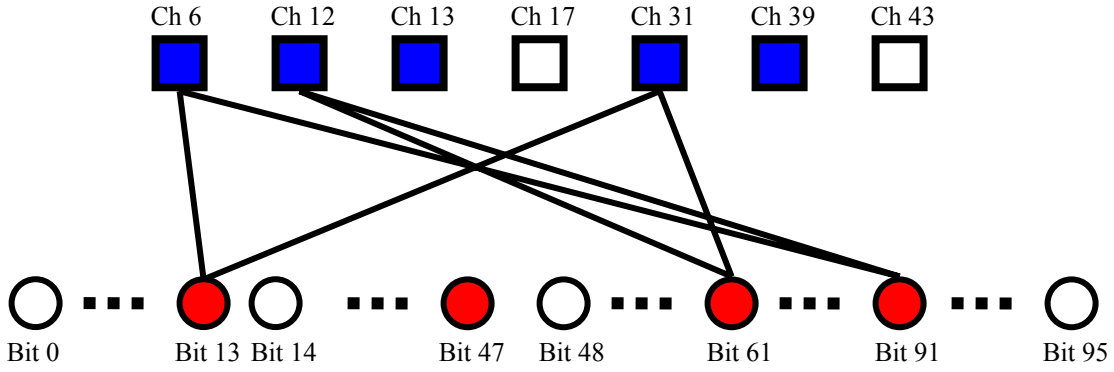

Figure 6. The second cycle 6 in the (4,2) trapping set of (48, 96) LDPC code.

It is very intuitive that these trapping sets involve bits with very short cycles. Therefore, we search the bipartite graph and list all the existing cycle-6s in the graph, which is the shortest cycle in the simulated (96, 48) LDPC code. As an example, Table 3 lists all the cycle-6s for bits 0, 1, and 61 and exemplifies one trapping set at the last row formed by the colored cycle-6s. The (4, 2) trapping set in Figure 6 is formed by the two cycle-6s, (61 13 91) and (61 47 91) as it is observed from the last column of Table 3.

Bit 0 Cycle-6s	Bit 1 Cycle-6s	Bit 61 Cycle-6s
0 7 54	1 60 85	61 14 74
0 20 54	1 60 78	61 23 82
0 65 72	1 9 58	61 13 91
0 20 66	1 55 79	61 7 56
0 7 20	1 2 22	61 47 91
0 49 73	1 70 78	61 24 47
	1 26 85	
	1 22 37	

(5,1) 0 20 31 54 66	(5,1) 1 2 18 22 37	(4,2) 13 47 61 91

Table 3. Cycle-6s list for bits 0, 1 and 61.

It is clear that (4, 2) trapping sets involves 2 cycle-6s, though it is not obvious which of 2 cycle-6 forms a strong trapping set. In addition, in (5, 1) trapping sets, there is an additional bit, which does not belong to any cycle-6s and but still appears in the trapping sets. For example, bits 31 does not form any cyle-6s with bit 0 but participates in the trapping set. Although combination of short cycles forms the basis for a trapping set, the final appearance of a trapping set is ambiguous and it is hard to determine by simple observation of the cycles in bipartite graph. In order to overcome this difficulty and identify the trapping sets for each bit, we first list all the member bits that form a cycle for that particular bit and then bias these bits at the same time. Then we run a fixed number of simulation loops, say Ls=1000 loops, with biased cycles and observe the resulting trapping sets. Table 4 shows the list of all cycle-6 bits for the first 9 bit of (96, 48) code.

All Cycle-6 Bits
0 7 54 20 65 72 66 49 73
1 60 85 78 9 58 55 79 2 22 70 26 37
2 15 31 18 95 63 1 22
3 6 69 95 18 57 65
4 23 81 82 32 46 25 84
5 16 45 68 87 8 48
6 3 69 95 21 24 36 92 59 84 38 39
7 36 39 58 73 9 56 61 20 66 0 54
8 5 68 48 87 40 81

Table 4. The list of all cycle-6 bits for the first 9 bit of (96, 48) LDPC code.

Compared to the MC search method for trapping sets in [3], the proposed search method only involves at most (N*Ls) IS simulations, yet it is very effective to locate all the trapping sets involved.

C. Classifiying Trapping Set Classes

Once all trapping sets are found, IS might be applied to all trapping sets one at a time. Then, the total BER could be estimated through addition of the BER values of the individual trapping sets. Figure 7 shows the trapping set in a LDPC graph and the associated bias

values. Contrary to Figure 2, only the bits that belong to the trapping set are biased with equally strong bias values of (-1), which adds up to (-2) to mis-satify the parity check.

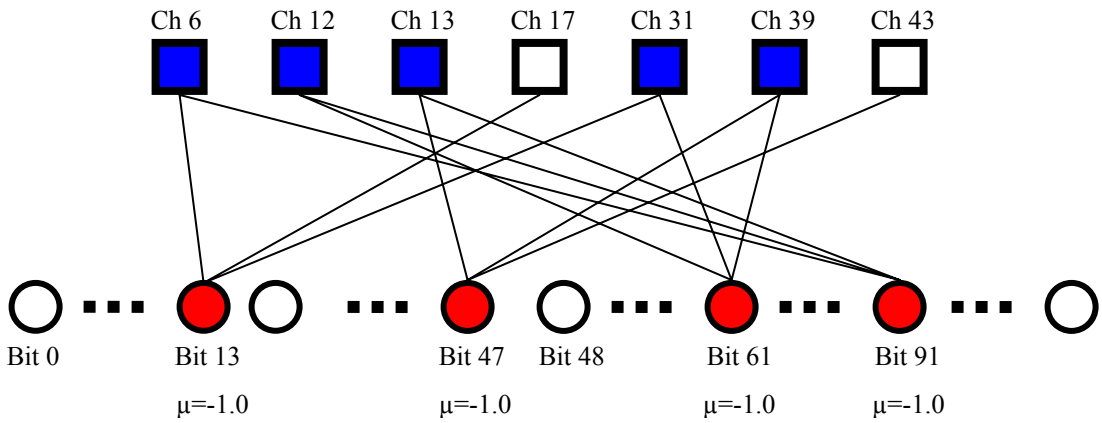

Figure 7. (4,2) trapping set and the proposed bias values.

Although this method might work well for the codes that have relatively few number of trapping sets, the efficiency of the algorithm could be deteriorated as the trapping set count gets large. Consequently, a similar drawback as “divide by N” problem might occur. In order to eliminate this problem, it is necessary to classify the all the trapping sets that have the same size and simulate one set from each class. As the BER figures of trapping sets belonging to the same class are approximately equivalent, simulating only one trapping set is sufficient. Figure 8 illustrates the equivalence of the BER values of different sets that belongs to same class at different SNRs. This will effectively reduce the complexity of the algorithm to be proportional to the number of dominant trapping set classes at very low BER regions.

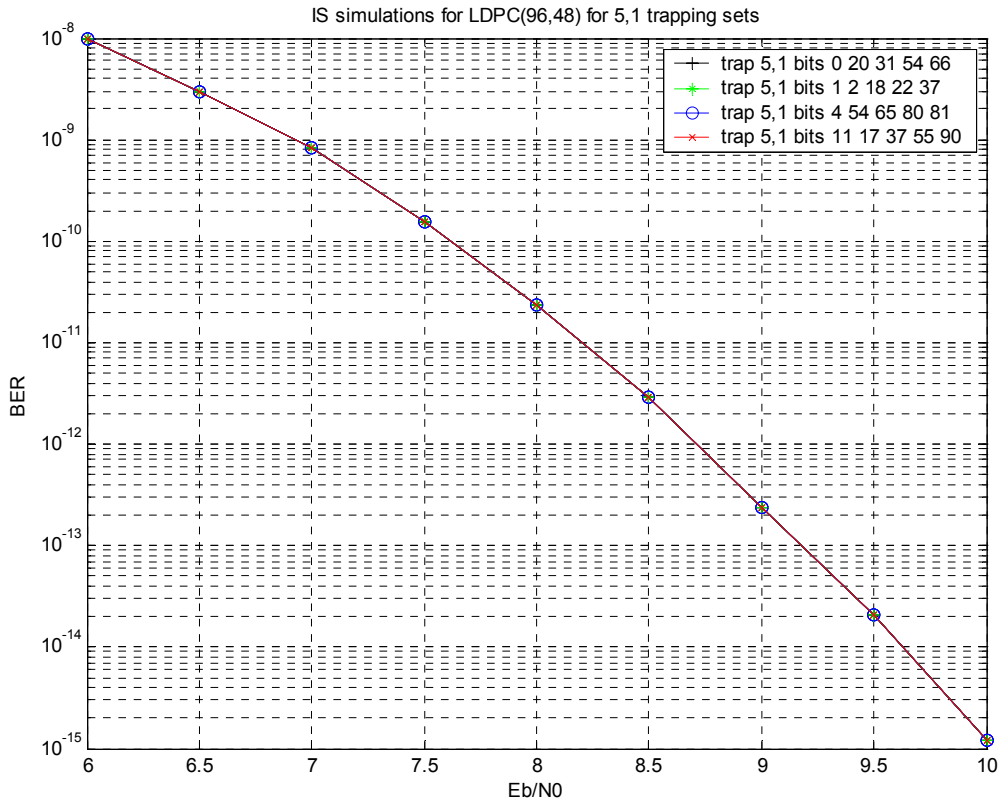

Figure 8. Performance equivalence of various (5,1) trapping sets of (96, 48) LDPC code.

Cumulative BER of each class is obtained by multiplying the simulated BER of one set by the total number of trapping sets in that class. Finally, total BER of the code is calculated as the sum of the BER of each class. Unlike the previous approaches, the proposed technique is independent of the block size of the code and allows the performance evaluation for any BER region with simulation gains in the order of 10^{10}.

IV. SIMULATION RESULTS

In this section, we present the performance of a regular (96, 48) and a (2000, 1000) LDPC code in an AWGN channel using IS technique with the trapping sets. (96, 48) LDPC code can be found in [8] and (2000, 1000) LDPC code is generated using the software provided in [9]. Both of these codes are random constructions and do not contain any cycle-4s.

We have implemented the approximate-min* algorithm presented in [10] which is robust to quantization issues and suitable for an efficient implementation of the check node operation. A 6 bit Sum-Product decoder, 1 sign bit and 5 quantization bits, is used and in all of the simulations the maximum iteration limit is set to 200. All-zero codeword transmission is assumed and BPSK modulation is employed. At high SNR regions, MC simulations were simulated until 20 codeword errors are collected and, therefore, IS simulations are terminated when normalized error estimator become less than or equal to 0.2236 as defined in equation (10).

First, we present the BER performance of different class of trapping sets at low BER regions. Figure 9 and Figure 10 illustrate the region of dominance for various size trapping sets at different SNR regions for (96, 48) and (2000, 1000) LDPC codes respectively.

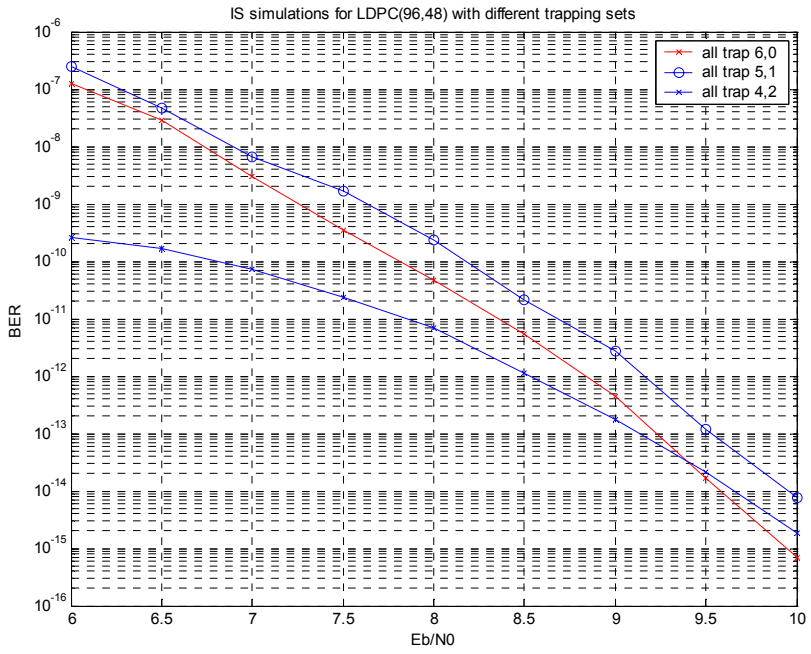

Figure 9. Performance of different class of trapping sets for (96, 48) LDPC code at different SNR regions.

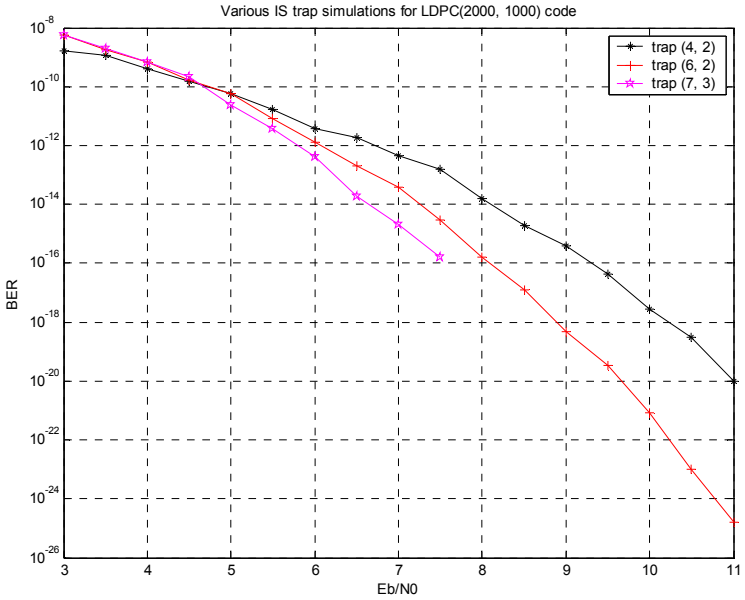

Figure 10. BER performance of different class of trapping sets for (2000, 1000) LDPC code.

From Figure 9 and Figure 10, it can be observed that as the SNR is increasing the number of error bits (x) in dominant trapping sets gets smaller. This is not surprising since as the SNR improves the probability of large number of error bits occurring simultaneously diminishes and trapping sets with smaller error bits dominate the performance. The same observation also can be deducted from Table 2 and we can conclude that at very high SNR (very low BER region), the decoder failures will be due to very small size trapping sets. Therefore, using IS to bias only these trapping sets enable us to simulate very low BER effectively independent of block size of the code.

In the following, we present the BER simulation performance of (96, 48) and (2000, 1000) LDPC codes and compare them to MC simulations. Figure 11 shows the BER performance of a (96, 48) LDPC code with a standard MC simulation and proposed IS method. This code is a very similar code that was studied in [1] and [2] and it has a minimum distance of 6 and there are two codewords at the minimum distance. As it is observed from Figure 11, IS simulation matches closely to MC results. The small performance gap at 7 dB is most likely due to insufficient number of error event simulation involved in Monte Carlo simulation.

Table 5 provides comparison data for IS and MC simulations of (96, 48) code, where it is clear that the IS estimates agree with MC counterparts at all SNR values. In order to measure the proposed technique's gain over MC, Table 5 lists the number of simulation loops in both cases and gives the achieved simulation gain according to (11). As it is evident from Table 5, IS simulations become very efficient as SNR increases and attains unprecedented gain of 10^{11} at BER of 10^{-14}.

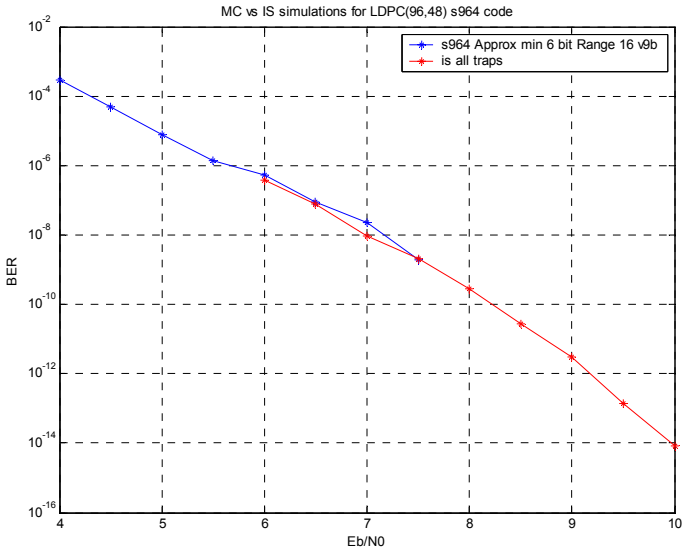

Figure 11. MC and IS BER simulations for (96, 48) LDPC code.

SNR	BER MC	BER IS	MC Loop	IS Loop	Gain
6	5.28×10^{-07}	3.75×10^{-07}	2 M	10K	200
6.5	8.96×10^{-08}	7.58×10^{-08}	12 M	6.5K	1.8×10^{3}
7	2.20×10^{-08}	9.53×10^{-09}	51 M	4.5K	1.1×10^{4}
7.5	1.38×10^{-09}	2.07×10^{-09}	600 M	5K	1.2×10^{5}
8		2.85×10^{-10}	70 G	2.5K	2.8×10^{7}
8.5		2.77×10^{-11}	720 G	2.5K	2.9×10^{8}
9		3.33×10^{-12}	6 T	2.4K	2.5×10^{9}
10		1.01×10^{-14}	2000 T	2.4K	8.3×10^{11}

Table 5. Comparison of MC and IS Simulation results for (96, 48) LDPC code

The proposed IS method also applied to a longer (2000, 1000) LDPC code. Figure 12 and Table 6 presents the IS and MC simulation results. Again, IS results are almost identical to MC counterparts and the proposed method's simulation gain increases rapidly as lower BER performance is evaluated. In this case, IS provides a remarkable simulation gain of 10^{14} at

BER of 10^{-20}. To our best knowledge, this is the first result which is able to simulate the performance of an LDPC code of length 2000 down to BER of 10-20.

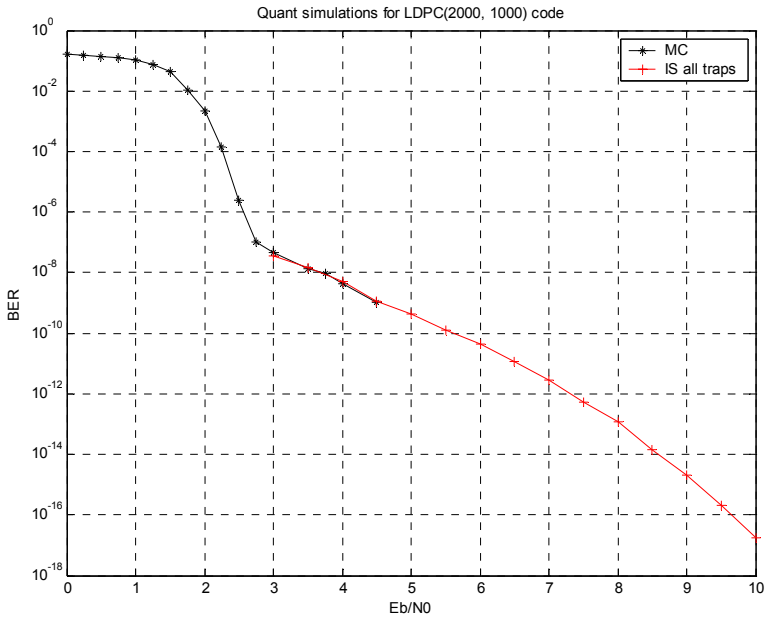

Figure 12. MC and IS BER simulations for (2000, 1000) LDPC code.

SNR	BER MC	BER IS	MC Loop	IS Loop	Gain
3	3.69×10^{-08}	3.56×10^{-08}	1.2 M	59.1K	20.3
3.5	1.49×10^{-08}	1.47×10^{-08}	2.8 M	51.8K	54.1
4	6.46×10^{-09}	6.41×10^{-09}	7.4 M	79.4K	93
4.5	1.58×10^{-09}	1.99×10^{-09}	26.1 M	64.8K	402.8
5		6.00×10^{-10}	9.87×10^{7}	115.8K	852
7		2.42×10^{-12}	1.74×10^{10}	259.8K	6.7×10^{4}
9		2.01×10^{-15}	1.99×10^{13}	2.6 K	7.6×10^{9}
11		5.02×10^{-20}	7.96×10^{17}	1.5K	5.3×10^{14}

Table 6. Comparison of MC and IS Simulation results for (2000, 1000) LDPC code.

V. CONCLUSION

In this work we have presented a highly efficient IS method for performance evaluation of LDPC codes at very low BERs and provided simulation results for (96, 48), and (2000, 1000) LDPC codes. In contrast to previous works in [1] and [2], an error set partitioning based on trapping set bits is considered. By effectively finding and biasing trapping sets, the presented technique accurately estimates performance of any size LDPC code at very low

BERs. Remarkable simulation gains of 10^{11} at BER of 10^{-14} for (96, 48) LDPC code and a gain of 10^{14} at BER of 10^{-20} for (2000, 1000) LDPC code are achieved.

Acknowledgements

We would like to thank Sudhir Gowda, Xiao-Yu Hu, Evangelos Eleftheriou and Y. Katayama for helpful discussions and reviewing the draft text and providing helpful comments and suggestions.

REFERENCES

1. B. Xia and W. E. Ryan, "On importance sampling for linear block codes," *Proc. IEEE International Conference on Communications*, vol. 4, pp. 2904-2908, May 2003.
2. R. Holzlöhner, A. Mahadevan, C. R. Menyuk, J. M. Morris, J. Zweck, "Evaluation of the very low BER of FEC Codes using dual adaptive importance sampling," *IEEE Comm. Letters*, vol. 2, Feb. 2005.
3. T. Richardson, "Error Floors of LDPC Codes," *Proc. 41th Annual Allerton Conf. on Communication, Computing and Control*, Monticello, IL, Oct 2003.
4. B. Xia and W. E. Ryan, "Estimating LDPC codeword error rates via importance sampling," *Proc. Int. Symp. on Information Theory*, June 2004.
5. D. Mackay and M. Postol, "Weakness of margulis and ramanujan-margulis low-density parity-check codes," *Electronic Notes in Theoretical Computer Science*, vol. 74, 2003.
6. MacKay , "Near Shannon limit performance of low density parity check codes," *IEE Electronics Letters*, vol. 32, no. 18, pp. 1645-1655, Aug. 1996.
7. X. –Y. Hu, M. P. C Fossorier, E. Eleftheriou, "On the computation of the minimum distance of low-density parity-check codes," *Proc. IEEE International Conference on Communications*, 2004.
8. D. MacKay, "Encyclopedia of Sparse Graph Codes," available at http://www.inference.phy.cam.ac.uk/mackay/codes/EN/C/96.3.964
9. R. Neal, "Software for Low Density Parity Check (LDPC) codes," available at http://www.cs.toronto.edu/~radford/ldpc.software.html
10. C. Jones, E. Valles, M. Smith, J. Villasenor, **"**Approximate-min***** constraint node updating for ldpc code decoding,**"** *IEEE Military Communications Conference*, vol. 1, pp. 157-162, Oct. 2003.
11. D. Lu and K. Yao, "Improved importance sampling technique for efficient simulation of digital communication systems," *IEEE J. Select. Areas Commun*., vol. 6, pp. 67-75, Jan. 1988.
12. P.J Smith, M. Shafi, G. Hongsheng, "Quick simulation: a review of importance sampling techniques in communications systems," *IEEE J. Select. Areas Commun*., vol. 15, pp. 597-613 , May 1997.
13. J.-C. Chen, D. Lu, J. S Sadowsky, K. Yao, "On importance sampling in digital communications. I. Fundamentals," *IEEE J. Select. Areas Commun*., vol. 11, pp. 289-299, April 1993.
14. J. S. Sadowsky, "A new method for Viterbi decoder simulations using importance sampling," *IEEE Trans. Commun.*, vol. 38, pp. 1341-1351, Sept. 1990.
15. T. Yamane, Y. Katayama, "Bit error rate analysis on iterative two-stage decoding of two dimensional codes by importance sampling**,"** *Proc. IEEE International Conference on Communications*, vol. 5, pp. 3140-3144, May 2003.
16. M. Ferrari, S. Bellini, "Importance sampling simulation of concatenated block codes," Communications, IEE Proceedings- , vol. 147, pp.245-251, Oct. 2000.
17. Winstead Chris, "Fast BER Simulation: Using Importance Sampling in Error Control Systems," http://www.ece.ualberta.ca/~winstead/papers/ImportanceSampling.pdf