# Algoritmos de Decodificação Abrupta para Códigos LDGM

Fernando Pujaico Rivera e Jaime Portugheis

***Resumo*— Desde que Gallager introduziu o algoritmo de decodificação *Bit-Flipping* ($BF$) com decisão abrupta para códigos *Low Density Parity Check* ($LDPC$), outras duas variantes foram propostas por Sipser e Spielman para os códigos conhecidos como "*Expander Codes*". Posteriormente, uma versão da decodificação $BF$ por decisão suave conhecida como decodificação *Modified Weighted BF* ($MWBF$), foi investigada. Este artigo propõe versões modificadas dos algoritmos de Sipser e Spielman. Resultados de simulações para códigos *Low Density Generator Matrix* ($LDGM$) sistemáticos, com comprimento longo mostraram um melhor desempenho da versão proposta. Adicionalmente, para um comprimento médio dos códigos LDGM, resultados de simulações mostraram um desempenho similar à decodificação $MWBF$ com a vantagem de não ser necessário o uso de operações em ponto flutuante.**

***Palavras-Chave*— Códigos Corretores de Erro, Códigos de Matrizes Geradoras de Baixa Densidade, Algoritmos de Decodificação, Decodificação por Decisão Abrupta.**

***Abstract*— Since Gallager introduced *Bit-Flipping* ($BF$) decoding with hard-decision for Low-Density Parity-Check Codes ($LDPC$), other two variants were proposed by Sipser and Spielman for expander codes. Later, a soft-decision version of BF decoding, known as Modified Weighted BF ($MWBF$) decoding, was investigated. This article proposes modified versions of Sipser and Spielman algorithms. Simulation results for long systematic Low-Density Generator Matrix ($LDGM$) codes show a better performance of the proposed versions. Moreover, for moderate length systematic LDGM codes, simulation results show performance similar to that of MWBF decoding with the advantage of not requiring floating-point operations.**

***Keywords*— Error correcting codes , Low density generator matrix codes, Decoding algorithms, Hard decision decoding.**

## I. Introdução

Gallager [1] ao introduzir a classe de códigos $LDPC$, também propôs um algoritmo muito simples de decodificação por decisão abrupta, isto é, um limiar de decisão é usado no valor real observado na saída do canal $AWGN$ (do inglês, Additive Gaussian Noise Channel), gerando um vetor recebido binário. Este algoritmo ficou conhecido como *Bit-Flipping* (BF). Sipser e Spielman estudaram em [2] uma sub-classe dos códigos $LDPC$, chamados "*expander codes*", e propuseram duas formas novas de algoritmos $BF$ com decisão abrupta. Kou et al. propuseram em [3] uma versão da decodificação $BF$ para decisão suave, isto é, um algoritmo que trabalha com o valor real da saída do canal. Esta versão é conhecida como decodificação *weighted BF* ($WBF$). Posteriormente, Zhang e Fossorier [4] modificaram a função flipping do algoritmo $WBF$ introduzindo um fator de correção que depende de um parâmetro a ser otimizado. Esta variante é denominada *Modified* $WBF$ ($MWBF$). Este artigo propõe e estuda duas versões modificadas dos algoritmos de Sipser e Spielman, que a partir de agora serão denominadas *Serial Hard Bit-Flipping* ($SHBF$) e *Parallel Hard Bit-Flipping* ($PHBF$). A seção II descreve o algoritmo BF e a seção III o algoritmo WBF. Modificações do algoritmo WBF são descritas na seção IV. A seção VI propõe modificações do algoritmo de Sipser e Spielman descrito na seção V. A seção VII descreve resultados de simulação dos algoritmos propostos na seção anterior e finalmente a seção VII apresenta conclusões e sugestões de futuros trabalhos.

Fernando Pujaico Rivera, está no Departamento de Comunicações, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas-SP. E-mail: fpujaico@decom.fee.unicamp.br

Jaime Portugheis estava no Departamento de Comunicações, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas-SP. Ele está agora na Faculdade de Tecnologia da Unicamp, E-mail: jaime@ft.unicamp.br

## II. Bit-flipping

O algoritmo de decodificação *Bit-Flipping* ($BF$) foi idealizado por Gallager [1] como uma proposta para identificar e corrigir os erros na transmissão de informação de um vetor $U$ de $K$ bits num canal $BSC$ (do inglês, Binary Symmetric Channel). Depois de que o vetor codificado, $V = U\,G$, transita através do canal com ruido, este chega a seu destino com erros. Este novo vetor binário é chamado de $Z$, e, para corrigir possíveis erros nele, o algoritmo atribui a cada bit $z_n$ de $Z$ uma confiabilidade $E_n$. Esta confiabilidade indica a menor ou maior possibilidade de que esse bit esteja errado. O critério para atribuir as confiabilidades segue a equação

$$E_n = \sum_{m \in \mathcal{M}(n)} s_m. \tag{1}$$

$S$ é o vetor de síndromes de $S = HZ$, em que $H$ é a matriz de verificação de paridade. A matriz $H$ pode ser representada mediante $M$ vetores binários $h_m$ de comprimento $N$, em que $m = 0, 1, \ldots, M-1$. Assim, $s_m = Z\ h_m^t$, na qual $h_m^t$ é um vetor coluna. Deste modo, definimos $\mathcal{M}(n)$ como o conjunto de todos os valores que pode tomar $m$ tal que $h_{m,n} = 1$, ou seja,

$$\mathcal{M}(n) = \{m | h_{m,n} = 1\}. \tag{2}$$

Pode-se entender melhor $\mathcal{M}(n)$ mediante a Figura 1, em que $\mathcal{M}(1) \in \{1, 2\}$ representa o conjunto dos índices dos bits $s_m$ que estejam ligados ao bit $z_1$.

Após estabelecer o critério de geração de confiabilidades $E_n$ para cada bit $z_n$ é necessário gerar também um critério para trocar os bits. O algoritmo $BF$ troca todos os bits com $E_n \geq$

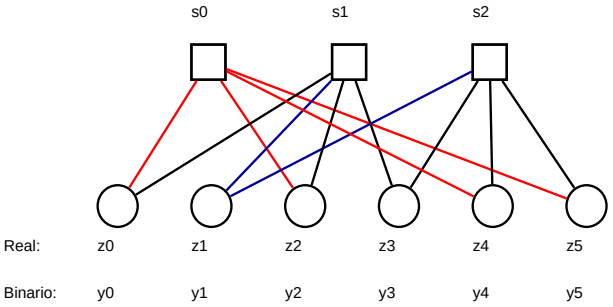


Fig. 1. Gráfico de Tanner de um código (N,K)=(6,3).

$\delta$. Na implementação que segue do algoritmo $BF$, se usa como limiar $\delta$ o valor máximo de $E_n$ no vetor **Z**. O Algoritmo 1, mostrado a seguir, descreve todo o procedimento.

**Algoritmo 1** Decodificação *Bit-Flipping*

1) Inicia-se com $k = 0$ e o vetor de decisão abrupta $Z^{(k)} = (z_1^{(k)}, \cdots, z_n^{(k)}, \cdots, z_N^{(k)}) = \mathbf{Z}$.
2) Calcule-se a síndrome $s_m = \sum_n h_{m,n} z_n^{(k)}$ (mod 2) para $m = 1, \cdots, M$. Se todos os valores são nulos então para-se a decodificação e o vetor $Z^{(k)}$ é entregue.
3) Para $n = 1, \cdots, N$ calcula-se a função de flipping $E_n$ em (1).
4) Identifica-se o conjunto $\{n^*\}$, na qual $n^* = \arg\max_n E_n$.
5) Troca-se de valor todos os bits $z_n^{(k)}$ em que $n \in \{n^*\}$, e faz-se $Z^{(k+1)} = Z^{(k)}$.
6) Se o número máximo de iterações, $IT$, não é atingido, faz-se $k = k+1$ e continua-se no passo 2. Caso contrário para-se a decodificação e se retorna **Z** .

Pelo visto no Algoritmo 1, a decodificação $BF$ pode ser entendida como um algoritmo baseado em confiabilidades, onde o conjunto de bits menos confiáveis são considerados errados e seu valor é trocado. A confiabilidade de cada bit é calculada somando a quantidade de bits de verificação de paridade que ligados ao bit codificado indiquem a existência de um erro. Muito parecido a uma votação, onde só os votos contra são contabilizados para o bit em estudo. Ao final, o conjunto de bits que tenham mais votos contra serão trocados.

## III. O ALGORITMO *Weighted Bit-Flipping*

O algoritmo de decodificação *Weighted Bit-Flipping* $(WBF)$ é uma variante do algoritmo $BF$ [3]. Ele está orientado a trabalhar com um vetor real $Y$ como dado de entrada. Este vetor $Y$ pode ser obtido por exemplo na saída de um canal $AWGN$. Para a geração das confiabilidades $E_n$ de cada bit usa-se uma ponderação de $|y_n|$ e dos valores de $s_m$, usando o valor $+1$ para $s_m = 1$ e o valor $-1$ para $s_m = 0$. Diferentemente do algoritmo $BF$, o algoritmo $WBF$ usa os bits de verificação de paridade corretos, com $s_m = 0$, além dos bits de verificação errados, com $s_m = 1$, e atribui a cada bit $s_m$ um fator de ponderação $w_m$, como segue

$$E_n = \sum_{m \in \mathcal{M}(n)} (2s_m - 1) w_m, \tag{3}$$

em que

$$w_m = \min_{i \in \mathcal{N}(m)} |y_i|. \tag{4}$$

O conjunto $\mathcal{M}(n)$ de (3) é igual ao do algoritmo $BF$. $\mathcal{N}(m)$ é o conjunto de todos os índices dos bits $y_i$ ligados a $s_m$. Por exemplo na Figura 1 o conjunto $\mathcal{N}(0) = \{0, 2, 4, 5\}$. Assim $w_m$ é o valor mínimo derivado do valor absoluto dos $y_i$ ligados ao bit $s_m$. Obtidas as confiabilidades para cada bit, o algoritmo $WBF$ troca o bit que tenha o maior valor. Dado que a confiabilidade usa valores reais, o mais provável é que se troque um só bit.

**Algoritmo 2** Decodificação *Weighted Bit-Flipping*

1) Inicia-se com $k = 0$ e o vetor $Z^{(k)} = (z_1^{(k)}, \cdots, z_n^{(k)}, \cdots, z_N^{(k)})$ com a decisão abrupta do vetor $Y$.
2) Avalia-se a síndrome $s_m = \sum_n h_{m,n} z_n^{(k)}$ (mod 2) para $m = 1, \cdots, M$. Se todos os valores são nulos então para-se a decodificação e o vetor $Z^{(k)}$ é entregue.
3) Para $n = 1, \cdots, N$ avalia-se a função de flipping $E_n$ em (3).
4) Identifica-se $n^* = \arg\max_n E_n$.
5) Troca-se de valor o elemento com índice $n^*$ de $Z^{(k)}$, e faz-se $Z^{(k+1)} = Z^{(k)}$.
6) Se $IT$ não é atingido, coloca-se $k = k + 1$ e vai-se ao passo 2. Caso contrário para-se a decodificação e **Z** é o vetor entregue.

## IV. OS ALGORITMOS: *Modified* $WBF$ E *Improved Modified* $WBF$

O algoritmo *Modified* $WBF$ $(MWBF)$ foi proposto em [4] como uma modificação ao algoritmo $WBF$. O algoritmo $MWBF$ utiliza a função de flipping

$$E_n(\alpha) = \sum_{m \in \mathcal{M}(n)} (2s_m - 1) w_m - \alpha |y_n|. \tag{5}$$

Na equação (5) pode-se ver a confiabilidade descrita pelos bits de verificação de paridade e também pela confiabilidade do bit do vetor de decisão abrupta. Esta última é ponderada com um sinal negativo dado que o comportamento desta é inverso ao da confiabilidade dos bits de verificação de paridade. O fator de ponderação $\alpha > 0$ é colocado devido ao fato de que é provável que a matriz de verificação de paridade, $H$, seja irregular e $\mathcal{M}(n)$ tenha distintas cardinalidades. Por isto é necessário obter experimentalmente o melhor valor para $\alpha$.

$MWBF$ usa o algoritmo 2 para a decodificação, com a diferença de que usa a equação (5) para a geração de confiabilidades $E_n$. O algoritmo de decodificação improvement $MWBF$ $(IMWBF)$ é uma variante do $MWBF$ proposta em [5]. $IMWBF$ vê a decodificação $WBF$ como uma primeira aproximação do algoritmo de passagem de mensagem. Nesse algoritmo a confiabilidade enviada de um bit de verificação de paridade exclui a mensagem do bit que

recebe a informação. Na equação (7) pode-se ver como o $n$-ésimo elemento é excluído do cálculo de $w_{n,m}$, ficando a confiabilidade $E_n$ para o algoritmo $IMWBF$ como

$$E_n(\alpha) = \sum_{m \in \mathcal{M}(n)} (2s_m - 1)w_{n,m} - \alpha|y_n|, \qquad (6)$$

em que

$$w_{n,m} = \min_{i \in \mathcal{N}(m) \setminus n} |y_i|. \qquad (7)$$

$IMWBF$ usa o algoritmo 2 para a decodificação, com a diferença que usa as equações (6) e (7) .

## V. Algoritmos de Sipser e Spielman

Sipser e Spielman propuseram em [2] uma nova maneira de gerar o vetor de confiabilidade e a estimação dos bits errados para um canal com saída abrupta. O primeiro algoritmo que propuseram foi um algoritmo de decodificação sequencial. Ele é mostrado como algoritmo 3.

**Algoritmo 3** Decodificação de Sipser e Spielman

1) Se houver um bit $z_n$ que tem mais bits de verificação de paridade $s_m$ não satisfeitos que satisfeitos, então troca-se este bit.
2) Repita o passo anterior até que nenhuma variável fique com bits de verificação de paridade não satisfeitos.

O algoritmo 3 pode ser descrito como os algoritmos de bit-flipping. Por exemplo, o passo 1 é equivalente a: atribua uma confiabilidade $+1$ quando $s_m = 1$ e uma confiabilidade $-1$ quando $s_m = 0$. Caso contrário, some as confiabilidades de todos os bits $s_m$ ligados a $z_n$. Se a confiabilidade resultante $E_n$ é maior que zero, troque o bit $z_n$. Assim, a função de flipping é dada por

$$E_n = \sum_{m \in \mathcal{M}(n)} (2s_m - 1), \qquad (8)$$

em que $\mathcal{M}(n)$ é como na equação (2).

Note que não se fala nada sobre que bit deve ser trocado. Dado que o algoritmo é sequencial se trocará um só bit por vez e como no algoritmo $BF$ se procederá ao cálculo da síndrome. Se ela for zero, finaliza-se a decodificação. Se for distinta de zero, se aplica o algoritmo 3 até que todas as síndromes sejam nulas, ou se atinja um número máximo de iterações. Sipser e Spielman também propuseram um algoritmo paralelo que consiste em realizar a troca de bits (flipping) de todos os bits com $E_n > 0$ numa única iteração. A partir de agora denominaremos o algoritmo sequencial de Sipser e Spielman como $SSSBF$ e o paralelo de $PSSBF$ .

## VI. Algoritmos BF modificados com decisão abrupta

Nesta seção propõe-se uma modificação do algoritmo de decodificação $SSSBF$. Esta modificação será denominada $PHBF$ (do inglês, *Parallel Hard Bit-Flipping*). Ela trabalha sobre um vetor de entrada binário $Z$ e tem a mesma regra de geração de confiabilidade, $E_n$, do algoritmo $SSSBF$. A regra foi vista na equação (8). A diferença aqui está no critério de troca dos bit errados: é igual ao algoritmo $BF$. Troca-se todos os bits com um $E_n \geq \delta$. Para a implementação atual usou-se como $\delta$ o maior valor de $E_n$. O algoritmo 4 detalha todo o procedimento para a decodificação $PHBF$.

**Algoritmo 4** Decodificação *Parallel Hard Bit-Flipping*

1) Inicia-se com $k = 0$ e o vetor de decisão abrupta $Z^{(k)} = (z_1^{(k)}, \cdots, z_n^{(k)}, \cdots, z_N^{(k)}) = \mathbf{Z}$.
2) Calcula-se a síndrome $s_m = \sum_n h_{m,n} z_n^{(k)}$ (mod 2) para $m = 1, \cdots, M$. Se todos os valores são nulos, então para-se a decodificação e se retorna o vetor $Z^{(k)}$.
3) Para $n = 1, \cdots, N$, avalia-se a função de flipping $E_n$ em (8).
4) Identifica-se o conjunto $\{n^*\}$, na qual $n^* = \arg\max_n E_n$.
5) Troca-se de valor todos os bits $z_n^{(k)}$ em que $n \in \{n^*\}$, e faz-se $Z^{(k+1)} = Z^{(k)}$.
6) Se $IT$ não é atingido, faz-se $k = k+1$ e vai-se ao passo 2. Caso contrário, para-se a decodificação e o vetor $\mathbf{Z}$ é entregue.

Existe uma relação entre o algoritmos $BF$ e $PHBF$ pois ambos utilizam um regra de confiabilidade parecida. A primeira vista poderia parecer que a confiabilidade de $PHBF$ (veja equação 8) é simplesmente uma combinação linear da confiabilidade de $BF$ (veja equação 1). Assim, não deveria existir nenhuma diferença quando se acha a confiabilidade do valor máximo. E portanto também na decodificação de $Z$. Isto é verdade só no caso em que o número de elementos de $\mathcal{M}(n)$ é constante. A verificação disto é fácil de ver se consideramos a quantidade de bits por coluna de $H$, $X(n)$,

$$X(n) = \sum_{m=0}^{M-1} h_{m,n}. \qquad (9)$$

$X(n)$ também por ser interpretado como a quantidade de bits de verificação de paridade $s_m$ que estão conectados à $z_n$.

Se consideramos que se tem $l$ bits de verificação de paridade errados para o bit $z_n$ então o valor da confiabilidade segundo o algoritmo $BF$ seria $E_n^{BF} = l$. Por outro lado, a confiabilidade para o algoritmo $PHBF$ seria $E_n^{PHBF} = (+1)l + (-1)(X(n) - l) = 2l - X(n)$. Unindo ambas idéias, obteríamos a equação

$$E_n^{PHBF} = 2E_n^{BF} - X(n). \qquad (10)$$

A equação (10) mostra claramente que se todos os bits $z_n$ possuem a mesma quantidade de bits de verificação de paridade, isto é, se a matriz $H$ possui um grau de coluna constante, os algoritmos $BF$ e $PHBF$ levam ao mesmo resultado na decodificação.

Uma variante do algoritmo $PHBF$ é o serial hard bit-flipping ($SHBF$). Neste algoritmo, a diferença do $PHBF$ está na troca de um só bit por iteração e no fato de dar-se prioridade ao bit de informação para o caso de matrizes sistemáticas. Assim, o passo 5 do algoritmo $SHBF$ será:

5 Troca-se o valor do primeiro bit de informação $z_n^{(k)}$ em que $n \in \{n^*\}$. Se este conjunto não contem nenhum bit de informação, troca-se o valor do primeiro bit de verificação de paridade encontrado.

## VII. Resultados de simulações

Nas simulações se usarão dois tipos de matrizes, as $LDGM$ e $EG-LPDC$ [3] (*Euclidean Geometry - LDPC*). No caso das matrizes $LDGM$ a matriz geradora $G$ tem a seguinte forma, $G = [P\ I_K]$. No caso das matrizes $EG-LDPC$ se usarão matrizes de tipo I [3]. Diferentemente das $LDGM$, as $EG-LDPC$ são matrizes regulares com uma quantidade de uns por linha e coluna da matriz de verificação de paridade muito superior a das $LDGM$ e com a vantagem de que as $EG-LDPC$ são matrizes cíclicas. Assim, a geração e verificação de paridade é mais simples de implementar. Com respeito a notação, se usará $d_v$ e $d_c$ para indicar a quantidade de uns por coluna e linha respectivamente da matriz de verificação de paridade $H$, e se usará $d_v^*$ e $d_c^*$ para indicar a quantidade de uns por coluna e linha, respectivamente, da matriz de paridade $P$. Além de obter-se os gráficos dos distintos algoritmos $BF$ também se mostrará os gráficos da curva estimada por Garcia-Frias e Zhong em [6], para a taxa de erro do bit ($BER$) na região inferior de erro para matrizes $LDGM$. Esta curva é chamada nos gráficos como "Lower Bound" e depende da quantidade de uns por linha $d_v^*$ da matriz $P$. Quando um algoritmo $BF$ atinge uma quantidade $IT$ máxima de iterações, ao reparar os bits do vetor codificado recebido, o algoritmo desiste e entrega na sua saída o mesmo vetor entregado a ele na sua entrada sem nenhuma modificação, e subtrai dele o vetor de informação. Para códigos $LDGM$ isto é simples dado que o vetor de informação é uma porção do vetor codificado. No caso dos códigos $EG-LDPC$ as simulações para mostrar a $BER$ contabilizam os bits errados no vetor codificado, não do vetor de informação como é mais comum. Isto, no caso dos códigos $EG-LDPC$ é uma boa aproximação estatística dado que a matriz de verificação de paridade é uma matriz regular e todos os bits decodificados por ela tem a mesma possibilidade de serem decodificados corretamente. Assim a estatística do total (vetor codificado) é aproximadamente igual a estatística de uma porção dele (vetor de informação). Esta tendência é maior quanto maior seja a quantidade $N$ de bits codificados.

O gráfico da Figura 2 apresenta as comparações de todas as versões do algoritmos $BF$ estudados neste artigo, incluindo a curva limite proposta por Garcia-Frias e Zhong [6], e o limite de Shannon para um canal $BSC$ ou $BI-AWGN$ (do inglês, Binary Input AWGN) com decisão abrupta. É usado para a codificação uma matriz $LDGM$ com $N = 30000$ bits codificados e $K = 20000$ bits de informação e taxa do código $R_c = 0.6667$. A matriz $P$ da matriz sistemática foi gerada aleatoriamente tentando conservar constante só a quantidade de uns por linha, obtendo-se um $d_v^* = 9$. Para todos os algoritmos $BF$ se usou uma quantidade máxima de iterações $IT = 300$.

Na Figura 2 se mostra que o melhor desempenho, é obtido pela decodificação $PHBF$, e as decodificações $SHBF$ e

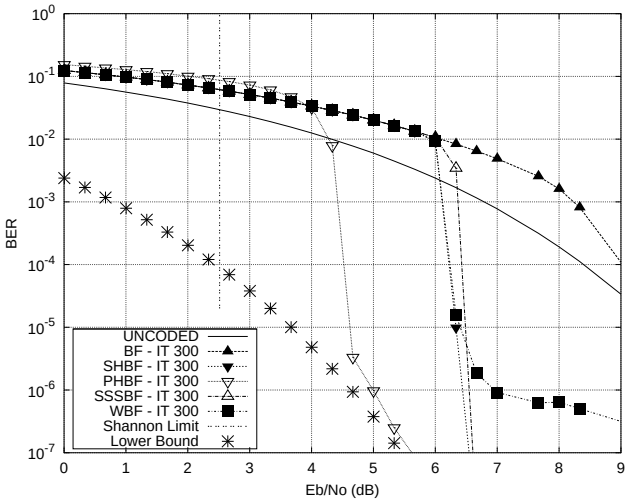


Fig. 2. Simulação dos algoritmos bit-flipping num canal $BI-AWGN$ com uma $LDGM$ de N=30000 bits de informação, K=20000 bits codificados, 300 iterações como máximo, $d_v^* = 9$ e $d_c^* = aleatorio$.

$WBF$ ficam com um pobre desempenho para uma estimação de um 1% (isto é $IT = 300$) dos bits errados como máximo por vetor codificado. A Figura 3 é a mesma que a Figura 2 com a diferença que se usa uma estimação de 10% (isto é $IT = 3000$) de bits errados como máximo por vetor codificado. Pode-se observar na Figura 3 que o algoritmo $SHBF$ se aproxima do desempenho do algoritmo $PHBF$ já que a sua capacidade de correção aumenta com o número de iterações e o algoritmo $PHBF$ já atingiu o seu melhor desempenho.

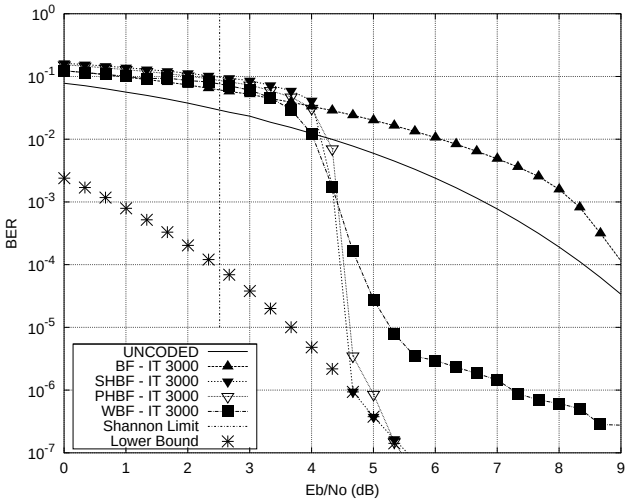


Fig. 3. Simulação dos algoritmos bit-flipping num canal $BI-AWGN$ com uma $LDGM$ de N=30000 bits de informação, K=20000 bits codificados, 3000 iterações como máximo, $d_v^* = 9$ e $d_c^* = aleatorio$.

Um ponto de estudo importante é o comportamento dos algoritmos $BF$ num canal $BI-AWGN$, usando na codificação e decodificação códigos de bloco com matrizes $EG-LDPC$ de tipo I [3]. Como já se tem visto na equação (10), é de

esperar que os algoritmos $BF$ e $PHBF$ tenham o mesmo desempenho. O gráfico da Figura 4 mostra o desempenho dos algoritmos bit-flipping para um código de bloco com uma matriz geradora e de verificação de paridade criada a partir de uma $EG(m, 2^s)$, com $m = 2$, $s = 5$ e usando o polinômio primitivo $p(X) = X^{10} + X^3 + 1$. O código terá $K = 781$, $N = 1023$ e $J = 1023$. Para a simulação se usa como máximo $IT = 50$ iterações, aproximadamente 5% dos $N = 1023$ bits codificados. Este código tem um $d_v = d_c = 32$, número muito superior ao valor de 9 que foi usado nos códigos $LDGM$. Isto se vê refletido no melhor desempenho de todas as curvas na simulação. É importante ressaltar que quanto maior seja o valor de $d_v$ melhor desempenho terão os algoritmos de decodificação, mas também terão maior complexidade na decodificação.

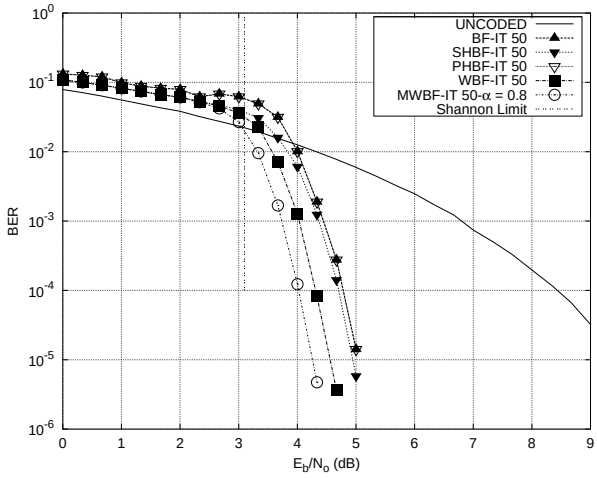


Fig. 4. Simulação dos algoritmos bit-flipping num canal $BI - AWGN$ com uma $EG - LDPC$ de tipo I, com K=781 bits de informação, N=1023 bits codificados, 50 iterações como máximo, $d_v = 32$ e $d_c = 32$.

Pelo visto até agora os algoritmos $PHBF$ e $SHBF$ tem um bom desempenho em matrizes $LDGM$ com um baixo grau de linha ($d_v$), superando em algumas situações a algoritmos com decisão suave. Isto não se conserva em matrizes $EG - LDPC$ por causa da regularidade na quantidade de uns por linha na matriz de verificação de paridade, o que faz com que seu desempenho seja igual ao algoritmo $BF$. Isto pode-se ver facilmente na equação (10), onde uma confiabilidade é uma função linear da outra, pelo qual o máximo e o mínimo estão na mesma posição em ambos casos, neste caso o incremento na complexidade de $PHBF$ e $SHBF$ não outorga nenhuma vantagem.

É importante notar que os códigos $LDPC$ irregulares não sistemáticos não foram testados neste trabalho porque se quer medir o desempenho dos algoritmos para um numero fixo (regular ou aproximadamente regular) de uns por coluna na matriz de verificação de paridade. Neste sentido se ressalta que o desempenho dos algoritmos depende deste valor como pode-se ver comparando a Figura 3 com $d_v^* = 9$ e a Figura 4 com $d_v = 32$.

## VIII. Conclusões

Este artigo apresentou os algoritmos de decodificação $PHBF$ e $SHBF$ como modificações nas formas paralela e sequencial do algoritmo de decodificação $BF$, proposto por Gallager. Estas modificações também estão baseadas no método de geração de confiabilidade do algoritmo modificado $BF$ proposto por Sipser e Spielman.

Para um comprimento moderado dos códigos $LDGM$ se observa que o algoritmo de decodificação $MWBF$ não apresenta nenhuma melhora significativa de desempenho quando comparado com a decodificação $PHBF$. Isto pode ser atribuído ao fato de que estes códigos tem uma distribuição irregular no peso das colunas da matriz de verificação de paridade $H$ ($N - K$ colunas de peso 1 e $K$ colunas de peso $d_v^*$) e o fator de ponderação da decodificação $MWBF$ atua com o mesmo valor sobre confiabilidades de distintos pesos por coluna de $H$.

Outro ponto a observar é que os algoritmos de decodificação $SHBF$ e $PHBF$ apresentam um desempenho semelhante com diminuição da taxa de erro quando comparados ao algoritmo $WBF$, este último sendo mais custoso em complexidade computacional por causa de seus cálculos em ponto flutuante.

Fica para futuros trabalhos investigar se existe uma maneira de decodificação $MWBF$ com melhor desempenho que a decodificação $WBF$ e a $PHBF$ considerando matrizes $LDGM$ sistemáticas.

## Referências

[1] R. Gallager, "Low-Density Parity-Check Codes." Cambridge,Massachusetts: MIT Press, 1963.

[2] M. Sipser e D. Spielman, "Expander Codes," *IEEE Trans. Inform. Theory*, v. 42, n. 6, pp. 1710-1722, Novembro 1996.

[3] Y. Kou, S. Lin e M. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inform. Theory*, v. 47, n. 7, pp. 2711-2736, Novembro 2001.

[4] J. Zhang e M. Fossorier, "A Modified Weighted Bit-Flipping Decoding of Low-Density Parity-Check Codes," *IEEE Commun. Lett.*, v. 8, n. 3, pp. 165-167, Março 2004.

[5] M. Jiang, C. Zhao, Z. Shi e Y. Chen, "An Improvement on the Modified Weighted Bit-Flipping Decoding Algorithm for LDPC Codes," *IEEE Commun. Lett.*, v. 9, n. 9, pp. 814-816, Setembro 2005.

[6] J. F. Garcia-Frias e W. Zhong, "Approaching Shannon performance by iterative decoding of linear codes with low-density generator matrix," *IEEE Commun. Lett.*, v. 7, n. 6, pp. 266-268, Junho 2003.