UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Elettronica

Ottimizzazione di codici LDPC con decodifica basata sulla metrica di Lee

LDPC code optimization with Lee metric-based decoding

Candidato:
Renat Kermenov

Relatore:
Prof. Franco Chiaraluce

Correlatore:
Ing. Paolo Santini
Ing. Massimo Battaglioni

Anno Accademico 2019-2020

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Elettronica

Ottimizzazione di codici LDPC con decodifica basata sulla metrica di Lee

LDPC code optimization with Lee metric-based decoding

Candidato:
Renat Kermenov

Relatore:
Prof. Franco Chiaraluce

Correlatore:
Ing. Paolo Santini
Ing. Massimo Battaglioni

Anno Accademico 2019-2020

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Elettronica
Via Brecce Bianche – 60131 Ancona (AN), Italy

Indice

Elenco delle figure

Introduzione

Il presente lavoro di tesi si basa sull'attività di tirocinio svolta tra maggio e settembre 2020. Nella prima parte dell'attività mi sono concentrato sullo studio delle nozioni preliminari della teoria dei codici e quello che riguarda lo stato dell'arte sulla metrica di Lee. Acquisite le nozioni di base, ho iniziato lo studio della nuova famiglia di codici (i codici LLDPC) introdotta [1] e del relativo algoritmo di decodifica Symbol Flipping. Gli obbiettivi dell'attività di tirocinio come della presente trattazione sono due:

1. Ottimizzare le prestazioni di tali codici studiandone la struttura della matrice di parità H e l'algoritmo di decodifica.

2. Effettuare simulazioni su un canale noto, ad esempio il BSC.

Il capitolo 1 è un capitolo introduttivo e riassume le basi della teoria dei codici con attenzione particolare ai codici lineari. Per una maggiore completezza si rimanda a [2] e a [3].
Il capitolo 2 introduce la metrica di Lee [4] e considerazioni sul tipo di canale in cui è vantaggioso adoperarla [5]. Vengono, inoltre, introdotti i codici LDPC nella metrica di Lee, ovvero gli LLDPC e l'algoritmo di decodifica Symbol Flipping, su cui poi verterà il capitolo 3.
Nel capitolo 3 si presenta un'analisi più dettagliata nella matrice di parità H di un codice QC-LLDPC, in particolare si prende in considerazione il fenomeno delle cancellazioni. In aggiunta viene studiato l'algoritmo di decodifica Symbol Flipping, dando particolare attenzione alla scelta della soglia di decodifica β.
Infine, nel capitolo 4, si prova a simulare i codici definiti nella metrica di Lee su un canale BSC. Per fare ciò, ogni simbolo della parola di codice q-ario deve essere mappato su una sequenza binaria, ma come vedremo non è una operazione sempre possibile.

Capitolo 1

Codifica di Canale

1.1 Introduzione

Il trasmettitore vuol inviare un messaggio u e lo codifica per ottenere una parola di codice c che trasmette sul canale affetto da rumore. Il ricevitore riceve il messaggio y che potrebbe non essere più una parola di codice essendo affetta dal rumore introdotto dal canale. Deve quindi decodificare y per ottenere la parola di codice $\hat{c}$ a cui e associato il messaggio $\hat{u}$. Se la decodifica va a buon fine risulta che $\hat{c} = c$ e $\hat{u} = u$. Lo schema è mostrato in Figura 1.1.

Figura 1.1: Modello base di codifica di canale.

Un esempio di canale è il *Binary Symmetric Channel* schematizzato in Figura 1.2. Ogni simbolo della parola di codice è rappresentato da un simbolo binario ed ha una probabilità p di essere equivocato. Il termine *Symmetric* viene usato in quanto la probabilità che avvenga lo scambio è lo stesso, sia che l'ingresso sia 0 che 1.

In generale l'alfabeto usato per definire il messaggio e la parola di codice non è binario ed il canale scambia i simboli della parola di codice con qualsiasi altro simbolo dell'alfabeto: si parla di canale simmetrico *q-ario*.

L'azione del canale può essere descritta come l'aggiunta (componente per componente) di un vettore errore e alla parola di codice che produce un'uscita $y = x + e$, in cui si assume che e abbia componenti che assumono valori discreti(in generale con è cosi, si veda per esempio il canale $AWGN$). Il vettore e può dipendere da x, anche se di

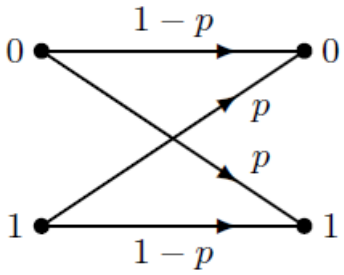

Figura 1.2: Schema del canale BSC.

norma si ricorre all'assunzione che e sia una variabile indipendente; in quest'ultimo caso di parla di *rumore additivo*; lo schema è riportato in Figura 1.3.

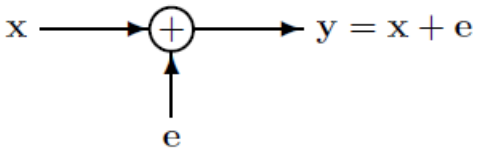

Figura 1.3: Rumore additivo per un canale *q-ario*.

1.2 Codici a blocco

Un codice a blocco (n, M) definito su un alfabeto F è un sottoinsieme non vuoto, che chiamiamo $\mathcal{C}$, di F^n che ha cardinalità (numero totale di elementi di un insieme) M.
La lunghezza dell'informazione del codice $\mathcal{C}$ è definita da $k = \log_{|F|}(M)$ ed il suo *rate* è $R = \frac{k}{n}$. M si dice dimensione del codice mentre n è la sua lunghezza.

Per misurare la distanza tra le parole di codice si definisce una metrica tra le parole di codice $d(c_1, c_2)$ su F^n.

Definizione 1 *Le condizioni affinché $d(c_1, c_2)$ sia una metrica sono:*

1. *$d(c_1, c_2) \geq 0$, e $d(c_1, c_2) = 0$ se e solo se $c_1 = c_2$*
2. *Simmetria: $d(c_1, c_2) = d(c_2, c_1)$ per tutti gli elementi c_1, c_2*
3. *Disuguaglianza triangolare: $d(c_1, c_3) \leq d(c_1, c_2) + d(c_2, c_3)$ per tutti gli elementi c_1, c_2, c_3.*

Definizione 2 *Si definisce distanza di Hamming $d_H(c_1, c_2)$ il numero di posizioni in cui i vettori c_1 e c_2 differiscono.*

Definizione 3 *Si definisce peso di Hamming $w_H(c)$ il numero di posizioni in cui c non è nullo.*

Si può facilmente verificare che la distanza di Hamming è una metrica ed è la più comunemente usata nella teoria dei codici.

Essendo $\mathcal{C}$ un insieme finito, si possono calcolare tutte le distanze $d(c_1, c_2)$ sugli elementi di $\mathcal{C}$. In questo elenco di distanze, si prende il valore minimo e lo si indica con $d(\mathcal{C})$. Questo parametro viene comunemente chiamato *Distanza Minima*

$$d(\mathcal{C}) = min(d(c_1, c_2)|c_1, c_2 \in \mathcal{C}, c_1 \neq c_2).$$

La distanza minima è un parametro molto importante in quanto fornisce il numero minimo di errori necessari per passare da una parola di codice ad un'altra.
Un codice (n, M) con distanza minima d viene spesso indicato con (n, M, d), (in alternativa si può usare la notazione (n, k, d)).

Quando una parola di codice entra nel canale vengono introdotti degli errori su alcuni elementi della parola. Possiamo *correggere* tali errori trovando la parola di codice la cui distanza di Hamming dal vettore ricevuto sia la più piccola possibile. In generale diciamo che il codice può *correggere* fino a τ errori se, in qualsiasi modo sono disposti i τ o meno errori nella parola di codice c, il vettore più vicino rimane c. Diciamo che una codice può *rilevare* fino a s errori se cambiando una parola di codice in al massimo s posizioni questa non diventa un'altra parola di codice.

Teorema 1 *Un codice $\mathcal{C}$ può rilevare fino ad s errori se $d(C) \geq s + 1$.*
Un codice $\mathcal{C}$ può correggere fino a τ errori se $d(C) \geq 2\tau + 1$.

Per la dimostrazione del Teorema 1 si rimanda al riferimento [3].
Per capire meglio il significato di questa decodifica si immagina di assegnare ad ogni parola di codice un punto su un piano ($n = 2$) e di tracciare attorno ad ognuno di questi punti una circonferenza di raggio $\tau \leq \frac{d(C)-1}{2}$; in questo modo all'interno di ogni circonferenza giace solo il punto c. La parola ricevuta y è un punto qualsiasi del piano e sarà decodificata con la parola di codice c se si trova all'interno della circonferenza associata a c, si veda la Figura 1.4.

1.3 Codici Lineari

Sia $GF(q)$ un campo finito (di Galois) di dimensione q. Se q è un numero primo $GF(q)$ coincide con l'anello di interi mod q, detto anche Z_q.

Definizione 4 *Un codice $\mathcal{C}$ denotato da (n, M, d) su un campo $F = GF(q)$ è detto lineare se $\mathcal{C}$ è un sottospazio lineare di F^n; in altre parole per ogni coppia $c_1, c_2 \in \mathcal{C}$ ed ogni coppia di scalari $a_1, a_2 \in F$ si ha che $a_1c_1 + a_2c_2 \in \mathcal{C}$.*

Se k è la dimensione di $\mathcal{C}$, diciamo che $\mathcal{C}$ è un codice lineare $[n, k, d]$ su F. La differenza $n - k$ è detta *ridondanza* di $\mathcal{C}$.

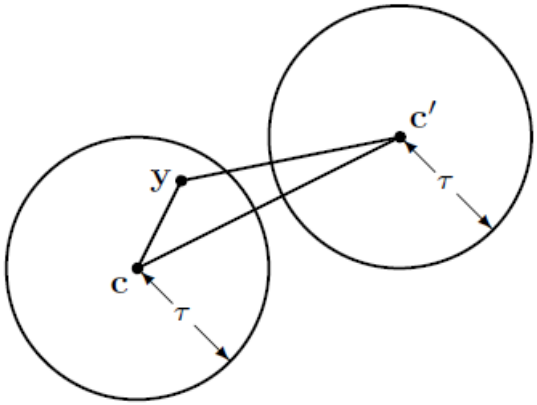

Figura 1.4: Sfere di raggio $\tau = \lfloor \frac{d(C)-1}{2} \rfloor$ centrate nelle parole di codice c e c'.

Ogni base di un codice lineare $\mathcal{C}$ con parametri $[n, k, d]$ su $F = GF(q)$ contiene k parole e la combinazione lineare di queste genera tutti gli elementi di $\mathcal{C}$.
Quindi $|\mathcal{C}| = M = q^k$ ed il rate del codice è $R = \frac{\log_q M}{n} = \frac{k}{n}$.

Definizione 5 *Una matrice generatrice di un codice lineare $[n, k, d]$ su F è una matrice $k \times n$ le cui righe formano una base del codice e la indichiamo con G.*

Proposizione 1 *Sia $\mathcal{C}$ un codice lineare $[n, k, d]$ su F. Allora*

$$d = min(w(c)|c \in \mathcal{C} \setminus \{0\}).$$

1.3.1 Codifica di un codice lineare

Sia $\mathcal{C}$ un codice lineare su F e G la sua matrice generatrice. Possiamo codificare un vettore di informazione $u \in F^k$ su una parola di codice $\mathcal{C}$ usando un mapping $F^k \to \mathcal{C}$ definito da

$$u \to uG.$$

Assumendo che $rank(G) = k$, si tratta di una funzione biunivoca.
Una matrice generatrice $k \times n$ è detta *sistematica* se ha la forma

$$(I|A)$$

dove I è una matrice identità $k \times k$ e A una matrice $k \times (n - k)$.
Usando una forma sistematica $G = (I|A)$ per la codifica, il mapping $u \to uG$ assume la forma $u \to (u|uA)$; in questo modo i primi k elementi della parola di codice formano la sequenza di informazione.

1.3.2 Matrice di parità

Definizione 6 *Sia $\mathcal{C}$ un codice lineare $[n, k, d]$ in F. Una matrice di parità di $\mathcal{C}$ è una matrice H $r \times n$ su F tale che per ogni $c \in F^n$ (il valore r verrà discusso in seguito)*

$$c \in \mathcal{C} \iff Hc^T = 0.$$

In altre parole, il codice $\mathcal{C}$ corrisponde al kernel destro di H in F^n. Dalla nota relazione tra la dimensione del kernel di una matrice ed il suo rango

$$rank(H) = n - dim(kerH) = n - k.$$

Quindi, quando le righe di H sono linearmente indipendenti abbiamo che $r = n - k$. Sia G una matrice generatrice $k \times n$ di G. Le righe di G sono lo *span* del $ker(H)$ ed, in particolare

$$HG^T = 0.$$

Inoltre,

$$dim(ker(H)) = n - rank(G) = n - k.$$

Le righe di H dunque, sono lo *span* del $ker(G)$. Quindi, una matrice di parità di un codice lineare può essere individuata determinando la base del kernel di una delle matrici generatrici del codice,
Nel caso in cui $G = (I|A)$ è una matrice sistematica possiamo assumere la matrice $H = (-A^T|I)$ come matrice di parità.

1.3.3 Decodifica di un codice lineare

Si è già visto che in decodifica è necessario trovare la parola di codice più vicina al vettore ricevuto dal canale e si vuole ora contestualizzare tale concetto ai codici lineari.
Sia $\mathcal{C}$ un codice lineare $[n, k, d]$ su $F = GF(q)$; allora possiamo considerare due approcci equivalenti per la decodifica:

* Data una parola ricevuta $y \in F^n$, trovare una parola di codice $c \in \mathcal{C}$ che minimizzi il valore $d(y, c)$.

* Data una parola ricevuta $y \in F^n$, trovare il vettore $e \in F^n$ che minimizzi $w(e)$ tale che $y - e \in \mathcal{C}$.

Anche se entrambi i modi sono equivalente il secondo, detto *decodifica di sindrome*, è il più efficiente ed anche il più utilizzato.

1.4 Codici Low Density Parity Check (LDPC)

Nel 1948 Cloude Shannon dimostrò l'esistenza di un limite superiore per il raggiungimento delle massime prestazioni di un codice. Se il codice ha un rate $R < C$,

dove C è la capacità del canale, allora il codice riesce a correggere con un margine di errore arbitrariamente piccolo.
La prima famiglia di codici che riuscì ad avvicinarsi a tale limite furono i *Codici Turbo*, introdotti nel 1993. Si scoprì presto che anche i codici LDPC riuscivano ad avvicinarsi a tale limite. Si tratta di codici lineari a blocco definiti da una matrice di parità sparsa (con tanti 0 e pochi 1) e proposti per la prima volta da Robert G. Gallager nella sua tesi di dottorato nel 1963 [6]. Gallager, inoltre, propose un algoritmo di decodifica chiamato *Bit Flipping* il quale scambia i bit della parola ricevuta fino a quando non è soddisfatta la condizione $Hy^T = 0$ (o equivalentemente $He^T = 0$) oppure non si è raggiunto il numero massimo di iterazioni. Sono state proposte diverse varianti di questo algoritmo, in base alla struttura della matrice di parità. Infatti esistono diversi modi per costruire la matrice di parità H dei codici LDPC. Ad esempio H può essere costruita in modo semi-random oppure in modo strutturato. I codici QC-LDPC, descritti in seguito, sono un esempio di codice LDPC che hanno una matrice di parità H costruita in modo strutturato.

1.4.1 Quasi Cyclic-LDPC (QC-LDPC)

Si tratta di una famiglia di codici LDPC largamente usata in diverse applicazioni, in particolare per le implementazioni hardware. Sono caratterizzati da una matrice di base B di dimensione $R \times C$. Ogni elemento della matrice $b_{i,j}$ è un numero intero tale che $b_{i,j} \geq -1$. La matrice H è costruita a partire dalla matrice B sostituendo tutti gli elementi $b_{i,j}$ con matrici circolanti, ottenute shiftando verso destra una matrice identità di dimensione $z \times z$ di $b_{i,j}$ posizioni. Se $b_{i,j} = -1$ viene sostituito da una matrice di zeri di dimensione $z \times z$. In questo modo H avrà dimensione $(Rz) \times (Cz)$.

Capitolo 2

Metrica di Lee

2.1 Introduzione

Lo studio dei codici per la correzione degli errori si concentra prevalentemente sui codici che si basano sulla metrica di Hamming. Tali codici sono progettati per correggere un certo numero di errori che possono essere stabiliti a priori nei codici algebrici oppure in termini probabilistici come nel caso di decoder come il Bit Flipping, dove per errore si intende la variazione di un elemento della parola trasmessa (o equivalentemente un elemento non nullo del vettore errore e). Qualsiasi sia il valore del coefficiente non nullo di e, anche nel caso di un canale *q-ario*, il peso assegnato a tale variazione è sempre lo stesso, ovvero 1.

Esempio 1 *Prendiamo in esame un vettore c con coefficienti in Z_5. c viene trasmesso su un canale 5-ario che introduce un errore e ed y è il vettore ricevuto.*

$$c = \begin{pmatrix} 4 & 1 & 0 & 3 \end{pmatrix}$$

$$y = \begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix}$$

$$e = \begin{pmatrix} 2 & 0 & 1 & 2 \end{pmatrix}$$

$$d_H(c, y) = w_H(e) = 3$$

La parola di codice varia 3 volte rispetto ad y essendo il vettore errore non nullo in 3 posizioni. La distanza di Hamming tra c ed y non tiene conto dell'entità delle variazioni (il peso di Hamming del vettore errore non tiene conto dell'ampiezza delle sue componenti).

Chiang e Wolf [5] per primi hanno mostrato l'importanza di dare un peso diverso ai diversi valori degli elementi del vettore errore, quando questi ultimi non sono equiprobabili. In particolare lo hanno fatto usando la *metrica di Lee*, introdotta nel 1958 [4] da C. S. George Lee.

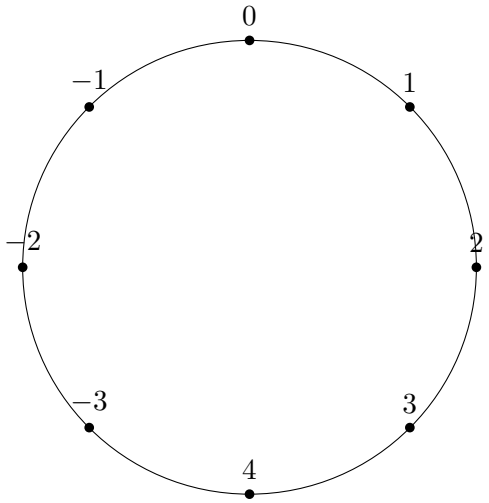

Figura 2.1: Metrica di Lee per gli interi mod q, per $q = 8$,

2.2 Peso di Lee e distanza di Lee

Sia Z_q un anello di interi mod q. Il *peso di Lee* di un elemento $\alpha \in Z_q$ è definito da

$$w_L(\alpha) = \min(\alpha, q - \alpha).$$

In alternativa possiamo rappresentare gli elementi di Z_q nel seguente modo

- $-\frac{q-1}{2}, -\frac{q-3}{2}, \ldots, -1, 0, 1, \ldots, \frac{q-1}{2}$ se q è dispari,
- $-\frac{q-2}{2}, \ldots, -1, 0, 1, \ldots, \frac{q-2}{2}, \frac{q}{2}$ se q è pari.

Considerando un generico elemento $\alpha \in Z_q$, rappresentato per comodità nel precedente modo, il suo peso di Lee può essere individuato prendendone il valore assoluto:

$$w_L(\alpha) = |\alpha|.$$

Il *peso di Lee* di un vettore $c = (c_1, \ldots, c_n) \in Z_q^n$ è la somma dei pesi di Lee delle sue componenti

$$w_L(c) = \sum_{i=1}^{n} |c_i|.$$

La *distanza di Lee* tra due vettori $x, y \in Z_q^n$ è definita come

$$d_L(x, y) = w_L(x - y) = \sum_{i=1}^{n} w_L(x_i - y_i).$$

Si può dimostrare che anche la distanza di Lee è una metrica.
Disponiamo gli elementi su una circonferenza come in Figura 2.1 dove ogni arco indica la distanza tra 2 elementi. La *distanza di Lee* tra 2 elementi è la distanza minima che bisogna percorrere sulla circonferenza tra un elemento ed un altro.

Esempio 2 *Si riprendono in esame gli stessi dati dell'Esempio 1, ma questa volta utilizzando la metrica di Lee*

$$c = \begin{pmatrix} 4 & 1 & 0 & 3 \end{pmatrix}$$

$$y = \begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix}$$

$$e = \begin{pmatrix} 2 & 0 & 1 & 2 \end{pmatrix}$$

$$d_L(c, y) = w_L(e) = 5.$$

In questo caso, alle 3 componenti non nulle del vettore errore vengono assegnati valori diversi in base al loro peso di Lee.

2.2.1 Metrica di Lee e scelta del canale

Si vuole discutere il problema della scelta di una metrica per un dato canale o trovare un canale adattato (matched) ad una certa metrica. Tale problema dipende dallo schema di decodifica, per questo si assume di usare una decodifica ML. Inoltre si suppone che il vettore errore sia indipendente dalla parola di codice trasmesso ed il canale è lineare. Per tali canali

$$Pr(y|c) = Pr(y - c|0) = Pr(e)$$

dove y è il vettore ricevuto, c è la parola di codice ed e è il vettore errore.

Definizione 7 *Una metrica ed un canale discreto senza memoria sono detti adattati (metched) ad una decodifica ML se lo schema di decodifica che associa il vettore ricevuto alla parola di codice più vicina, fornice sempre la parola di codice più probabilmente trasmessa.*

Definizione 8 *Una metrica ed un canale discreto senza memoria sono detti strettamente adattati (strictly matched) ad una decodifica ML se, dati i vettori errore e ed e' risulta che*

$$w(e) < w(e') \iff Pr(e) > Pr(e').$$

In [5] viene mostrato che le 2 definizioni sono equivalenti.

Teorema 2 *Un canale discreto, simmetrico e senza memoria come mostrato in Figura 3.3 è rigorosamente adattato alla metrica di Lee per la decodifica ML se e solo $p_i = \frac{p_1^i}{p_0^{i-1}}$ e $p_0 > p_1$.*

Per la dimostrazione del Teorema 2 si rimanda al [5].

Corollario 1 *Se un canale discreto, simmetrico e senza memoria come mostrato in Figura 3.3 è strettamente adattato alla metrica di Lee allora $p_0 > p_1 > p_2 > \cdots > p_M$, dove $M = \lfloor \frac{q}{2} \rfloor$.*

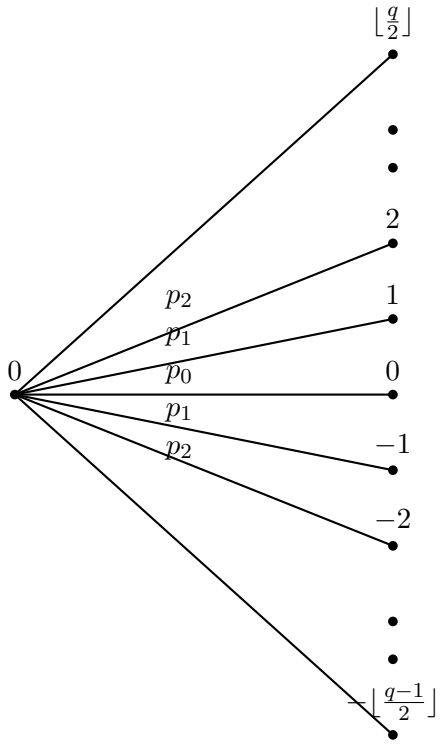

Figura 2.2: Probabiltà per un canale *q-ario* discreto, senza memoria e simmetrico.

Dal Teorema 2 si ha che $p_i = \frac{p_1^i}{p_0^{i-1}}$ per $i > 1$. Si riscrive come

$$p_i = p_0(\frac{p_1}{p_0})^i$$

che è una funzione decrescente in i essendo $p_0 > p_1$. Inoltre, essendo $p_0 > p_1$

$$p_0 > p_1 > p_2 \cdots > p_M.$$

Esempio 3 *Sia Z_4 un anello di interi, vogliamo definire un canale strettamente abbinano alla metrica di Lee su tale anello. Utilizzando il Teorema 2 sappiamo che*

$$p_2 = \frac{p_1^2}{p_0}$$

In linea di principio p_0 e p_1 possono essere scelti arbitrariamente, con l'unico vincolo che $p_0 > p_1$. Dobbiamo comunque rispettare il vincolo di normalizzazione sulle probabilità, ovvero

$$p_0 + 2p_1 + p_2 = 1$$

dove p_1 è stato considerato due volte perché è la probabilità associata sia ad 1 *che a*

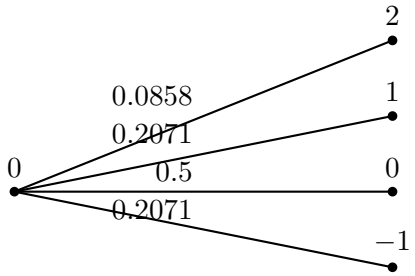

Figura 2.3: Probabiltà per un canale *4-ario* discreto, senza memoria e simmetrico adattato alla metrica di Lee.

-1. *Si ottiene quindi la seguente equazione*

$$p_0 + 2p_1 + \frac{p_1^2}{p_0} = 1.$$

Scegliamo un valore arbitrario di p_0*, ad esempio* $p_0 = 1/2$ *ed otteniamo la seguente equazione*

$$p_1^2 + p_1 - \frac{1}{4} = 0$$

che è un'equazione di secondo grado la cui soluzione positiva è $p_1 \simeq 0.2071$*, dunque il canale è definito nel seguente modo*

$$p_0 = 0.5$$

$$p_1 = 0.2071$$

$$p_2 = \frac{p_1^2}{p_0} = 0.0858$$

Dal Corollario 1 evince il fatto che in un canale adattato alla metrica di Lee i simboli del vettore errore e sono polarizzati verso gli elementi di Z_q con peso di Lee più basso. Del resto assegnare un contenuto informativo maggiore (probabilità minore) a simboli che danno un contributo maggiore in termini di peso di Lee è un risultato che ci si poteva aspettare a priori. La condizione sulle probabilità del Teorema 2 è complicata da raggiungere, mentre la disuguaglianza del Corollario 1 è sicuramente più pratica e sarà quella usata nei capitoli successivi. Sarà comunque scorretto usarla come condizione sufficiente per dire che un canale sia adattato alla metrica di Lee, dunque verrà usata l'espressione *debolmente adattato*.

2.3 Codici Low Lee Density Parity Check (LLDPC)

Nel Capitolo 1 si è parlato dei codici LDPC che, classicamente, vengono definiti su un alfabeto binario nella metrica di Hamming. Si tratta di matrici sparse (pochi 0 e

tanti 1) che hanno il vantaggio di semplificare l'onere computazionale degli algoritmi di decodifica iterativi.
Sia Z_q un anello di interi, si definiscono i codici LDPC su tale anello (che chiameremo LLDPC) come i codici che possiedono una matrice di parità H sparsa nella metrica di Lee, ovvero il peso di Lee delle righe e delle colonne di H è piccolo rispetto alla lunghezza del codice n. Il codice viene detto regolare se si fissano i pesi di Lee delle colonne e delle righe di H, rispettivamente w_c e w_r. In maniera analoga a come sono stati definiti i codici QC-LDPC nella metrica di Hamming si definiscono i codici QC-LLDPC. Per ulteriori dettagli si fa riferimento a [1]. Nella presente trattazione i QC-LLDPC vengono costruiti affiancando due matrici circolanti che hanno le colonne dello stesso peso di Lee

$$H = [H_0 H_1].$$

Il peso risultante delle righe è dunque il doppio rispetto a quello delle colonne

$$w_r = 2w_c.$$

Costruire una matrice in questo modo consente di poterla rappresentare con soli 2 vettori con conseguente vantaggio in termini di memoria occupata e parallelizzazione dei calcoli.

2.4 Symbol Flipping

Per consentire una decodifica efficiente dei codici LLDPC si introduce un algoritmo di decodifica chiamato *Symbol Flipping (SF)* (Algoritmo 1) che si ispira al classico Bit Flipping.
Si tratta di un algoritmo iterativo che, iterazione dopo iterazione, aggiorna il valore della sindrome e del vettore errore dati in ingresso. Vengono inoltre fissati i parametri i_{max} e β (per ora costante), che rappresentano, rispettivamente il numero massimo di iterazioni e la soglia di decodifica. In uscita si ha la stima del vettore errore ed il valore della sindrome (auspicabilmente un vettore di zeri). La procedura di decodifica è descritta nell'Algoritmo 1.
Si inizializza la stima del vettore errore e ed il contatore i a zero. Il ciclo while prosegue fino a quando i non raggiunge i_{max} oppure la sindrome s non diventa un vettore nullo. Ad ogni iterazione la sindrome viene copiata in una variabile s' (s' rimarrà fissa per tutta la durata dell'iterazione mentre verrà aggiornato s). Viene inizializzato il contatore massimo ed il contatore del valore massimo per consentire nuovi aggiornamenti. Il ciclo for dello step 8 dell'algoritmo esplora tutte le colonne di H (j indica l'indice della colonna di H, ma anche dell'indice del vettore errore), mentre il ciclo for dello step 9 fa lo stesso per gli elementi dell'anello Z_q escluso l'elemento nullo. Il contatore massimo (σ_M) viene aggiornato se

$$\sigma = w - w_L(s' - ah_j \mod q) > \sigma_M.$$

Si può facilmente intuire che andando avanti con l'indice j, σ_M diventa sempre più grande e di conseguenza sarà meno probabile che vi sia un aggiornamento (nel capitolo successivo discuteremo più approfonditamente di questo fenomeno). Se quest'ultimo aggiornamento avviene, sarà possibile che vi sia l'aggiornamento di s ed e a condizione che

$$\sigma_M > \beta.$$

Il valore di β è molto importante in quanto da questa variabile dipendono le prestazioni del codice. Di solito si può trovare la soglia ottima di decodifica (ovvero quella che consente le prestazioni migliori) per un dato codice facendo un certo numero di prove.

Algorithm 1 Symbol Flipping

Input: $H \in Z_q^{r \times n}$, $s \in Z_q^r$, $i_{max} \in N$, $\beta \in N$
Output: $e \in Z_q^n$, $s \in Z_q^r$

1: $e \leftarrow 0_n$ `Stima del vettore errore`
2: $i \leftarrow 0$ `Numero di iterazioni`
3: **while** $(w_L(s) > 0)$ **and** $(i < i_{max})$ **do**
4: $s' \leftarrow s$ `Copia della sindrome`
5: $\sigma_M \leftarrow -1$ `Contatore massimo`
6: $a_M \leftarrow 0$ `Contatore del valore massimo`
7: $w \leftarrow w_L(s')$
8: **for** $j \leftarrow 0$ **to** $n-1$ **do**
9: **for** $a \in Z_q \backslash \{0\}$ **do**
10: $\sigma \leftarrow w - w_L(s' - ah_j \mod q)$
11: **if** $\sigma > \sigma_M$ **then**
12: $\sigma_M \leftarrow \sigma$
13: $a_M \leftarrow a$
14: **if** $\sigma_M > \beta$ **then**
15: $e_j \leftarrow e_j + a_M \mod q$ `Aggiornamento del vettore errore`
16: $s \leftarrow s - a_M h_j \mod q$ `Aggiornamento della sindrome`
17: **end if**
18: **end if**
19: **end for**
20: **end for**
21: $i \leftarrow i + 1$ `Aggiornamento del numero di iterazioni`
22: **end while**

Capitolo 3

Ottimizzazione

3.1 Introduzione

In questo capitolo si prenderanno in esame i codici QC-LLDPC e l'algoritmo di decodifica Symbol Flipping già definiti nel capitolo 2.
L'obbiettivo è quello di ottimizzare le prestazioni del codice in termini di capacità correttiva. Per simulare il canale verrà estratto il vettore errore e in modo random (fissato l'alfabeto Z_q, la dimensione n ed il peso di Lee t).
Dopo una serie di simulazioni è risultato che un vettore scelto in questo modo avrà le sue componenti non nulle fortemente polarizzate verso gli elementi (vedi Figura 3.1, Figura 3.2) di Z_q con peso di Lee piccolo. Dunque, si suppone in modo di avere a che fare con un canale *debolmente adattato alla metrica di Lee*.

3.2 Considerazioni sulla sindrome

Si è visto nel capitolo 2 quanto la sindrome sia importante nell'algoritmo di decodifica Symbol Flipping, in particolare il suo peso di Lee. In questo paragrafo proveremo a capire se le matrici di parità H doppio circolanti che restituiscono mediamente un valore $w_L(eH^T)$ più piccolo siano migliori o peggiori per le prestazioni del Symbol Flipping.
Per fare ciò mettiamo appunto un esperimento nel seguente modo

1. Si generano 1000 matrici doppio circolanti $H_i = [H_0 H_1]$, con $i = 1, 2, \ldots, 1000$.

2. Per ogni matrice si considerano vettori con peso di Lee differente (t=1,2,..., 100) e_t.

3. Per ogni valore di t si generano 10 vettori random.

4. Per ogni matrice H_i ed ogni valore di t vengono calcolati 10 valori $w_L(e_t H_i^T)$ e si calcola la media su questi 10 valori $\bar{w_L}(e_t H_i^T)$.

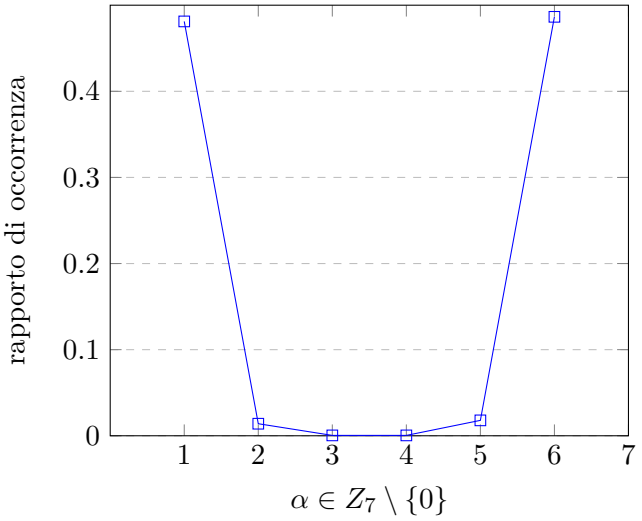

Figura 3.1: Rapporti delle occorrenze degli elementi di un vettore $e \in Z_7^{100}$ con peso di Lee $t = 7$.

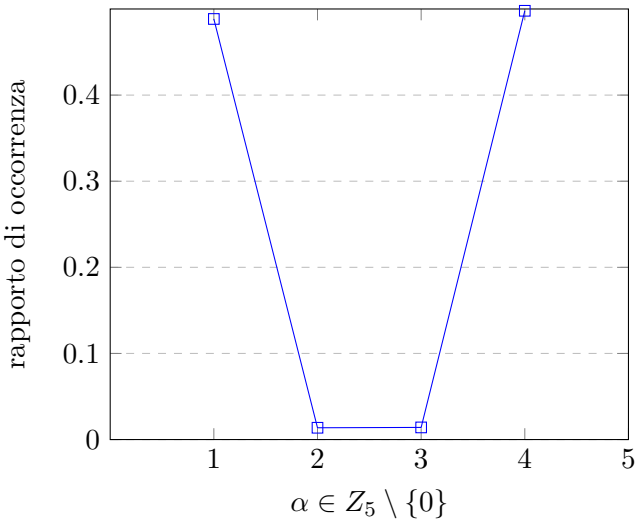

Figura 3.2: Rapporti delle occorrenze degli elementi di un vettore $e \in Z_5^{100}$ con peso di Lee $t = 7$.

5. I valori così ottenuti vengono inseriti in una matrice di dimensione 100×1000

$$M = \begin{pmatrix} \bar{w_L}(e_1 H_1^T) & \dots & \bar{w_L}(e_1 H_i^T) & \dots & \bar{w_L}(e_1 H_{1000}^T) \\ \vdots & & \vdots & & \vdots \\ \bar{w_L}(e_t H_1^T) & \dots & \bar{w_L}(e_t H_i^T) & \dots & \bar{w_L}(e_t H_{1000}^T) \\ \vdots & & \vdots & & \vdots \\ \bar{w_L}(e_{100} H_1^T) & \dots & \bar{w_L}(e_{100} H_i^T) & \dots & \bar{w_L}(e_{100} H_{1000}^T) \end{pmatrix}$$

6. Su ogni riga della matrice si sceglie il valore minimo $\min \bar{w_L}(e_1 H_i^T)$ e si inserisce l'indice corrispondente a tale valore (i) in un vettore m, che avrà dunque 100 elementi

$$m = \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_{100} \end{pmatrix}$$

7. Infine si sceglie il valore più ricorrente di m che corrisponde all'indice della matrice che si voleva trovare.

Le matrici ottenute da questo esperimento hanno prestazioni notevolmente peggiori rispetto alle matrici generate in modo random. Ciò è dovuto al fatto che in queste matrici è molto probabile ottenere *cancellazioni*. Il fenomeno delle cancellazioni, per comodità verrà spiegato con un esempio.

Esempio 4 *Si consideri un vettore $e \in Z_5^4$ ed una matrice $H \in Z_5^{4\times 3}$*

$$e = \begin{pmatrix} 2 & 0 & 1 & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} 1 & 1 & 1 & 2 \\ 0 & 4 & 1 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix}$$

Si ha che

$$eH^T = 2\begin{pmatrix} 1 & 0 & 2 \end{pmatrix} + 1\begin{pmatrix} 1 & 1 & 0 \end{pmatrix} + 1\begin{pmatrix} 2 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2+1+2 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 4 \end{pmatrix} \mod 5$$

Il primo elemento del vettore eH^T ha subito una cancellazione in quanto essendo nullo, ma allo stesso tempo risultato di una sommatoria di diversi elementi non nulli.

Nell'Esempio 4 è avvenuto il fenomeno della cancellazione su una sola componente della sindrome. Se avvenisse su più componenti parleremo di *cancellazioni multiple*. Queste ultime, si suppone per ora, sono una delle cause dell'abbassamento delle prestazioni di un codice. Nel paragrafo successivo proveremo a dare una forma alle matrici di parità H in modo da ridurre la probabilità che avvengano.

Per completezza definiamo un altro tipo di cancellazione "più debole" con un altro esempio.

Esempio 5 *Si consideri un vettore* $e \in Z_6^4$ *ed una matrice* $H \in Z_6^{4\times 3}$

$$e = \begin{pmatrix} 2 & 0 & 4 & 0 \end{pmatrix}$$

$$H = \begin{pmatrix} 2 & 0 & 1 & 3 \\ 3 & 5 & 1 & 1 \\ 1 & 1 & 3 & 1 \end{pmatrix}$$

$$eH^T = 2\begin{pmatrix} 2 & 3 & 1 \end{pmatrix} + 4\begin{pmatrix} 1 & 1 & 3 \end{pmatrix} = \begin{pmatrix} 4 & 0 & 2 \end{pmatrix} + \begin{pmatrix} 4 & 4 & 0 \end{pmatrix} \mod 6$$

I 2 vettori nel termine più a destra dell'uguaglianza hanno subito ciascuno una cancellazione, rispettivamente nel secondo e terzo elemento, essendo questi nulli e derivati da prodotto di elementi non nulli:

$$3 \times 2 = 0 \mod 6$$

$$4 \times 3 = 0 \mod 6.$$

La cancellazione dell'Esempio 5 la chiameremo per l'appunto *cancellazione debole*, in quanto affligge le singole componenti della sommatoria . Questo tipo di cancellazione si può facilmente evitare lavorando su anelli Z_q con q primo, che sono anche campi; in questo modo il risultato del prodotto è nullo solo se uno dei due coefficienti moltiplicativi è nullo.

3.3 Struttura della matrice di parità

Nel precedente paragrafo sono state trovate matrici che hanno prestazioni molto peggiore rispetto alle matrici generate in modo random a causa delle cancellazioni e a partire da queste matrici si vuole dedurre la struttura che una matrice di parità H *non* deve avere. Si focalizzerà l'attenzione sulla posizione degli elementi non nulli, trascurandone il valore. Le matrici ottenute nel paragrafo precedente sono state osservate in modo attento ma poco formale e le deduzioni fatte in questa sezione si possono considerare a dir poco euristiche, ma che comunque saranno sostenute da risultati sperimentali. Si introduce il concetto di *ciclo* come un fenomeno che aumenta la probabilità che avvengano *cancellazioni multiple*.

3.3.1 Cicli

Definizione 9 *Si prendano 4 elementi non nulli* $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in Z_q$ *di una matrice sparsa* H*. I 4 elementi di* Z_q *formano un ciclo se* α_1 *e* α_2 *appartengono alla stessa riga della matrice,* α_3 *ed* α_4 *appartengono anch'essi alla stessa riga diversa da quella*

precedente, α_1 ed α_3 appartengono alla stessa colonna della matrice ed α_2 ed α_4 appartengono alla stessa colonna diversa da quella precedente:

$$H = \begin{pmatrix} & \vdots & & \vdots & \\ \dots & \alpha_1 & - - - - & \alpha_2 & \dots \\ & | & & | & \\ & | & & | & \\ \dots & \alpha_3 & - - - - & \alpha_4 & \dots \\ & \vdots & & \vdots & \end{pmatrix}.$$

Si suppone che questo tipo di struttura dia probabilità maggiori per cancellazioni multiple. Per capire meglio ciò che si intende si guardi l'Esempio 6.

Esempio 6 *Si consideri una matrice $H \in Z_5^{7\times 8}$*

$$H = \begin{pmatrix} 0 & 4 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Si noti che gli elementi in posizioni $(1,2),(1,6),(5,2)$ e $(5,6)$ formano un ciclo. Si prenda un vettore $e \in Z_5^8$

$$e = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 4 & 0 & 0 \end{pmatrix}$$

Allora

$$eH^T = 1\begin{pmatrix} 4 & 0 & 0 & 0 & 1 & 3 & 0 \end{pmatrix} + 4\begin{pmatrix} 1 & 0 & 0 & 0 & 4 & 0 & 0 \end{pmatrix} =$$

$$= \begin{pmatrix} 4+1 & 0 & 0 & 0 & 1+4 & 3 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 0 \end{pmatrix} \mod 5$$

Abbiamo assistito a 2 cancellazioni e ciò danneggerebbe molto un decodificatore Symbol Flipping.

Nella Definizione 9 è stato definito il ciclo in cui compaiono 4 elementi, ma è possibile definira analogamente un ciclo con 6 elementi

$$H = \begin{pmatrix} & \vdots & & \vdots & \\ \dots & \alpha_1 & - - - - & \alpha_2 & \dots \\ & | & & | & \\ & | & & | & \\ \dots & \alpha_3 & - - - - & \alpha_4 & \dots \\ & | & & | & \\ & | & & | & \\ \dots & \alpha_5 & - - - - & \alpha_6 & \dots \\ & \vdots & & \vdots & \end{pmatrix}.$$

con 8 elementi

$$H = \begin{pmatrix} & \vdots & & \vdots & \\ \dots & \alpha_1 & - - - - & \alpha_2 & \dots \\ & | & & | & \\ & | & & | & \\ \dots & \alpha_3 & - - - - & \alpha_4 & \dots \\ & | & & | & \\ & | & & | & \\ \dots & \alpha_5 & - - - - & \alpha_6 & \dots \\ & | & & | & \\ & | & & | & \\ \dots & \alpha_7 & - - - - & \alpha_8 & \dots \\ & \vdots & & \vdots & \end{pmatrix}.$$

e così via.
Maggiore è il numero di elementi coinvolti, maggiore sarà la probabilità di avere cancellazioni multiple.

3.3.2 Applicazione ai QC-LLDPC

In questo paragrafo si vuole studiare come ridurre la probabilità delle cancellazioni multiple nei codici QC-LLDPC. Assumiamo senza perdere di generalità che H sia una matrice circolante (l'estensione ad array di matrici circolanti sarà immediata). Una matrice circolante $H \in Z_q^{n \times n}$ è rappresentata da un singolo vettore $h \in Z_q^n$ nel seguente modo

$$H = \begin{pmatrix} h & h(\leftarrow 1) & h(\leftarrow 2) & \dots & h(\leftarrow n-1) \end{pmatrix}$$

dove con $(\leftarrow k)$ si indica lo shift verso il basso di k posizioni.
Siano $\alpha_1, \alpha_2, \alpha_3$ tre elementi non nulli di h,

$$h = \begin{pmatrix} \vdots \\ \alpha_1 \\ \uparrow \\ d_1 \\ \downarrow \\ \alpha_2 \\ \uparrow \\ d_2 \\ \downarrow \\ \alpha_3 \\ \vdots \end{pmatrix}$$

dove d_1 è il numero di indici che separano α_1 e α_2 e d_2 è il numero di indici che separano α_2 e α_3. Nella matrice circolante definita in precedenza ci sarà una colonna fatta nel seguente modo

$$h(\leftarrow k) = \begin{pmatrix} \vdots \\ \alpha_2 \\ \uparrow \\ d_2 \\ \downarrow \\ \alpha_3 \\ \vdots \\ \vdots \end{pmatrix}$$

tale che α_1 ed α_2 occupino lo stesso indice. Se $d_1 = d_2$ si ottiene un ciclo con 4 elementi, infatti

$$H = \begin{pmatrix} \vdots & & & \vdots & \\ \alpha_1 & \dots & \dots & \alpha_2 & \dots \\ \uparrow & & & \uparrow & \\ d_1 & & & d_2 = d_1 & \\ \downarrow & & & \downarrow & \\ \alpha_2 & \dots & \dots & \alpha_3 & \\ \vdots & & & \vdots & \end{pmatrix}.$$

Esempio 7 *Si prende un un vettore* $h \in Z_5^7$

$$h = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

in cui la distanza tra il primo elemento non nullo d_1 *ed il secondo è uguale alla distanza tra il terzo ed il secondo* d_2*. A partire da questo vettore si genera una matrice circolante* $H \in Z_q^{7\times 7}$

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 2 \\ 2 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 2 & 0 & 0 & 1 \end{pmatrix}$$

osserviamo che si formano dei cicli di 4 elementi a causa del fatto che $d_1 = d_2$*, ad esempio tra gli elementi in posizione* $(1,1), (1,5), (4,1)$ *e* $(4,5)$*.*
Se si prende un altro vettore

$$h' = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

in cui $d_1 \neq d_2$*, esso genera una matrice circolante*

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 2 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 2 \\ 2 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 2 & 0 & 1 \end{pmatrix}$$

in cui non è presente nessun ciclo.

Vogliamo ora formalizzare meglio questo concetto.
Si chiama *supporto* di h il vettore che contiene i suoi elementi non nulli

$$h_{supp} = \begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_k \end{pmatrix}$$

e sia

$$h_{ind} = \begin{pmatrix} i_1 & i_2 & \dots & i_k \end{pmatrix}$$

il vettore che indica le posizioni degli elementi di h_{supp} nel vettore h, con k che è il numero degli elementi del supporto. Dati due elementi del supporto α_j e α_s, si definisce la loro distanza (il numero di posizioni che li separano) come

$$d_{j,s} = i_j - i_s.$$

Affermazione 1 *Una matrice circolante H non possiede cicli se*

$$d_{j,s} \neq d_{j',s'}$$

in qualsiasi modo si scelgano gli indici $j, j', s, s' < n$.

Dunque, per verificare che una matrice circolante non abbia cicli occorre calcolarsi tutte le distanze $d_{j,s}$ e verificare che nessuna di esse sia uguale. Se ciò non è possibile è auspicabile che ce ciò accada il meno frequentemente possibile. Se il numero di elementi del supporto diventa grande diventa un calcolo piuttosto lungo, ma avendo a che fare con matrici sparsi (dunque anche il vettore h sarà sparso) il supporto avrà sempre una numerosità contenuta.
Nel caso in cui H sia formato da più matrici circolanti(nel nostro caso due) basta calcolare tutte le distanze per ogni matrice per poi confrontarle tutte insieme.

3.3.3 Caso pratico

Si prendono in esame i codici QC-LLDPC formati da due blocchi di matrici circolanti

$$H = [H_0 H_1].$$

Per definire un codice occorrono 2 vettori. Prendiamo in esame 2 codici con gli stessi parametri:

1. Lunghezza del codice $n = 160$
2. Alfabeto Z_5
3. Peso di Lee delle colonne $w_c = 6$
4. Soglia di decodifica $\beta = 2$

Codice 1: Rappresentiamo il vettore h_0 con

$$h_{0supp} = \begin{pmatrix} 4 & 1 & 4 & 4 & 1 & 1 \end{pmatrix},$$

$$h_{0ind} = \begin{pmatrix} 6 & 12 & 25 & 48 & 73 & 77 \end{pmatrix}$$

che possiede un inseme di distanze $d_{j,s}$

$$d_{1,2} = 6,$$

$$d_{1,3} = 19,$$

$$d_{1,4} = 42,$$

$$d_{1,5} = 77,$$

$$d_{1,6} = 71,$$

$$d_{2,3} = 13,$$

$$d_{2,4} = 36,$$

$$d_{2,5} = 51,$$

$$d_{2,6} = 65,$$

$$d_{3,4} = 23,$$

$$d_{3,5} = 48,$$

$$d_{3,6} = 52,$$

$$d_{4,5} = 25,$$

$$d_{4,6} = 29,$$

$$d_{5,6} = 4.$$

Rappresentiamo il vettore h_1 con

$$h_{1supp} = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 \end{pmatrix},$$

$$h_{1ind} = \begin{pmatrix} 4 & 10 & 36 & 44 & 66 \end{pmatrix}$$

che possiede un insieme di distanze $d'_{j,s}$

$$d'_{1,2} = 6,$$

$$d'_{1,3} = 32,$$

$$d'_{1,4} = 40,$$

$$d'_{1,5} = 62,$$
$$d'_{2,3} = 26,$$
$$d'_{2,4} = 24,$$
$$d'_{2,5} = 56,$$
$$d'_{3,4} = 8,$$
$$d'_{3,5} = 30,$$
$$d'_{4,5} = 33.$$

Enumerando tutte le distanze di h_0 ed h_1 si trova una sola coincidenza:

$$d_{1,2} = d'_{1,2} = 6.$$

La condizione imposta dall'Affermazione 1 viene infranta una sola volta.

Codice 2: Rappresentiamo il vettore h_0 con

$$h_{0supp} = \begin{pmatrix} 4 & 4 & 4 & 1 & 1 & 4 \end{pmatrix},$$

$$h_{0ind} = \begin{pmatrix} 24 & 29 & 57 & 71 & 72 & 75 \end{pmatrix}$$

che possiede un inseme di distanze $d_{j,s}$

$$d_{1,2} = 5,$$
$$d_{1,3} = 33,$$
$$d_{1,4} = 37,$$
$$d_{1,5} = 48,$$
$$d_{1,6} = 56,$$
$$d_{2,3} = 28,$$
$$d_{2,4} = 42,$$
$$d_{2,5} = 33,$$
$$d_{2,6} = 46,$$
$$d_{3,4} = 14,$$
$$d_{3,5} = 15,$$
$$d_{3,6} = 18,$$
$$d_{4,5} = 1,$$

$$d_{4,6} = 4,$$

$$d_{5,6} = 3.$$

Rappresentiamo il vettore h_1 con

$$h_{1supp} = \begin{pmatrix} 1 & 1 & 4 & 4 & 1 & 1 \end{pmatrix},$$

$$h_{1ind} = \begin{pmatrix} 7 & 37 & 53 & 54 & 58 & 66 \end{pmatrix}$$

che possiede un insieme di distanze $d'_{j,s}$

$$d'_{1,2} = 30,$$

$$d'_{1,3} = 46,$$

$$d'_{1,4} = 47,$$

$$d'_{1,5} = 51,$$

$$d'_{1,6} = 59,$$

$$d'_{2,3} = 16,$$

$$d'_{2,4} = 17,$$

$$d'_{2,5} = 21,$$

$$d'_{2,6} = 29,$$

$$d'_{3,4} = 1,$$

$$d'_{3,5} = 5,$$

$$d'_{3,6} = 13,$$

$$d'_{4,5} = 4,$$

$$d'_{4,6} = 12,$$

$$d'_{5,6} = 8.$$

Enumerando tutte le distanze di h_0 ed h_1 si ottengono le seguenti coincidenze:

$$d_{4,5} = d'_{3,4} = 1,$$

$$d_{1,2} = d'_{3,5} = 5,$$

$$d_{1,3} = d'_{2,5} = 33,$$

$$d_{2,6} = d'_{1,3} = 46,$$

$$d_{4,6} = d'_{4,5} = 4.$$

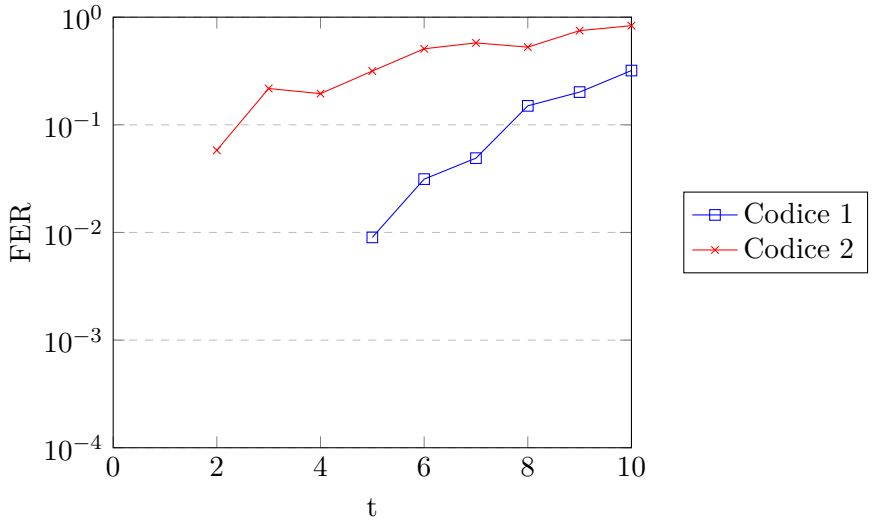

Figura 3.3: Confronto tra il FER del Codice 1 e del Codice 2 con decodifica Symbol Flipping

La condizione imposta dall'affermazione 1 viene infranta ben cinque volte.

Dalle considerazioni fatte fin'ora ci aspettiamo che il Codice 1 abbia delle prestazioni migliori rispetto al Codice 2. I risultati della simulazione dei due codici vengono riportati nella Figura 3.3. Sull'asse delle ordinante troviamo il *Frame Error Rate (FER)* che è il rapporto tra le parole di codice decodificate correttamente ed il numero di parole trasmesse. Ulteriori dettagli sulle simulazioni verranno specificati successivamente. Questa simulazione dà la conferma delle ipotesi fatte, ovvero che la presenza dei cicli aumenta la probabilità di ottenere cancellazioni multiple le quali peggiorano notevolmente le prestazioni del codice.

3.4 Scelta della soglia

Nel precedente capitolo si è detto quanto sia importante la scelta della soglia di decodifica β. Si vuole mostrare la sua importanza facendo delle simulazioni con lo stesso codice, facendo variare la soglia. Per tutte le simulazioni (anche quelle fatte in precedenza) si fisseranno i seguenti parametri

* *max-num-tx*=1000 $\rightarrow$ numero massimo di valori trasmessi per ogni t
* *max-dec-errors*=30 $\rightarrow$ numero massimo di decodifiche fallite per ogni t
* *max-num-iter*=5 $\rightarrow$ numero massimo di decodifiche che effettua il decoder (nell'algoritmo i_{max}).

Per fare la prova viene preso un codice con le seguenti caratteristiche

1. Lunghezza del codice $n = 160$.

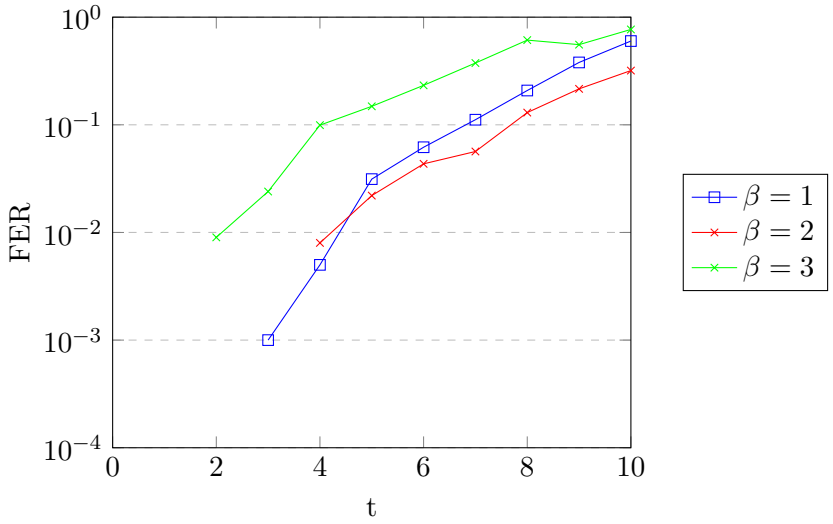

Figura 3.4: Prestazioni del codice in Z_5, $n = 160$ e $w_c = 6$ al variare di β.

2. Alfabeto Z_5,
3. La soglia di decodifica $\beta = 1, 2$ e 3,
4. Peso di Lee delle colonne della matrice di parità $w_c = 6$.

Per una seconda prova si prende un altro codice con le seguenti caratteristiche

1. Lunghezza del codice $n = 200$,
2. Alfabeto Z_7,
3. Soglia di decodifica $\beta = 3, 4, 5$ e 9,
4. Peso di Lee delle colonne $w_c = 13$.

Le prestazioni in termini di *FER* variano notevolmente al variare (vedi Figura 3.4 e Figura 3.5) di β, in particolare $\beta = 2$ risulta essere la soglia ottima per il primo codice preso in esame, mentre il secondo codice ha 3 soglie ottime $\beta = 3, 4$, e 5.

3.4.1 Alcune considerazioni sull'algoritmo di decodifica Symbol Flipping

Per la descrizione dettagliata dell'algoritmo si fa riferimento alla sezione 2.4. In questa sezione si vuole fare una serie di osservazioni con l'obbiettivo di ottimizzare ancora di più la scelta della soglia β.

Affermazione 2 *La variabile σ_{MAX} cresce all'aumentare dell'indice j, dunque la probabilità di aggiornare questa variabile diminuisce all'aumentare dell'indice j.*

L'affermazione 2 deriva dalle considerazioni fatte nella sezione 2.4.

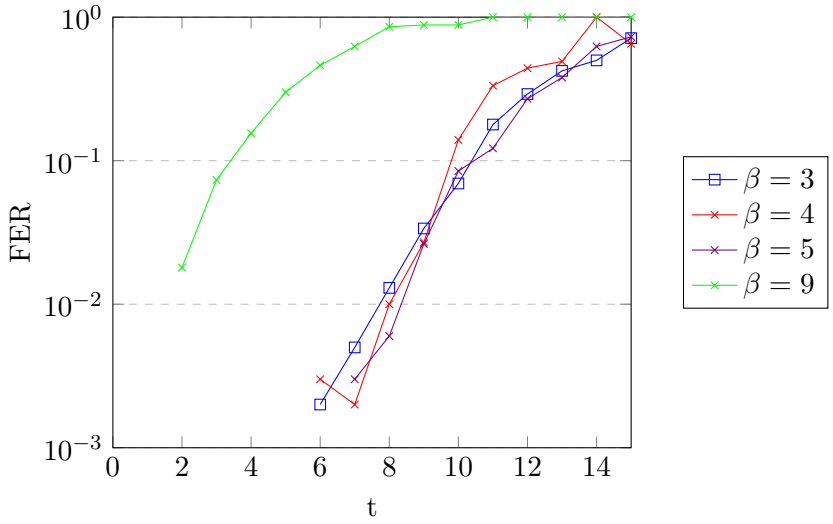

Figura 3.5: Prestazioni del codice in Z_7, $n = 200$ e $w_c = 13$ al variare di β.

Affermazione 3 *L'aggiornamento della sindrome s e del vettore e risulta meno probabile all'aumentare dell'indice j.*

L'Affermazione 3 è conseguenza evidente dell'Affermazione 2. Dall'affermazione 3 consegue che gli elementi del vettore e con indice j più basso vengono aggiornate prima rispetto agli elementi con indice più alto. Dunque è plausibile che gli elementi di e con indice più basso vengono aggiornati nelle prime iterazioni della decodifica, mentre gli elementi con indice più grande nelle iterazioni successive. Più si va avanti con le iterazioni però, minore sarà la probabilità che si verifichi la condizione

$$\sigma_{MAX} > \beta.$$

Infatti i valori di σ, da cui dipende σ_{MAX} diminuiranno a causa del fatto che il peso di Lee della sindrome decresce progressivamente con le iterazioni: lo svantaggio che hanno gli elementi di e con indice più alto può quindi essere colmato abbassando la soglia per tali indici.

3.4.2 Introduzione della soglia variabile

La soglia fissa che vediamo nell'algoritmo di Symbol Flipping 1 della sezione 2.4 viene sostituita da una soglia decrescente con l'indice j

$$\beta_j < \beta_{j-1}.$$

L'algoritmo di Symbol Flipping così aggiornato lo vediamo nell'Algoritmo 1.

Algorithm 2 Symbol Flipping

Input: $H \in Z_q^{r \times n}$, $s \in Z_q^r$, $i_{max} \in N$, $\beta \in N^n$
Output: $e \in Z_q^n$, $s \in Z_q^r$

```
1: e ← 0_n          Stima del vettore errore
2: i ← 0            Numero di iterazioni
3: while (w_L(s) > 0) and (i < i_max) do
4:    s' ← s        Copia della sindrome
5:    σ_M ← −1      Contatore massimo
6:    a_M ← 0       Contatore del valore massimo
7:    w ← w_L(s')
8:    for j ← 0 to n − 1 do
9:       for a ∈ Z_q\{0} do
10:         σ ← w − w_L(s' − a h_j  mod q)
11:         if σ > σ_M then
12:            σ_M ← σ
13:            a_M ← a
14:            if σ_M > β_j then
15:               e_j ← e_j + a_M  mod q     Aggiornamento del vettore errore
16:               s ← s − a_M h_j  mod q     Aggiornamento della sindrome
17:            end if
18:         end if
19:      end for
20:   end for
21:   i ← i + 1        Aggiornamento del numero di iterazioni
22: end while
```

Rimane da verificare se effettivamente questa piccola modifica porta dei vantaggi. Si considera un codice con le seguenti caratteristiche

1. Lunghezza del codice $n = 200$,
2. Alfabeto Z_5,
3. Peso di Lee delle colonne della matrice di parità $w_c = 13$.

I risultati riportati nella Figura 3.6 confrontano le prestazioni delle 3 soglie ottime del codice ($\beta = 3, 4$ e 5) con una soglia variabile in questo modo

$$\beta_j = \begin{cases} 5 & 0 < j < 50 \\ 4 & 50 \leq j < 150 \\ 3 & 150 \leq j \leq 200 \end{cases}$$

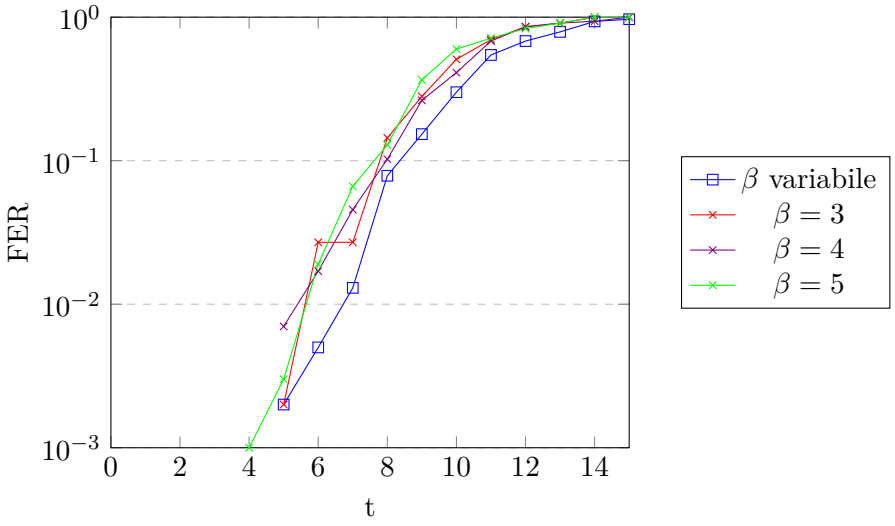

Figura 3.6: Prestazioni del codice in Z_5, $n = 200$ e $w_c = 13$ al variare di β.

Per la seconda prova prendiamo lo stesso codice della Figura 3.5, prendendo una soglia variabile costruita nel seguente modo

$$\beta_j = \begin{cases} 5 & 0 < j < 33 \\ 4 & 33 \leq j < 133 \\ 3 & 133 \leq j \leq 200 \end{cases}$$

in Figura 3.7 i risultati della simulazioni della soglia variabile confrontati con le 3 soglie ottime trovate nel precedente paragrafo.
Per entrambi i codici testati (Figura 3.6 e Figura 3.7) la soglia β_j è stata costruita in modo induttivo, partendo dalle 3 soglie ottime ed andando un po' a tentativi sul come posizionarle (in ogni caso in modo decrescente). Le simulazioni danno ragione di credere che far decrescere la soglia rispetto all'indice j migliori le prestazioni del codice.

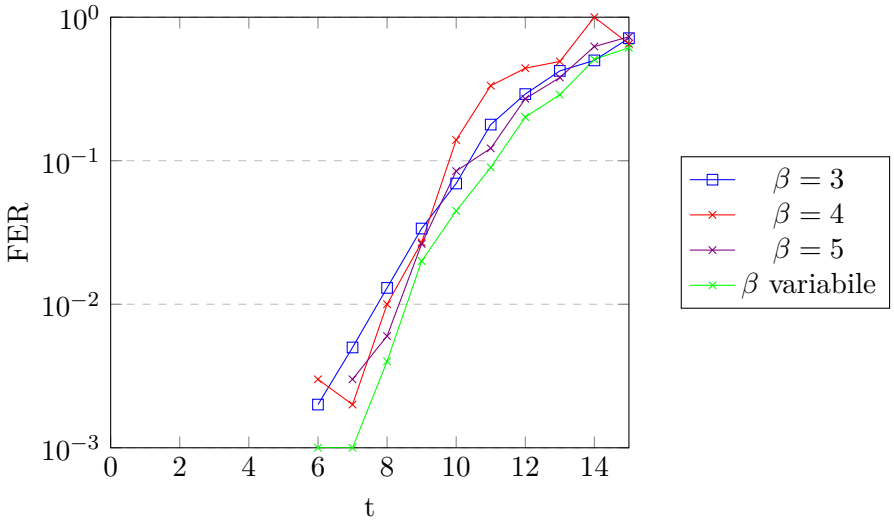

Figura 3.7: Prestazioni del codice in Z_7, $n = 200$ e $w_c = 13$ al variare di β.

Capitolo 4

BSC

4.1 Introduzione

Fino a questo momento le simulazioni sono state fatte generando un vettore e di un certo peso di Lee in modo random ed sommandolo alla parola di codice. È utile testare i codici di cui abbiamo ampiamente discusso (i QC-LLDPC) su un canale noto in letteratura ed a tale scopo si è scelto il canale BSC, lo schema in Figura 4.1. Come è evidente dalla figura il canale funziona solo se la parola di codice trasmessa

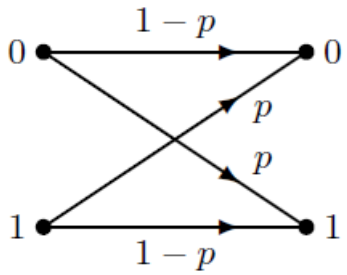

Figura 4.1: Schema del canale BSC.

è una parola di codice binaria, mentre i codici QC-LLDPC che sono stati presi in considerazione fin'ora sono definiti su un alfabeto q-ario. Occorre, quindi, "adattare" i simboli di Z_q sull'alfabeto binario e di questo verrà discusso nella sezione successiva.

4.2 Mapping

Per consentire la trasmissione della parola di codice su un canale BSC si applica una funzione di mapping γ su ogni simbolo q-ario per ottenere in uscita una sequenza di simboli binari di lunghezza fissa. Di seguito elenchiamo due caratteristiche che dovrebbero avere l'insieme del dominio e codominio di γ

1. La numerosità delle sequenze binarie deve essere pari a q (per consentire la biunivocità tra i simboli di Z_q e le sequenze binarie).

2. Le distanze di Lee tra gli elementi dell'anello Z_q e le distanze di Hamming delle sequenze di bit devono avere una corrispondenza biunivoca.

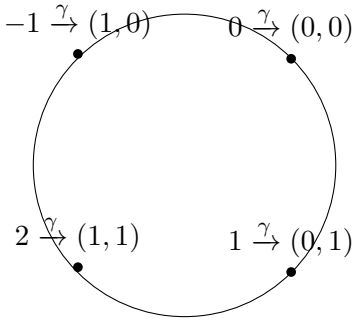

Figura 4.2: Mapping dei simboli in Z_4.

La prima condizione vincola la scelta dell'alfabeto a Z_q con q che deve essere una potenza di 2.
Si considera quindi $\gamma : Z_4 \to Z_2^2$ con una disposizione secondo il mapping di Gray, come illustrato in 4.2

$$0 \in \mathbb{Z}_4 \xrightarrow{\gamma} (0,0) \in \mathbb{Z}_2^2,$$
$$1 \in \mathbb{Z}_4 \xrightarrow{\gamma} (0,1) \in \mathbb{Z}_2^2,$$
$$2 \in \mathbb{Z}_4 \xrightarrow{\gamma} (1,1) \in \mathbb{Z}_2^2,$$
$$-1 \in \mathbb{Z}_4 \xrightarrow{\gamma} (1,0) \in \mathbb{Z}_2^2.$$

Vediamo che in questo caso anche la condizione 2 è verificata: le distanze di Hamming tra le sequenze di due bit corrispondono alle distanze di Lee sulla circonferenza in Figura 4.2. Andiamo quindi a definire lo schema di codifica e decodifica di un codice q-ario su un canale BSC

1. Il messaggio di lunghezza k viene codificato tramite un codice $\mathcal{C} \subseteq Z_4^n$,
2. Gli elementi della parola di codice vengono mappati tramite γ e si ottiene una sequenza binaria di $2n$ simboli,
3. La sequenza binaria transita su un canale BSC,
4. Alla sequenza in uscita dal canale BSC viene applicato il demapping (funzione inversa al mapping) γ^{-1},
5. La sequenza ottenuta nel passaggio precedente viene decodificata tramite il codice $\mathcal{C}$.

In particolare al decodificatore arriva una sequenza il cui generico elemento può essere riscritto come

$$y_i = \gamma^{-1}(\gamma(c_i) + e_i),$$

dove c_i è un generico elemento della parola di codice di $c \in \mathcal{C}$ ed e_i è una coppia di bit, ciascuno dei quali introdotto dal canale BSC.

Il canale BSC è caratterizzato da una certa probabilità di errore sul bit ρ e di conseguenza la probabilità che il bit rimanga integro è pari a $1-\rho$. A partire dalla probabilità di errore sul bit si possono calcolare le probabilità di errore sulle sequenze di due bit e_i e quindi anche sul simbolo Z_q che chiamiamo e. Quello che in realtà ci interessa è di trovare le probabilità che il peso di Lee di e assuma un determinato valore.

$$Pr(e=0) = Pr(e_i=(0,0)) = (1-\rho)^2,$$

$$Pr(e=1) = Pr(e_i=(0,1)) = (1-\rho)\rho,$$

$$Pr(e=2) = Pr(e_i=(1,1)) = \rho^2,$$

$$Pr(e=-1) = Pr(e_i=(1,0)) = \rho(1-\rho).$$

Di conseguenza

$$p_0 = Pr(w_L(e)=0) = Pr(e=0) = (1-\rho)^2,$$

$$p_1 = Pr(w_L(e)=1) = Pr(e=1) = Pr(e=-1) = \rho(1-\rho),$$

$$p_2 = Pr(w_L(e)=2) = Pr(e=2) = \rho^2.$$

Dunque possiamo accorpare la sequenza

$$\gamma \to BSC \to \gamma^{-1}$$

e considerarlo come un canale 4-ario con le probabilità appena elencate.
Si potrebbe pensare di mappare anche l'alfabeto Z_8 sull'insieme delle sequenze di 3 bit. Ciò è possibile in linea di principio, ma andrebbe in contrasto con la seconda delle condizioni elencate a inizio paragrafo. Infatti l'insieme delle distanze di Hamming tra le sequenze di 3 bit assume 4 valori: $0,1,2,3$, mentre l'insieme delle distanze di Lee di un anello Z_8 ne assume 5: $0,1,2,3,4$; quindi non ci può essere una corrispondenza biunivoca tra i due insiemi di distanze.

4.3 Canale adattato (matched channel)

Nel Capitolo 2 si è parlato di *canale adattato* o *canale debolmente adattato* e delle caratteristiche che devono avere. Si vogliono imporre queste caratteristiche al canale descritto nella sezione precedente.

Debolmente adattato:

$$p_0 > p_1 > p_2$$

$$\begin{cases} p_0 > p_1 \\ p_1 > p_2 \end{cases}$$

$$\begin{cases} (1-\rho)^2 > \rho(1-\rho) \\ \rho(1-\rho) > \rho^2 \end{cases}$$

$$\begin{cases} \rho < \frac{1}{2} \\ \rho < \frac{1}{2} \end{cases}$$

$$\rho < \frac{1}{2}.$$

Adattato

$$\begin{cases} p_0 > p_1 \\ p_2 = \frac{p_1^2}{p_0} \end{cases}$$

$$\begin{cases} \rho < \frac{1}{2} \\ \rho^2 = \frac{(\rho(1-\rho))^2}{(1-\rho)^2} \end{cases}$$

$$\begin{cases} \rho < \frac{1}{2} \\ 1 = 1 \end{cases}$$

La seconda equazione del sistema è sempre vera, quindi il sistema è verificato per $\rho < \frac{1}{2}$.

Osserviamo, dunque, che il canale costruito in questo modo è adattato alla metrica di Lee per $\rho < \frac{1}{2}$, il ché, si può considerare sempre vero.

4.4 Simulazione

Si vuole ora simulare il canale ottenuto usando il codici QC-LLDPC. Nel capitolo 2 è stato detto che i codici definiti su un alfabeto q-ario con q che non è un numero primo sono soggette alle *cancellazioni deboli* che degradano le prestazioni dell'algoritmo Symbol Flipping. Questo fenomeno, in realtà, può essere evitato nel caso di $q = 4$ definendo una matrice di parità H in cui non compare l'elemento 2. Infatti l'unica cancellazione possibile è

$$2 \times 2 = 0 \mod 4.$$

Detto ciò si definiscono le caratteristiche del codice che si è preso in esame

1. Lunghezza del codice $n = 200$,
2. Alfabeto Z_4,
3. Peso di Lee delle colonne della matrice di parità $w_c = 5$,
4. Soglia di decodifica variabile

$$\beta_j = \begin{cases} 2 & 0 < j < 166 \\ 1 & 166 \leq j \leq 200 \end{cases}$$

Soglia di decodifica fissa $\beta = 2$.

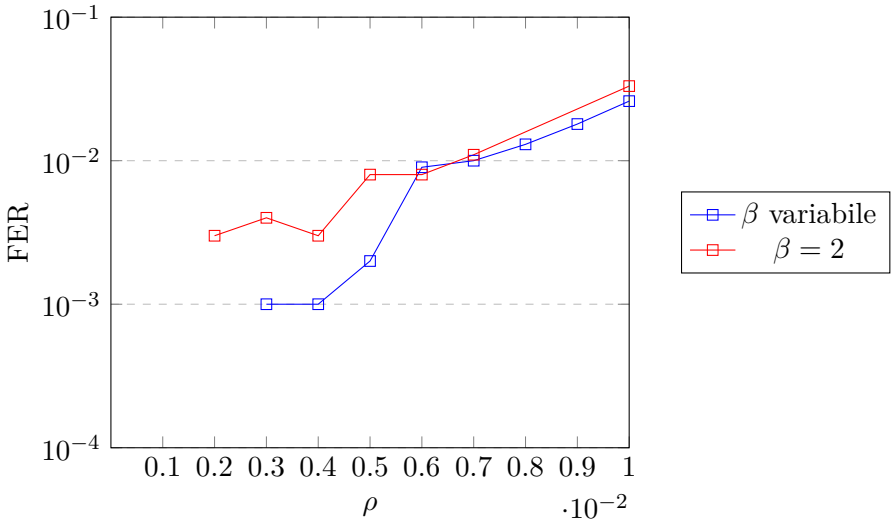

Figura 4.3: Prestazioni su canale BSC.

In Figura 4.3 i risultati della simulazione del codice sul canale BSC. Si è scelto un intervallo di ρ in cui le prestazioni del codice sono osservabili. Anche in questo caso far variare β risulta la scelta ottimale.
Si può stimare il peso di Lee del vettore e in ingresso al decodificatore a partire dalle probabilità $p_1, p_2, \ldots, p_M$ dove $M = \lfloor \frac{q}{2} \rfloor$. Nel caso preso in esame, ovvero Z_4, il simbolo generico α del vettore e assume mediamente il peso di Lee di

$$\tilde{w_L}(\alpha) = 2\rho^2 + 2\rho(1-\rho) = 2\rho.$$

Quindi il peso di Lee del vettore e lungo n simboli

$$\tilde{w_L}(e) = 2\rho n$$

nel nostro caso essendo $n = 200$

$$\tilde{w_L}(e) = 2\rho 200.$$

Siccome nella Figura 4.3 ρ varia tra 10^{-2} e 10^{-3}, il peso di Lee del vettore errore varia all'incirca tra 4 e 0.

Conclusioni

Nel capitolo 3 sono state individuate le caratteristiche che una matrice di parità H non dovrebbe avere per abbassare la probabilità di avere cancellazioni multiple che, come si è visto dalle simulazioni, peggiorerebbero notevolmente le prestazioni dell'algoritmo Symbol Flipping. Si è inoltre introdotta la soglia di decodifica che varia rispetto all'indice j dell'Algoritmo 1 che migliora in modo non trascurabile le prestazioni di quest'ultimo rispetto all'utilizzo della soglia fissa.
Nel capitolo 4 sono state eseguite le simulazioni su un canale BSC tramite un mapping dei simboli dell'anello Z_4 su sequenze di due bit. Si è inoltre dimostrato che un canale costruito in questo modo è abbinato alla metrica di Lee.
Per quanto riguarda gli scenari applicativi si è discusso solo della codifica di canale, però si intravede una possibile applicazione dei codici LLDPC alla crittografia basata sui codici, integrandola ad esempio nel cifrario asimmetrico di McEliece [7] in ottica della crittografia post quantum. Questo argomento sta ricevendo particolare interesse a causa del veloce sviluppo dei computer quantistici i quali integrano l'algoritmo di Shor [8]: questo algoritmo è in grado di fattorizzare grandi numeri e risolvere il problema del logaritmo discreto in tempi ragionevoli. La difficoltà della risoluzione di questi problemi sta alla base della sicurezza dei cifrari asimmetrici dell'attuale standard, quali RSA e ElGamal, i quali dovranno necessariamente essere sostituiti. I crittosistemi basati sui codici vengono considerati ad oggi una valida alternativa.

Bibliografia

[1] Paolo Santini, Massimo Battaglioni, Franco Chiaraluce, Marco Baldi, and Edoardo Persichetti. Low-lee-density parity-check codes. In *ICC 2020 IEEE International Conference on Communications (ICC): Doublin, Ireland*, pages 1–6. IEEE, 2020.

[2] Ron M Roth. *Introduction to coding theory*. Cambridge University Press, 2006.

[3] Wade Trappe. *Introduction to cryptography with coding theory*. Pearson Education India, 2006.

[4] Chang Y Lee. Some properties of nonbinary error-correcting codes. *IRE Transactions on Information Theory*, 4(2):77–82, 1958.

[5] J Chung-Yaw Chiang and Jack K Wolf. On channels and codes for the lee metric. *Information and Control*, 19(2):159–173, 1971.

[6] Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.

[7] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

[8] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science: Santa Fe, New Mexico, USA*, pages 124–134. IEEE, 1994.