Università degli Studi di Milano

Dipartimento di Fisica

IMPLEMENTAZIONE DELL'ALGORITMO DI BELIEF PROPAGATION PER LA DECODIFICA DEI CODICI CORRETTORI

Relatore:
Prof. Sergio CARACCIOLO

Correlatore:
Dott. Pietro ROTONDO

Candidato:
Alessandro TAMMARO
Matr. n. 871899

A.A. 2017-2018

Sommario

La meccanica statistica dei sistemi disordinati è stata sviluppata ben oltre il suo obiettivo iniziale di spiegare i fatti sperimentali di quei particolari magneti in cui legami ferromagnetici e antiferromagnetici sono distribuiti in modo casuale. In particolare, a partire dagli anni Ottanta ci si è accorti che una serie di problemi interdisciplinari, che includono le reti neurali, l'ottimizzazione combinatoria e l'inferenza bayesiana, possono essere studiati con i medesimi strumenti utilizzati nell'ambito dei vetri di spin.

La teoria dei codici correttori (e più in generale la teoria dell'informazione) formulata inizialmente da C. Shannon negli anni Quaranta, fornisce uno degli esempi più significativi di quanto sopra affermato. Il primo a comprendere la connessione tra gli ECC e i sistemi disordinati fu N. Sourlas nel 1989, che si accorse dell'equivalenza tra il problema di decoding negli ECC e la ricerca del ground state di una particolare classe di modelli di Ising con interazioni disordinate. Le hamiltoniane fully-connected introdotte da Sourlas furono successivamente modificate per includere il caso di connettività finita (più rilevante dal punto di vista pratico) da Y. Kabashima e D. Saad a fine anni Novanta.

Ad oggi, lo stato dell'arte della teoria ECC è costituito dai *turbo codici* e dai *Low-Density Parity-Check* codes (LDPC) o *Gallager codes*, i quali impiegano una tecnica di decoding conosciuta come *belief propagation*. In un articolo del 2003, Javier Garcia-Frias e Wei Zhong si sono chiesti se fosse possibile riuscire a replicare le prestazioni dei codici LDPC, applicando la stessa tecnica di decoding ad una particolare concatenazione di codici di Sourlas a connettività finita, i quali vengono identificati all'interno della comunità di computer science come *Low-Density Generator-Matrix* codes (LDGM). Il vantaggio di questi codici, che prendono il nome di *Serially Concatenated Low-Density Generator-Matrix* codes (SCLDGM), rispetto ai turbo codici e ai codici LDPC, risiede in un notevole abbassamento della complessità delle procedure di encoding e decoding a parità di prestazioni.

In questo elaborato ci proponiamo, attraverso l'implementazione dell'algoritmo di belief propagation, di studiare alcune proprietà della classe di codici LDGM e di studiarne in maniera sistematica le prestazioni al variare dei principali parametri in gioco. Forniremo inoltre un esempio delle effettive prestazioni che si possono raggiungere grazie alla tecnica della concatenazione.

Nel dettaglio, l'elaborato si compone di tre parti, la prima delle quali dedicata ad una introduzione alla teoria ECC ed un'analisi della meccanica statistica dei codici di Sourlas sulla base del modello di vetri di spin. Il Capitolo 1 si concentra sull'introduzione alla teoria ECC e al Teorema di Shannon. Sono riportate le teorie dei più semplici codici correttori, quali i Repetition Codes e l'Hamming Code (7,4) e dei codici della classe dei *linear block codes* a cui appartengono i LDGM e i Gallager codes (LDPC, *Low-Density Parity-Check*). Inoltre viene fornita una descrizione di come si possa costruire un legame tra la teoria ECC e i modelli di vetri di spin. Il Capitolo 2 definisce gli aspetti teorici dei codici di Sourlas e presenta una definizione formale del parametro di overlap.

La seconda parte è dedicata all'implementazione numerica dell'algoritmo di belief propagation. Nello specifico, dapprima si definisce una funzione hamiltoniana di vetri di spin e si mostra come può essere rappresentata tramite factor graph.

Di seguito viene fornita una descrizione qualitativa dei processi dell'algoritmo *sum-product message passing* (SPA), alla base di BP, per factor graph ad albero. Infine viene presentata la teoria dei codici SCLDGM, *Serially Concatenated Low-Density Generator-Matrix*.

La terza parte è dedicata ad uno studio numerico approfondito delle prestazioni dei codici introdotti nei capitoli precedenti. Nel Capitolo 4 vengono discusse le condizioni al contorno delle simulazioni dei codici LDGM. Nel capitolo 5 sono presentate per prime le prestazioni dei codici LDGM in funzione dei principali parametri in gioco: il grado dell'interazione K, il rate R e l'errore intrinseco del canale p. Infine vengono presentate le prestazioni dei codici SCLDGM. Il primo risultato che mostriamo è un confronto dei dati numerici ottenuti da codici LDGM a $R = 1/2$, $R = 1/3$ e $R = 1/4$, con la stima proveniente dalla previsione teorica. Otteniamo che i grafici riproducono in maniera soddisfacente i risultati presenti in letteratura. Nella seconda fase, attraverso le simulazioni su un codice SCLGDM, si dimostra come la concatenazione aumenta notevolmente le prestazioni di un singolo LDGM, a prezzo di un leggero abbassamento del rate R, abbattendo il problema dell'error floor presente in questi codici [12].

Indice

Parte I

ERROR CORRECTING CODES E FORMULAZIONE MECCANICO-STATISTICA DEL PROBLEMA

Capitolo 1

Error correcting codes

L'affidabilità nella trasmissione di informazione tramite canali rumorosi gioca un ruolo chiave nella società moderna. La teoria alla base di questo studio prende il nome di *error-correcting code* (ECC). Alcuni dei suoi aspetti possono essere compresi e formalizzati proprio attraverso il modello di vetri di spin: il rumore nel canale di trasmissione può essere posto in analogia con le interazioni casuali nei vetri di spin e la sequenza di bit con la corrispondente configurazione del modello di Ising. Gli studi sulla teoria dell'informazione furono iniziati da Claude Shannon (1916-2001), ingegnere e matematico statunitense, il quale ne formulò delle prime nozioni di base sviluppando una struttura per la manipolazione di questi oggetti astratti [22].

1.1 Error correcting codes

1.1.1 Trasmissione dell'informazione

Supponiamo di voler trasmettere un messaggio di N bit da un luogo a un altro, per fare ciò è necessario utilizzare un *canale di trasmissione*. In generale un canale è soggetto a rumore, pertanto il messaggio di uscita può risultare corrotto. L'obiettivo che ci si propone consiste proprio nel capire come dedurre il messaggio originale dall'output rumoroso.
Il processo di trasmissione si compone di diverse fasi:

- la fase di *encoding* (o canale di codificazione), in cui l'informazione viene tradotta in simboli per la comunicazione o la conservazione;
- la fase di *trasferimento*, in cui il messaggio codificato è poi trasmesso attraverso un canale rumoroso;
- la fase *decoding*, ossia il processo di recupero dell'informazione originale dal canale rumoroso.

Essendo il linguaggio macchina di tipo binario, composto quindi di sequenze di 0 e 1, l'azione del rumore sull'informazione si traduce nella mutazione di uno 0 in 1 o viceversa. Appare dunque evidente come nel processo di encoding sia necessario rendere ridondante il messaggio originale aggiungendo ulteriore informazione.

Un semplice esempio di questo processo è costituito dalla ripetizione multipla (solitamente 3) della sequenza di bit da inviare. Se le sequenze ricevute sono identiche, si può assumere che il canale sia privo di rumore. Diversamente, se uno specifico bit risulta diverso in una delle sequenze, si ha una buona probabilità di dedurre il valore corretto (0 o 1) per maggioranza.

Un metodo più sofisticato è il *parity-check code* basato sui parity-bit. Un parity-bit è un bit aggiunto alla stringa di un codice binario affinché controlli che il numero di 1 all'interno di una specifica sequenza sia pari o dispari. Nel caso in cui tale numero sia pari, viene aggiunto un parity-bit uguale a 0, in caso contrario uguale a 1. Se il rumore del canale è assunto non essere eccessivamente grande, con buona approssimazione si può ritenere che il messaggio di output sia privo di errori se la parità dei parity-bit coincide con la parità della stringa ricevuta. Se ciò non dovesse accadere, si riscontra un errore (*error detection*). La fase di correzione degli errori (*error correction*) sarà oggetto dei prossimi paragrafi, e verrà trattata nel caso particolare di *binary simmetric channel* (BSC)[1].

1.1.2 Repetition code

Il primo semplice esempio di algoritmo di encoding e decoding è il *repetition code*. Supponiamo di voler inviare attraverso un BSC una semplice sequenza di 7 bit $\boldsymbol{\xi}$, ad esempio

$$\boldsymbol{\xi} = 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0$$

e che il rumore del canale sia pari a $p = 0.1$. Come già osservato nel precedente paragrafo, per poter ridurre l'errore di trasmissione è necessario rendere ridondante il messaggio di input. Nel caso del repetition code R_M, l'algoritmo di encoding prevede di inviare ogni bit copiato M volte. Se M = 3, il vettore di encoding $\boldsymbol{J}^0$ sarà quindi

$$\boldsymbol{J}^0 = 000 \quad 000 \quad 111 \quad 000 \quad 111 \quad 111 \quad 000$$

Possiamo pensare al rumore del canale come un vettore $\boldsymbol{\Delta}$ che viene aggiunto in modulo 2 al vettore originale, $(1+1)\, mod\, 2 = 0$. In questo caso lo scegliamo uguale a

$$\boldsymbol{\Delta} = 000 \quad 001 \quad 000 \quad 000 \quad 101 \quad 000 \quad 000$$

Detto $\boldsymbol{J}$ il vettore di output, ossia $\boldsymbol{J} = \boldsymbol{J}^0 + \boldsymbol{\Delta}$, allora questo sarà uguale a

$\boldsymbol{\xi}$	0	0	1	0	1	1	0	
$\boldsymbol{J}^0$	$\overbrace{000}$	$\overbrace{000}$	$\overbrace{111}$	$\overbrace{000}$	$\overbrace{111}$	$\overbrace{111}$	$\overbrace{000}$	+
$\boldsymbol{\Delta}$	000	001	000	000	000	101	000	=
$\boldsymbol{J}$	000	001	000	000	111	010	000	

A questo punto è necessario procedere al decoding di $\boldsymbol{J}$: il vettore decodificato che chiamiamo $\hat{\boldsymbol{\xi}}$ dovrà nuovamente essere una sequenza di 7 bit. La decisione ottimale di decoding (ossia quella che minimizza la probabilità di errore) è quella di trovare

[1] Il *binary simmetric channel* è un modello di comunicazione basato sul fatto che ogni bit ha una probabilità $(1-p)$ di essere trasmesso correttamente e probabilità do essere p invertito:

$$P(y=0\,|\,x=0) = 1-p\,; \qquad P(y=0\,|\,x=1) = p\,;$$
$$P(y=1\,|\,x=0) = p\,; \qquad P(y=1\,|\,x=1) = 1-p\,.$$

per ogni bit quale valore di $\boldsymbol{\xi}$ è più probabile, una volta noto $\boldsymbol{J}$. Consideriamo per la decodifica di ogni singolo bit ξ_i la sequenza di tre bit $\boldsymbol{J}_i = J_{i_1} J_{i_2} J_{i_3}$. Per il *Teorema di Bayes* (per una trattazione formale si rimanda alla sezione §2.1.1) la probabilità di trovare ξ_i una volta noto $\boldsymbol{J}_i$ corrisponde a [2]

$$P(\xi_i \,|\, J_{i_1} J_{i_2} J_{i_3}) = \frac{P(J_{i_1} J_{i_2} J_{i_3} |\, \xi_i \,)\, P(\xi_i)}{P(J_{i_1} J_{i_2} J_{i_3})} \tag{1.1}$$

dove si definiscono:

- *prior* la probabilità $P(\xi_i)$;
- *likelihood* la probabilità $P(J_{i_1} J_{i_2} J_{i_3} |\, \xi_i \,)$;
- *posterior* la probabilità $P(\xi_i \,|\, J_{i_1} J_{i_2} J_{i_3})$.

Quindi, noti i due posterior $P(\xi_i = 0 \,|\, \boldsymbol{J}_i)$ e $P(\xi_i = 1 \,|\, \boldsymbol{J}_i)$, l'algoritmo di decoding assegna $\hat{\xi}_i = 1$ se $P(\xi_i = 1 \,|\, \boldsymbol{J}_i) > P(\xi_i = 0 \,|\, \boldsymbol{J}_i)$ o $\hat{\xi}_i = 0$ altrimenti. Nel calcolo di quest'ultimi, assumiamo per semplicità che le probabilità dei due prior siano uguali e note $P(\xi_i = 0) = P(\xi_i = 1) = 0.5$. Massimizzare la probabilità del posterior $P(\xi_i \,|\, \boldsymbol{J}_i)$ è equivalente a massimizzare la probabilità $P(\boldsymbol{J}_i \,|\, \xi_i)$. Poiché stiamo utilizzando un BSC, la probabilità $P(\boldsymbol{J}_i \,|\, \xi_i)$ è indipendente per ogni bit, da cui

$$P(\boldsymbol{J}_i \,|\, \xi_i) = P(\boldsymbol{J}_i(\xi_i) \,|\, \boldsymbol{J}_i^0(\xi_i)) = \prod_{n=1}^{M} P(J_{i_n}(\xi_i) \,|\, J_{i_n}^0(\xi_i)),$$

dove nel nostro caso $M = 3$. Ricordando che p è la probabilità che ogni bit venga mutato dal rumore del canale, possiamo scrivere $P(J_{i_n}(\xi_i) \,|\, J_{i_n}^0(\xi_i))$ come

$$P(J_{i_n}(\xi_i) \,|\, J_{i_n}^0(\xi_i)) = \begin{cases} 1-p & \text{se } J_{i_n} = J_{i_n}^0(\xi_i) \\ p & \text{se } J_{i_n} \neq J_{i_n}^0(\xi_i) \end{cases}.$$

Il rapporto delle due probabilità likelihood è dunque

$$\frac{P(\boldsymbol{J}_i \,|\, \xi_i = 1)}{P(\boldsymbol{J}_i \,|\, \xi_i = 0)} = \prod_{n=1}^{M} \frac{P(J_{i_n}(\xi_i) \,|\, J_{i_n}^0(1))}{P(J_{i_n}(\xi_i) \,|\, J_{i_n}^0(0))}, \tag{1.2}$$

dove ogni termine della produttoria è uguale a

- $\frac{1-p}{p}$ se $J_{i_n} = 1$,
- $\frac{p}{1-p}$ se $J_{i_n} = 0$.

Definendo infine il parametro $\gamma \equiv \frac{1-p}{p}$ (maggiore di 1 poiché $p = 0.1$ per ipotesi), il rapporto (1.2) sarà uguale a γ^s, con $s \in \mathbb{Z}$, da cui

- $\hat{\xi}_i = 0$ se $s < 0$;
- $\hat{\xi}_i = 1$ se $s > 0$.

[2] Ricordiamo che, dati due eventi A e B, si definisce probabilità condizionata di A rispetto a B la probabilità che si verifichi l'evento A, sapendo che l'evento B è verificato. In genere si indica con $P(A \,|\, B)$

Nel caso specifico dell'esempio proposto, l'algoritmo di decoding restituisce la seguente decodifica:

Sequenza ricevuta $\boldsymbol{J}_i$	$\frac{P(J_{i_n}(\xi_i) \mid J^0_{i_n}(1))}{P(J_{i_n}(\xi_i) \mid J^0_{i_n}(0))}$	bit decodificato $\hat{\xi}_i$
000	γ^{-3}	0
001	γ^{-1}	0
010	γ^{-1}	0
100	γ^{-1}	0
101	γ^{1}	1
110	γ^{1}	1
011	γ^{1}	1
111	γ^{3}	1

da cui segue la sequenza di output

$\boldsymbol{\xi}$	0	0	1	0	1	1	0	
$\boldsymbol{J}^0$	$\overbrace{000}$	$\overbrace{000}$	$\overbrace{111}$	$\overbrace{000}$	$\overbrace{111}$	$\overbrace{111}$	$\overbrace{000}$	+
$\boldsymbol{\Delta}$	000	001	000	000	101	000	000	=
$\boldsymbol{J}$	$\underbrace{000}$	$\underbrace{001}$	$\underbrace{111}$	$\underbrace{000}$	$\underbrace{010}$	$\underbrace{111}$	$\underbrace{000}$	
$\hat{\boldsymbol{\xi}}$	0	0	1	0	0	1	0	
		$\star$			$\times$			

Analizzando il risultato ottenuto si può vedere come l'algoritmo di decoding sia riuscito a correggere l'errore nella seconda terna ($\star$), mentre non è riuscito a correggere quello nella quinta ($\times$). É evidente come nel caso del repetion code R_3, l'algoritmo non è in grado di risolvere il problema di due errori in una singola tripletta. Ma è importante sottolineare che se p è la probabilità di errore sul singolo bit introdotto dal canale BSC, l'algoritmo R_3 riscala l'errore come $p_b = \mathcal{O}(p^2)$: in questo caso con $p = 0.1$ si ottiene $p_b \approx 0.03$. Aumentando il numero di copie M è possibile ridurre l'errore, andando però ad appesantire di molto il messaggio da inviare. Se infatti definiamo il rate di trasmissione R di un canale come

$$R := \frac{N}{N_B}, \tag{1.3}$$

dove N è il numero di bit del messaggio originale e N_B il numero di bit del messaggio di encoding (in generale, $N_B = M \cdot N$), ci si accorge che la velocità di trasmissione $R = 1/M$ del messaggio tenda a 0 al crescere di M.

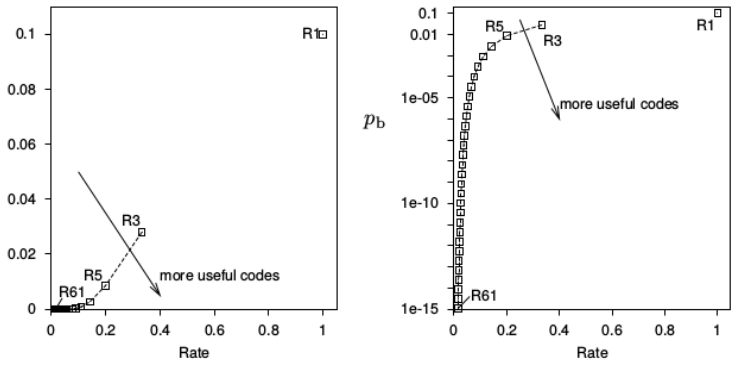


Figura 1.1: Andamento della probabilità di errore p_b in funzione del rate R per i repetition code $(R_1 \div R_6)$ in un BSC con $p = 0.1$. La freccia mostra l'ipotetica posizione di un codice con il miglior rendimento in termini di errore commesso e rate di trasmissione. Nel grafico a destra p_b è espresso in scala logaritmica [15].

1.1.3 Hamming code

Un altro importante algoritmo di encoding e decoding per un canale BSC è costituito dall'*Hamming code*. A differenza del precedente repetition code, questo si basa sull'idea di aggiungere ridondanza a interi blocchi di dati invece di codificare un singolo bit alla volta, un *block code*. In un block code (N_B, N), una sequenza di N bit viene codificata in una stringa di lunghezza $N_B > N$. Gli $N_B - N$ bit extra sono funzioni lineari degli N bit originali e vengono denominati *parity-bit*. Prendiamo in esame l'algoritmo di Hamming code $(7, 4)$, chiamiamo $\boldsymbol{\xi}$ il vettore originale e $\boldsymbol{J}^0$ quello codificato. Supponiamo di voler inviare il messaggio

$$\boldsymbol{\xi} = 1 \quad 0 \quad 0 \quad 0$$

L'algoritmo di encoding funziona come segue:

- i primi quattro bit vengono inviati uguali al messaggio originale, ossia $\xi_1\xi_2\xi_3\xi_4 = J_1^0\, J_2^0 J_3^0 J_4^0$:

$$\boldsymbol{J}^0 = 1 \quad 0 \quad 0 \quad 0 \quad * \quad * \quad *$$

- J_5^0 è la somma[3] dei primi tre bit $\xi_1 + \xi_2 + \xi_3$:

$$\boldsymbol{J}^0 = 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad * \quad *$$

- J_6^0 è la somma degli ultimi tre bit $\xi_3 + \xi_4 + \xi_5$:

$$\boldsymbol{J}^0 = 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad *$$

- J_7^0 è la somma del primo, del terzo e del quarto $\xi_1 + \xi_3 + \xi_4$:

$$\boldsymbol{J}^0 = 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1$$

[3]Le somme sono da intendersi in modulo 2, d'ora in avanti sarà dato come sottinteso.

Il canale introduce poi un errore $\boldsymbol{\Delta}$ come per esempio

$$\boldsymbol{\Delta} = 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1$$

da cui si ottiene il vettore $\boldsymbol{J} = \boldsymbol{J}^0 + \boldsymbol{\Delta}$ di output

$\boldsymbol{J}^0$	1	0	0	0	1	0	1	+
$\boldsymbol{\Delta}$	0	1	0	0	0	0	0	=
$\boldsymbol{J}$	1	1	0	0	1	0	1	

Se per ipotesi il canale di trasmissione è un BCS, allora ogni bit della sequenza codificata può essere modificato con eguale probabilità p, compresi i parity-bit. La decodifica ottimale corrisponde, quindi, a quella stringa $\boldsymbol{\xi}$ la cui codifica $\boldsymbol{J}^0$ differisce dal segnale ricevuto $\boldsymbol{J}$ per il minor numero di bit. Questo algoritmo di decoding può essere implementato per metodo grafico ed è rappresentato nella figura 1.2. Inseriamo i bit del vettore $\boldsymbol{J}$ all'interno di tre circonferenze come nella figura 1.2(a). Calcoliamo la parità z_i di ciascuna di esse e costruiamo il vettore $\boldsymbol{z} = (z_1, z_2, z_3)$ definito *sindrome*. É facile osservare che $\boldsymbol{J}^0 = \boldsymbol{J}$ solo se la parità di ogni circonferenza è pari a 0. In caso contrario, l'obiettivo è quello di modificare il minor numero di bit affinché la sindrome sia pari a $\boldsymbol{z} = (0, 0, 0)$. Nel caso riportato in esempio si ha $\boldsymbol{z} = (1, 1, 0)$, sintomo del fatto che il canale ha introdotto un errore nel messaggio di input. Osservando la figura 1.2(b), si nota come l'unico bit condiviso dalle due circonferenze tratteggiate ed esterno a quella continua sia $J_2 = 1$, risultando quindi il candidato ottimale per la correzione della sindrome. Ponendo infatti $J_2 = 0$ tutte le circonferenze assumono parità nulla, come espresso in figura 1.2(c).

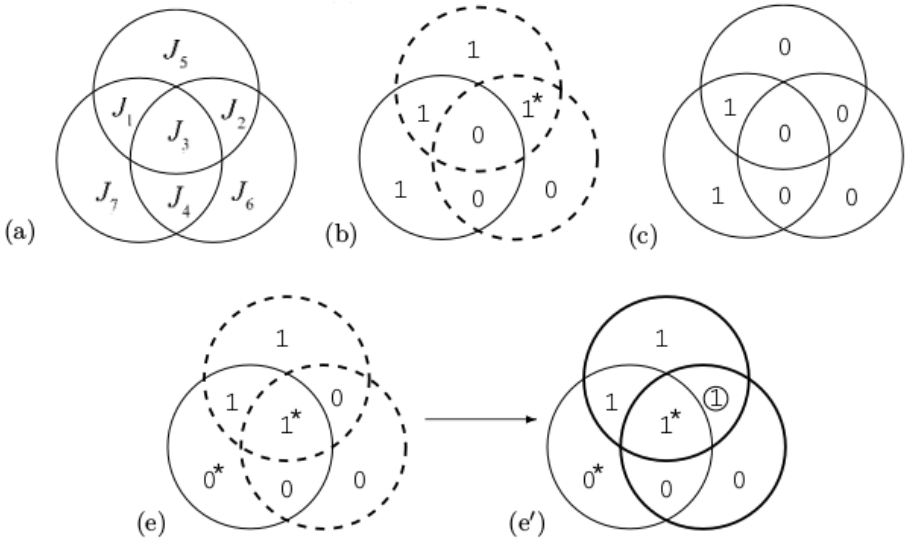


Figura 1.2: Rappresentazione grafica del decoding dell'Hamming code $(7, 4)$. Le circonferenze continue sono a parità uguale a 0, mentre quelle tratteggiate uguale a 1. I bit contrassegnati con * sono quelli mutati dal canale.

In questo modo è facile individuare per ogni sindrome $\boldsymbol{z}$ quale bit è responsabile dell'errore:

Sindrome $\boldsymbol{z}$	000	001	010	011	100	101	110	111
bit errato	/	J_7	J_6	J_5	J_4	J_1	J_2	J_3

Analogamente come per il repetition code, l'Hamming code presenta un limite nel caso in cui più di un bit venga mutato dal rumore del canale. In figura 1.2(e) vine mostrato il caso in cui due bit, ossia J_3 e J_7, vengono modificati. La sindrome $\boldsymbol{z} = (1, 1, 0)$ ci suggerisce come prima che il bit errato sia J_2, che viene pertanto modificato dall'algoritmo ottimale di decoding. Il risultato però è quello di un vettore di output $\boldsymbol{J}$ che contiene tre errori al posto di due, come in figura 1.2(e').

Definiamo *probabilità di errore di blocco* p_B, la probabilità che uno o più bit all'interno di un blocco codificato non siano uguali ai corrispondenti originali,

$$p_B := P(\hat{\boldsymbol{\xi}} \neq \boldsymbol{\xi}).$$

La *probabilità di errore per bit* p_b è data dunque dalla probabilità media che un singolo bit codificato non corrisponda al suo originale

$$p_b = \frac{1}{N} \sum_{i=1}^{N} P(\hat{\xi_i} \neq \xi_i).$$

Nel caso di Hamming code $(7, 4)$, un errore di decoding avviene solo se il rumore modifica più di un bit in una sequenza di sette. La probabilità di errore di blocco è quindi la probabilità che due o più bit vengano modificati in un blocco. Questa probabilità scala come $p_b = \mathcal{O}(p^2)$ analogamente al caso del repetition code R_3, ma è da sottolineare come il rate dell'Hamming code sia $R = 4/7$, contro $R = 1/3$ dell'R_3.

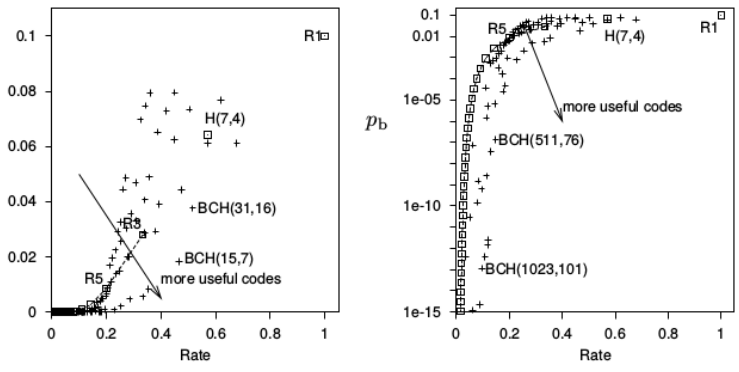


Figura 1.3: Andamento della probabilità di errore p_b in funzione del rate R per i repetition codes, per l'Hamming code (7,4) e per gli algoritmi BCH con lunghezza dei blocchi fino a 1023 bit. Il canale utilizzato è sempre di tipo BSC con probabilità di errore $p = 0.1$. Nel grafico a destra p_b è espresso in scala logaritmica [15].

1.1.4 Linear block codes

Un esempio molto più significativo di algoritmi di coding e decoding è costituito dai *linear block codes*. Prima però di poter presentare due esempi di questi codici è necessaria una breve introduzione. Supponiamo di voler inviare una sequenza di informazione binaria (contenente 0 o 1) $\boldsymbol{b}$ di lunghezza N. Un block code è una mappa lineare che associa ad ogni vettore binario $\boldsymbol{b} \in \mathbb{B}^N$ un vettore binario $\boldsymbol{c} \in \mathbb{B}^K$, con $K > N$, e K è la lunghezza della codifica. L'insieme di tutte le codifiche $\boldsymbol{c}$ costituisce il block code $\mathcal{C}$. Per ogni block code $\mathcal{C}$ corrispondente ad un codice (K, N), esistono 2^N distinte codifiche $\boldsymbol{c}$ identificabili univocamente. Diamo ora la seguente definizione:

Definizione 1.1.4.1. *Un block code (K, N) è detto lineare, se e solo se, forma uno sottospazio n-dimensionale della k-tuple binarie $\mathbb{B}^K$, $\mathcal{C} \subset \mathbb{B}^K$ con $dim\{\mathcal{C}\} = N$. Poichè forma un sottospazio esso contiene tutte le k-tuple nulle, i.e. $\mathbf{0}_{1\times K} \in \mathcal{C}$.*

Poichè un linear code forma un sottospazio di dimensione N, esistono N vettori binari indipendenti appartenenti a $\mathbb{B}^K$, che formano una base di dimensione N. In particolare, se denotiamo questi vettori indipendenti con $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_N$, allora ogni loro combinazione lineare genera una codifica $\mathcal{C}$. Inserendo i vettori all'interno di una matrice $N \times K$

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_N \end{bmatrix},$$

si crea una matrice detta *matrice generatrice*. Ogni codifica $\boldsymbol{c}$ può essere generata dalle righe di $\boldsymbol{G}$; usando la notazione dei paragrafi precedenti, sia $\boldsymbol{\xi}$ il messaggio originale di lunghezza N, questo viene codificato in un messaggio $\boldsymbol{J}^0 \in \mathcal{C}$ di lunghezza $K > N$ come:

$$\boldsymbol{J}^0 = \boldsymbol{\xi}\boldsymbol{G} \, mod \, 2.$$

Definizione 1.1.4.2. *Si definisce systematic linear block code, ogni linear block code che possiede una matrice generatrice $\boldsymbol{G}$ della forma:*

$$\boldsymbol{G}_s = \left[\boldsymbol{I}_N \, \boldsymbol{P} \right].$$

$\boldsymbol{I}_N$ denota la matrice identità $N \times N$, mentre la matrice $\boldsymbol{P}$ è una matrice binaria $N \times (K - N)$. Un systematic block code ha la proprietà che i primi N bit del messaggio codificato sono uguali agli N bit del messaggio di input, in questo modo è possibile risalire facilmente ai bit di informazione al termine della decodifica. Per ottenere la matrice generatrice sistematica $\boldsymbol{G}_s$, applichiamo permutazioni e operazioni fra righe alla matrice $\boldsymbol{G}$. Le matrici $\boldsymbol{G}$ e $\boldsymbol{G}_s$ sono dette equivalenti in quanto le generano le stesse codifiche. Ogni linear block code possiede una equivalente rappresentazione sistematica, in quanto qualunque scelta di N colonne linearmente indipendenti della matrice generatrice $\boldsymbol{G}$ forma una base dello spazio $\mathbb{B}^N$. Non di meno, lo spazio $\mathbb{B}^N$ può essere generato completamente dalle colonne di una matrice identità $N \times N$. Quindi, spostando le N colonne linearmente indipendenti, nelle prime N colonne di $\boldsymbol{G}$, dopo operazioni fra le righe è possibile ricreare la matrice identità $\boldsymbol{I}_N$.

Definizione 1.1.4.3. *La matrice parity check denominata $\boldsymbol{H}$ è definita come la matrice $(K - N) \times K$ le cui righe sono ortogonali alle righe della matrice $\boldsymbol{G}$.*

Definizione 1.1.4.4. *Due vettori non nulli* $\boldsymbol{c}_i$ *e* $\boldsymbol{c}_j$ *sono ortogonali se e solo se* $\boldsymbol{c}_i \cdot \boldsymbol{c}_j = 0$.

Ogni matrice generatrice sistematica $\boldsymbol{G}_s$ ha una sua corrispondente matrice parity check sistematica $\boldsymbol{H}_s$, i.e.

$$\boldsymbol{H}_s = \left[\boldsymbol{P}^T \boldsymbol{I}_{K-N} \right].$$

Ovviamente risulta vera la seguente catena di uguaglianze $\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{G}_s\boldsymbol{H}^T = \boldsymbol{G}\boldsymbol{H}_s^T = \boldsymbol{G}_s\boldsymbol{H}_s^T = \boldsymbol{0}_{N\times(K-N)}$. Per comodità di notazione d'ora in avanti considereremo $\boldsymbol{G} \equiv \boldsymbol{G}_s$ e $\boldsymbol{H} \equiv \boldsymbol{H}_s$, usando lo stesso simbolo per indicare sia la matrice generica che nella sua forma sistematica data la loro equivalenza. Presentiamo di seguito due esempi di codici appartenenti a questa classe.

Low-density parity-check codes (LDPC) o Gallager codes

I *Gallager codes* [6], detti anche *low-density parity-check codes* (LDPC), appartengono alla famiglia dei "very good" codes.

Definizione 1.1.4.5. *Si definisce "very good" code un code block che realizza una arbitrariamente piccola probabilità di errore per bit* p_b *a qualsiasi rate* R *di comunicazione al di sotto della capacità del canale* C.

Questi codici vengono definiti a partire dalla loro matrice parity check $\boldsymbol{H}$, e come indica il loro nome questa matrice ha la caratteristica di avere una struttura "very sparse".

Definizione 1.1.4.6. *Dato un vettore tipo binario* $\boldsymbol{b}$*, si definisce peso il numero dei suoi elementi non nulli.*

Definizione 1.1.4.7. *Si definisce densità il rapporto tra il numero di elementi non nulli e la dimensione del vettore o della matrice.*

Definizione 1.1.4.8. *Un vettore o una matrice si definiscono "sparse" se la loro densità è inferiore a 0.5. Mentre si definiscono "very sparse" se la densità tende a 0 all'aumentare della dimensione.*

Definizione 1.1.4.9. *Considerando un LDPC code con una matrice parity check* $\boldsymbol{H}$ *di dimensione* $(K-N)\times K$*, il codice si dice regolare se e solo se, un numero costante di bit codificati non nulli sono coinvolti in ciascuna equazione e, un costante numero di equazioni contiene ciascun simbolo. In particolare se indichiamo con* w_r *e* w_c *il numero di elementi non nulli in ciascuna riga e ciascuna colonna di* $\boldsymbol{H}$ *rispettivamente, allora se*

$$M\, w_r = K\, w_c, \tag{1.4}$$

dove $M = K - N$*, il codice è regolare. In ogni altro caso è irregolare.*

Per esempio il codice con la seguente matrice parity check

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix},$$

è regolare in quanto righe e colonne hanno un peso costante ($w_c = 2$ e $w_r = 4$).

Consideriamo di seguito il caso di un Gallager code regolare, quindi di codice con una matrice parity-check $\boldsymbol{H}$ "very sparse", in cui ogni riga ha lo stesso peso k e ogni colonna lo stesso peso j. La matrice parity check $\boldsymbol{H}$ che definisce il codice viene derivata a partire da una matrice "very sparse" $\boldsymbol{A}$ costruita nel seguente modo: definendo $M = K - N$ il numero di parity checks, si fissa peso w_c per colonna e si crea una matrice $\boldsymbol{A}$ rettangolare $M \times K$ random con esattamente peso w_c per colonna e peso $w_r = w_c\, K/M$ per riga. Attraverso un processo di eliminazione Gaussiana e di ordinamento delle colonne, si deriva una matrice parity-check in forma sistematica[4]

$$\boldsymbol{H}^T = \begin{bmatrix} \boldsymbol{P} \\ \boldsymbol{I}_M \end{bmatrix}$$

Ridefinendo $\boldsymbol{A} = \boldsymbol{H}$ si è nella condizione in cui la matrice $\boldsymbol{A} = [\boldsymbol{C}_1|\boldsymbol{C}_2]$ è composta da due matrici "very sparse" $\boldsymbol{C}_1$ e $\boldsymbol{C}_2$. La matrice $\boldsymbol{C}_2$ è una matrice quadrata di dimensione $M \times M$ invertibile, mentre $\boldsymbol{C}_1$ è rettangolare $M \times N$. A questo punto è possibile definire la matrice generatrice del Gallager code come

$$\boldsymbol{G}^T = \begin{bmatrix} \boldsymbol{I}_N \\ \boldsymbol{P}^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_N \\ \boldsymbol{C}_2^{-1}\boldsymbol{C}_1 \end{bmatrix}$$

dove l'espressione $\boldsymbol{P}^T = \boldsymbol{C}_2^{-1}\boldsymbol{C}_1$ è intesa come il prodotto matriciale in modulo 2.

Low-density generator-matrix codes (LDGM)

I *low-dendity generator-matrix codes* (LDGM), denominati nella letteratura di meccanica statistica come *codici di Sourlas*, appartengono invece alla famiglia dei "bad" codes.

Definizione 1.1.4.10. *Si definisce "bad" code un code block che realizza una arbitrariamente piccola probabilità di errore per bit p_b mandando il rate di informazione R a 0.*

La differenza sostanziale rispetto ai LDPC codes risiede nel fatto che questi vengono identificati non dalla matrice parity check $\boldsymbol{H}$, bensì direttamente dalla matrice generatrice $\boldsymbol{G}$. Fissati i parametri di ingresso N e K viene generata in modo random una matrice generatrice "very sparse" $\boldsymbol{G}$ in forma sistematica

$$\boldsymbol{G} = \left[\, \boldsymbol{I}_N \, \boldsymbol{P} \right].$$

Per entrambi questi codici presentati, supponendo di trasmettere il messaggio all'interno di un canale BSC, verrà aggiunto del rumore $\boldsymbol{\Delta}$ al vettore codificato $\boldsymbol{J}^0$, di conseguenza il segnale ricevuto $\boldsymbol{J}$ sarà dato da

$$\boldsymbol{J} = (\boldsymbol{\xi}\boldsymbol{G} + \boldsymbol{\Delta}) \mod 2. \tag{1.5}$$

Il decoding del messaggio corrotto avviene attraverso un algoritmo chiamato *belief propagation* il quale verrà affrontato in modo dettagliato nella sezione §3.2.

[4] C'è la possibilità che le righe originali della matrice non siano indipendenti; questo renderebbe il codice irregolare e si avrebbe una matrice $\boldsymbol{A}$ per un codice con lo stesso K ma più piccolo M, quindi con un rate maggiore.

1.2 Vetri di spin e ECC

Per approfondire lo studio degli ECC con gli strumenti teorici forniti dalla meccanica statistica, è conveniente applicare il modello di Ising dei vetri di spin al nostro problema utilizzando ± 1 al posto di 0 e 1. L'operazione di base per una sequenza di bit è la somma modulo 2, che corrisponde al prodotto di Ising spin se identifichiamo 0 con $S_i = 1$ e 1 con $S_i = -1$. Per esempio, pensiamo $0 + 1 = 1$ come $1 \times (-1) = -1$ e analogamente $1 + 1 = 0$ come $(-1) \times (-1) = 1$. Identificheremo dunque una sequenza di bit come una configurazione di Ising spin. La generazione di un parity-bit in un parity-check code sarà legata al prodotto di un appropriata sequenza di spin.

Per la costruzione di un modello analogo a quello di Ising, associamo un parametro di spin ξ_i per ogni sito, e definiamo l'interazione tra due siti i e j come $J_{ij}^0 = \xi_i \xi_j$. L'Hamiltoniana assume la forma

$$H = -\sum_{\langle ij \rangle} \xi_i \xi_j S_i S_j,$$

in cui il ground state è chiaramente definito dalla configurazione di spin in cui $S_i = \xi_i \ \forall i$ (o analogamente $S_i = -\xi_i \ \forall i$).

Ritornando all'error-correcting code, nel caso generale l'interazione è data da $\mathbf{J}^0 = \{J^0_{i_1 \cdots i_r} = \xi_{i_1} \cdots \xi_{i_r}\}$, con r intero, per opportune combinazioni di $\{i_1 \cdots i_r\}$. Pertanto, il vettore di dati che verrà poi immesso nel canale rumoroso sarà $\mathbf{J}^0$ invece che $\boldsymbol{\xi}$. Il messagio di encoding è ridondante rispetto a quello originale, dal momento che il numero di interazioni N_B è maggiore del numero di spin N.

In generale, il ground state dell'hamiltoniana di un canale rumoroso sarà diverso rispetto a quello dell'hamiltoniana di un canale error-free. Tuttavia, se il rumore del canale è sufficientemente piccolo, possiamo trattare quest'ultimo come una perturbazione del sistema error-free: in prima approssimazione la configurazione originale di spin costituisce ancora il ground state del sistema rumoroso. Infatti come è stato mostrato nei paragrafi precedenti, è possibile dedurre il messaggio originale anche se esiste una piccola quantità di rumore nel canale, a patto di adottare un'appropriata procedura di encoding e decoding. Nella teoria ECC si dimostra l'esistenza di una soglia di ridondanza del messaggio di encoding tale per cui è possibile ricostruire il messaggio originale senza errori. Questa soglia prende il nome di *limite di Shannon* [22]. Ricordando la definizione di *rate di trasmissione* espressa dalla (1.3), per un BSC si dimostra che è possibile una trasmissione senza errori nel limite di una sequenza molto lunga ($N \to \infty$):

$$R < C, \tag{1.6}$$

dove C è detta *capacità del canale* ed è uguale a

$$C := 1 - H_2(p) = 1 - \left[p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p} \right],$$

dove p è la probabilità che un bit venga mutato dal rumore. L'equazione (1.6) prende il nome di *channel coding theorem* di Shannon ed enuncia che è possibile trasmettere un messaggio privo di errore in un canale BSC nel momento in cui il rate non supera la capacità propria del canale stesso ed è implementata un'opportuna procedura di encoding e decoding. Un esempio di codice che satura asintoticamente il limite di Shannon è il *codice di Shannon* [24]: si prendono tutti i possibili

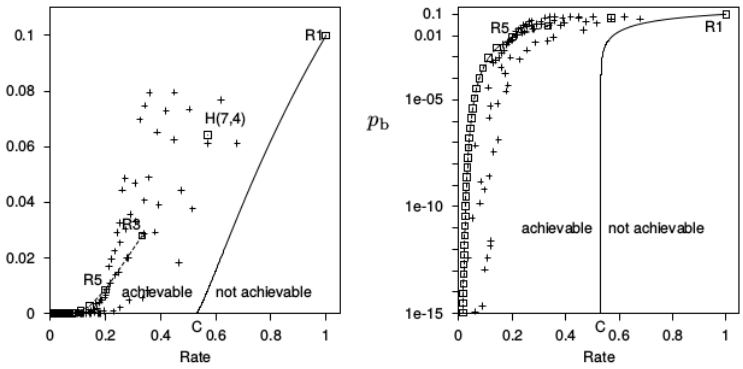


Figura 1.4: *Teorema della codifica di un canale* di Shannon. La curva continua rappresenta il limite di Shannon per un canale BSC con probabilità di errore $p = 0.1$. Rate tali che $R = C$, sono raggiungibili con una p_b arbitrariamente piccola quando $R \to \infty$. I punti mostrano la performance di alcuni algoritmi, tra cui i repetition codes e l'Hamming code $(7, 4)$. Nel grafico a destra p_b è espresso in scala logaritmica [15].

prodotti di r spin da N siti e, mandando $N \to \infty$ prima e $r \to \infty$ poi, si riesce a saturare il limite di Shannon. É importante notare, però, che in questo modo la disuguaglianza (1.6) si riduce ad una uguaglianza in cui entrambi i membri tendono a 0, in particolare $R \to 0$, quindi la trasmissione non è molto efficiente. All'inizio di questo paragrafo si è osservato come, per un canale poco rumoroso, il ground state del sistema di Ising spin coincida con la configurazione originale di spin $\boldsymbol{J}^0$. In presenza di un rumore maggiore il ground state non coincide con $\boldsymbol{J}^0$, questo suggerisce che tale configurazione sia uno degli stati eccitati del sistema. Un decoding più accurato potrebbe quindi essere effettuando ricercando stati ad una specifica temperatura finita T_p che dipende dal tasso di errore p.

Capitolo 2

Meccanica statistica dei codici di Sourlas

La parte che segue propone un approfondimento riguardo alla meccanica statistica alla base dei *codici di Sourlas*, fornendo una prima descrizione generale delle grandezze fondamentali necessarie per lo studio della teoria ECC con l'utilizzo del modello di vetri di spin.

2.1 Probabilità condizionata

Supponiamo che la configurazione di Ising spin $\boldsymbol{\xi} = \{\xi_i\}$ sia stata generata con una distribuzione di probabilità $P(\boldsymbol{\xi})$ chiamata *prior*. Il nostro obiettivo rimane quello di dedurre la configurazione originale di spin dall'output del canale rumoroso nel modo più accurato possibile. Seguendo il procedimento delle sezioni precedenti, definiamo interazione un insieme di prodotti di r spin come

$$J^0_{i_1 \dots i_r} = \xi_{i_1} \dots \xi_{i_r} \, (= \pm 1) \tag{2.1}$$

e supponiamo di inviare questo vettore all'interno del canale. Consideriamo sempre un BSC, allora l'output $J_{i_1 \dots i_r}$ è mutato rispetto all'input $J^0_{i_1 \dots i_r} = \xi_{i_1} \dots \xi_{i_r}$ in $-\xi_{i_1} \dots \xi_{i_r}$ con probabilità p, mentre con probabilità $1 - p$ viene restituito quello corretto.

La probabilità di output di un BSC può essere espressa in termini di *probabilità condizionata*:

$$P(J_{i_1 \dots i_r} \,|\, \xi_{i_1} \dots \xi_{i_r}) = \frac{\exp\Big(\beta_p J_{i_1 \dots i_r} \xi_{i_1} \dots \xi_{i_r}\Big)}{2 \, \cosh \beta_p}, \tag{2.2}$$

dove β_p è definito come

$$\beta_p : \qquad e^{2\beta_p} = \frac{1-p}{p}. \tag{2.3}$$

In accordo con la (2.3), l'equazione (2.2) è pari a $1 - p$ quando $J_{i_1 \dots i_r} = \xi_{i_1} \dots \xi_{i_r}$, ed è pari a p quando $J_{i_1 \dots i_r} = -\xi_{i_1} \dots \xi_{i_r}$. Questa è dunque una buona caratterizzazione per la probabilità condizionata del canale.

Assumendo che la (2.2) si applichi ad ogni set $(i_1 \dots i_r)$ indipendentemente, allora la probabilità totale è data dal prodotto delle singole probabilità

$$P(\boldsymbol{J} \mid \boldsymbol{\xi}) = \frac{1}{(2\cosh\beta_p)^{N_B}} \exp\Big(\beta_p \sum J_{i_1\dots i_r}\xi_{i_1}\dots\xi_{i_r}\Big), \tag{2.4}$$

dove la somma è presa su tutti i possibili set $(i_1 \dots i_r)$ che generano le interazioni definite dalla (2.1). Il parametro N_B è pari al numero di termini di questa sommatoria.

2.1.1 Formula di Bayes

Per lo scopo che ci siamo prefissati, ossia di dedurre $\boldsymbol{\xi}$ da $\boldsymbol{J}$, è necessario definire la probabilità $P(\boldsymbol{\xi} \mid \boldsymbol{J})$ detta *posterior*. Ricordando l'espressione della *probabilità congiunta*

$$P(A, B) = P(A|B)\,P(B) = P(B|A)\,P(A),$$

definita come la probabilità che si verifichino entrambi due eventi A e B, si ricava in modo naturale la *Formula di Bayes*

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} = \frac{P(B|A)\,P(A)}{\sum_C P(B|C)\,P(C)}. \tag{2.5}$$

A questo punto esprimiamo $P(\boldsymbol{\sigma} \mid \boldsymbol{J})$ sfruttando proprio la (2.5):

$$P(\boldsymbol{\sigma}|\boldsymbol{J}) = \frac{P(\boldsymbol{J}|\boldsymbol{\sigma})\,P(\boldsymbol{\sigma})}{Tr_\nu P(\boldsymbol{J}|\boldsymbol{\nu})\,P(\boldsymbol{\nu})}, \tag{2.6}$$

dove con $\boldsymbol{\sigma} = \{\sigma_1, \dots, \sigma_N\}$ abbiamo indicato le variabili dinamiche per il decoding. Per uniformare la notazione, il messaggio finale decodificato sarà indicato con $\hat{\boldsymbol{\xi}} = \{\hat{\xi}_1, \dots, \hat{\xi}_N\}$, mentre riserviamo $\boldsymbol{\xi} = \{\xi_1, \dots, \xi_N\}$ per la configurazione del messaggio originale.

Dalla (2.6) è possibile dedurre $\boldsymbol{\xi}$ solo una volto noto il prior $P(\boldsymbol{\sigma})$. Per semplificare l'analisi teorica, assumiamo che ogni messaggio sia generato con eguale probabilità[1], in questo modo $P(\boldsymbol{\sigma})$ può essere considerato costante. Sostituendo poi la (2.4) nella (2.6), il posterior assume la forma:

$$P(\boldsymbol{\sigma}|\boldsymbol{J}) = \frac{\exp\Big(\beta_p \sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)}{\mathrm{Tr}_\nu \exp\Big(\beta_p \sum J_{i_1\dots i_r}\nu_{i_1}\dots\nu_{i_r}\Big)}. \tag{2.7}$$

Se $\boldsymbol{J}$ è fissato, la (2.7) non è altro che il fattore di Boltzman[2] per un vetro di spin del modello di Ising con interazione $\boldsymbol{J}$. Abbiamo così stabilito un'equivalenza formale tra il problema dell'inferenza probabilistica dei messaggi per un BSC e la meccanica statistica dei vetri di spin.

[1] Questa assunzione non è necessariamente irrealistica in quanto l'informazione, prima della codifica, può essere considerata come una distribuzione quasi uniforme di 0 ed 1.

[2] Sia H l'hamiltoniana di Ising spin, si definisce *equazione di Gibbs-Boltzmann* la distribuzione di probabilità

$$P(\boldsymbol{S}) = \frac{e^{-\beta H}}{Z}$$

dove con $S \equiv \{S_i\}$ indichiamo l'insieme degli stati di spin e con $Z = \mathrm{Tr}\, e^{-\beta H}$ la funzione di partizione.

2.1.2 Maximum a posteriori probability (MAP) e maximer of posterior marginals (MPM)

Riprendendo l'osservazione già fatta all'interno della sezione §1.1.2, sappiamo che massimizzare $P(\boldsymbol{\sigma}|\boldsymbol{J})$ rispetto a $\boldsymbol{\sigma}$ equivale a massimizzare il posterior $P(\boldsymbol{J}|\boldsymbol{\sigma})$ nell'ipotesi di uniformità di $P(\boldsymbol{\sigma})$. In questo modo si identifica il ground state dell'hamiltoniana

$$H = -\sum J_{i_1\dots i_r}\sigma_{i_1\dots i_r}. \tag{2.8}$$

Questo metodo di decoding prende il nome di *maximum a posteriori probability* (MAP).

Un'altra procedura di decoding è il *maximer of posterior marginals* (MPM), il quale concentra la sua attenzione su un singolo bit e non sull'intera sequenza. Applicando alla (2.7) l'operatore traccia su $\sigma_{j\neq i}$ (operazione che prende il nome di *marginalizzazione*)) fissiamo tutte le variabili di spin a meno di σ_i, ottenendo il posterior di singolo spin

$$P(\sigma_i|\boldsymbol{J}) = \frac{\mathrm{Tr}_{\boldsymbol{\sigma}(\neq\sigma_i)}\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)}{\mathrm{Tr}_{\nu}\,\exp\Big(\beta_p\sum J_{i_1\dots i_r}\nu_{i_1}\dots\nu_{i_r}\Big)}. \tag{2.9}$$

Confrontando poi i risultati di $P(\sigma_i = 1|\boldsymbol{j})$ e $P(\sigma_i = -1|\boldsymbol{j})$, assegniamo all'$i$-esimo spin la decodifica $\hat{\xi} = 1$ se il primo è maggiore del secondo o viceversa $\hat{\xi} = -1$,

$$\hat{\xi}_i = \mathrm{sgn}\{P(\sigma_i = 1|\boldsymbol{j}) - P(\sigma_i = -1|\boldsymbol{j})\}.$$

Questa espressione può essere riscritta come

$$\hat{\xi}_i = \mathrm{sgn}\bigg(\sum_{\sigma_i=\pm1}\sigma_i P(\sigma_i|\boldsymbol{J})\bigg) = \mathrm{sgn}\bigg(\frac{\mathrm{Tr}_{\boldsymbol{\sigma}}\,\sigma_i P(\sigma_i|\boldsymbol{J})}{\mathrm{Tr}_{\boldsymbol{\sigma}}\,P(\sigma_i|\boldsymbol{J})}\bigg) := \mathrm{sgn}\langle\sigma_i\rangle_{\beta_p}. \tag{2.10}$$

dove $\langle\sigma_i\rangle_{\beta_p}$ costituisce la magnetizzazione locale. L'equazione (2.10) suggerisce di calcolare la magnetizzazione locale a una temperatura finita $T_p = \beta_p^{-1}$, e assegna il suo segno come valore di $\hat{\xi}_i$. Per quanto riguarda invece il MAP, chiaramente la minimizzazione dell'energia porta alla condizione $\beta \to \infty$.

2.1.3 Canale Gaussiano

Fino ad ora abbiamo sempre considerato un BSC, risulta però a volte conveniente considerare altri canali come il *canale Gaussiano*. In questo caso il segnale codificato $\xi_{i_1}\dots\xi_{i_r} = (= \pm1)$ è inviato all'interno del canale con una certa ampiezza $J_0\xi_{i_1}\dots\xi_{i_r}$. L'output risulta uniformemente distribuito intorno al suo input con una distribuzione Gaussiana di varianza J^2:

$$P(J_{i_1\dots i_r}|\xi_{i_1}\dots\xi_{i_r}) = \frac{1}{\sqrt{2\pi}J}\exp\bigg\{-\frac{(J_{i_1\dots i_r} - J_0\xi_{i_1}\dots\xi_{i_r})^2}{2J^2}\bigg\}.$$

Ripercorrendo i calcoli eseguiti nelle sezioni precedenti, supponendo sempre che il prior sia uniforme, il posterior può essere scritto usando la formula di Bayes come

$$P(\boldsymbol{\sigma}|\boldsymbol{J}) = \frac{\exp\Big((J_0/J^2)\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)}{\mathrm{Tr}_{\nu}\,\exp\Big((J_0/J^2)\sum J_{i_1\dots i_r}\nu_{i_1}\dots\nu_{i_r}\Big)}. \tag{2.11}$$

Confrontando questa con la (2.7) si può notare che il posterior di un canale Gaussiano corrisponde a quello di un BSC con $\beta_p = J_0/J^2$. I successivi argomenti posso essere quindi sviluppati per entrambi i canali in modo analogo.

2.2 Parametro di overlap

2.2.1 Misura della performance di decoding

Risulta conveniente introdurre una misura del successo della decodifica che esprima quanto il messaggio decodificato $\hat{\boldsymbol{\xi}}$ si avvicina a quello originale $\boldsymbol{\xi}$, in modo da avere una stima della correttezza dell'informazione ottenuta. Cosicché il ragionamento che segue sia valido sia per il metodo MAP e MPM, non specifichiamo il valore del parametro β.

Il prodotto di $\hat{\xi}_i$ e del corrispondente bit originale ξ_i ($\xi_i \langle \sigma_i \rangle_\beta$ dalla (2.10)) è pari a 1 se questi coincidono o -1 altrimenti. La strategia da seguire è quindi quella di massimizzare la probabilità che questo prodotto sia 1, infatti, nel momento in cui il prodotto tende a 1 per ogni bit, i vettori $\hat{\boldsymbol{\xi}}$ e $\boldsymbol{\xi}$ tendono a sovrapporsi perfettamente. Mediando tale prodotto sulla probabilità di output del canale $P(\boldsymbol{J}|\boldsymbol{\xi})$ e sul prior $P(\boldsymbol{\xi})$, definiamo il *parametro di overlap*

$$m(\beta) = \mathrm{Tr}_{\boldsymbol{\xi}} \sum_{\boldsymbol{J}} P(\boldsymbol{\xi}) P(\boldsymbol{J}|\boldsymbol{\xi}) \xi_i \, \mathrm{sgn} \langle \sigma_i \rangle_\beta . \tag{2.12}$$

Assumendo che il prior sia uniforme e pari a $P(\boldsymbol{\xi}) = 2^{-N}$, ricordando la (2.4), si può riscrivere la (2.12) come

$$m(\beta) = \frac{1}{2^N (2 \cosh \beta_p)^{N_B}} \sum_{\boldsymbol{J}} \mathrm{Tr}_{\boldsymbol{\xi}} \exp\Big(\beta_p \sum J_{i_1 \dots i_r} \xi_{i_1} \dots \xi_{i_r} \Big) \xi_i \, \mathrm{sgn} \langle \sigma_i \rangle_\beta . \tag{2.13}$$

L'overlap è strettamente legato alla nozione di *distanza di Hamming*, definita come il numero di bit diversi nelle corrispondenti posizioni tra due stringhe distinte. Più due messaggi sono simili tra loro maggiore è l'overlap e minore è la distanza di Hamming. Quando due messaggi coincidono, $\hat{\xi}_i = \xi_i \forall i$, allora $m(\beta) = 1$ e distanza di Hamming nulla, mentre per due messaggi completamente invertiti, $m(\beta) = -1$ e distanza di Hamming N.

2.2.2 Limite superiore dell'overlap

Il parametro di overlap $m(\beta)$ appena definito presenta una proprietà molto interessante, esso è una funzione non-monotona in β con un massimo in $\beta = \beta_p$,

$$m(\beta) \leq m(\beta_p). \tag{2.14}$$

Per dimostrare la (2.14) prendiamo il valore assoluto di entrambi i membri della (2.13) e osserviamo che:

$$m(\beta) \leq \frac{1}{2^N (2 \cosh \beta_p)^{N_B}} \sum_{\boldsymbol{J}} \Big| \mathrm{Tr}_{\boldsymbol{\xi}} \exp\Big(\beta_p \sum J_{i_1 \dots i_r} \xi_{i_1} \dots \xi_{i_r} \Big) \xi_i \, \mathrm{sgn} \langle \sigma_i \rangle_\beta \Big| =$$

$$= \frac{1}{2^N (2 \cosh \beta_p)^{N_B}} \sum_{\boldsymbol{J}} \frac{\{ \mathrm{Tr}_{\boldsymbol{\xi}} \, \xi_i \exp\Big(\beta_p \sum J_{i_1 \dots i_r} \xi_{i_1} \dots \xi_{i_r} \Big) \}^2}{\Big| \mathrm{Tr}_{\boldsymbol{\xi}} \, \xi_i \exp\Big(\beta_p \sum J_{i_1 \dots i_r} \xi_{i_1} \dots \xi_{i_r} \Big) \Big|} =$$

$$= \frac{1}{2^N(2\,\cosh\beta_p)^{N_B}} \sum_{\boldsymbol{J}} \mathrm{Tr}_{\boldsymbol{\xi}}\, \xi_i \exp\Big(\beta_p \sum J_{i_1\dots i_r}\xi_{i_1}\dots\xi_{i_r}\Big)\cdot$$
$$\cdot \frac{\mathrm{Tr}_{\boldsymbol{\xi}}\, \xi_i \exp\Big(\beta_p \sum J_{i_1\dots i_r}\xi_{i_1}\dots\xi_{i_r}\Big)}{\Big|\mathrm{Tr}_{\boldsymbol{\xi}}\, \xi_i \exp\Big(\beta_p \sum J_{i_1\dots i_r}\xi_{i_1}\dots\xi_{i_r}\Big)\Big|}.$$

Riprendendo la definizione (2.10) di $\langle\sigma_i\rangle_{\beta_p}$ osserviamo che possiamo riscriverla nella forma:

$$\begin{aligned}
\mathrm{sgn}\langle\sigma_i\rangle_{\beta_p} &= \mathrm{sgn}\sum_{\sigma_i=\pm1}\sigma_i P(\sigma_i|\boldsymbol{J}) = \\
&= \mathrm{sgn}\sum_{\sigma_i=\pm1}\sigma_i \frac{\mathrm{Tr}_{\boldsymbol{\sigma}(\neq\sigma_i)}\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)}{\mathrm{Tr}_{\boldsymbol{\nu}}\exp\Big(\beta_p\sum J_{i_1\dots i_r}\nu_{i_1}\dots\nu_{i_r}\Big)} = \\
&= \mathrm{sgn}\,\frac{\mathrm{Tr}_{\boldsymbol{\sigma}}\,\sigma_i\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)}{\Big|\mathrm{Tr}_{\boldsymbol{\nu}}\exp\Big(\beta_p\sum J_{i_1\dots i_r}\nu_{i_1}\dots\nu_{i_r}\Big)\Big|} = \\
&= \mathrm{sgn}\,\frac{\mathrm{Tr}_{\boldsymbol{\sigma}}\,\sigma_i\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)}{\Big|\mathrm{Tr}_{\boldsymbol{\sigma}}\,\sigma_i\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)\Big|}.
\end{aligned}$$

Nella catena di uguaglianze, alla terza riga, si osserva che il contributo alla funzione *segno* è dato solo dal numeratore, in quanto il denominatore è una somma di grandezze positive[3]. Sostituiamo quindi al denominatore un'altra grandezza strettamente positiva e, in virtù della definizione della funzione segno, si ha

$$sgn\frac{\mathrm{Tr}_{\boldsymbol{\sigma}}\,\sigma_i\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)}{\Big|\mathrm{Tr}_{\boldsymbol{\sigma}}\,\sigma_i\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)\Big|} = \frac{\mathrm{Tr}_{\boldsymbol{\sigma}}\,\sigma_i\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)}{\Big|\mathrm{Tr}_{\boldsymbol{\sigma}}\,\sigma_i\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)\Big|}$$

Possiamo pertanto riscrivere la disuguaglianza precedente come:

$$m(\beta) \le \frac{1}{2^N(2\,\cosh\beta_p)^{N_B}} \sum_{\boldsymbol{J}} \mathrm{Tr}_{\boldsymbol{\xi}}\,\xi_i\exp\Big(\beta_p\sum J_{i_1\dots i_r}\sigma_{i_1}\dots\sigma_{i_r}\Big)\ \mathrm{sgn}\langle\sigma_i\rangle_{\beta_p} = m(\beta_p).$$

Abbiamo quindi dimostrato che il parametro di overlap ha un massimo proprio in corrispondenza di $\beta = \beta_p$. Questo significa che il metodo MPM valutato in $\beta = \beta_p$ presenta il risultato ottimale in termini di massimizzazione dell'overlap, infatti prende anche il nome di *strategia ottimale di Bayes*. Notiamo inoltre che, nonostante il MAP massimizzi il posterior $P(\boldsymbol{\xi}|\boldsymbol{J})$, la sua probabilità di errore per un singolo bit è maggiore che per l'MPM valutato in β_p.

[3] In generale $\mathrm{Tr}_{\boldsymbol{\nu}}\exp(\dots) > 0$, il che implica $\mathrm{Tr}_{\boldsymbol{\nu}}\exp(\dots) = |\mathrm{Tr}_{\boldsymbol{\nu}}\exp(\dots)|$.

Parte II

IMPLEMENTAZIONE NUMERICA DELL'ALGORITMO DI BELIEF PROPAGATION PER LA DECODIFICA DEI CODICI CORRETTORI

Capitolo 3

Rappresentazione dei codici LDGM in factor graph e l'algoritmo di belief propagation

Nel capitolo precedente abbiamo mostrato alcune generalità riguardo alla meccanica statistica dei codici correttori. Obiettivo di questo capitolo è definire una procedura tale per cui sia possibile costruire un algoritmo di correzione degli errori mediante il modello di vetri di spin.

3.1 L'hamiltoniana H dei codici LDGM

3.1.1 Definizione di H

Nella sezione §2.1 è stato dimostrato come l'obiettivo dell'algoritmo di deconding (sempre nell'ipotesi di uniformità della probabilità $P(\boldsymbol{\sigma})$) sia quello di massimizzare la probabilità $P(\boldsymbol{\sigma}|\boldsymbol{J})$ affinché si possano ottenere prestazioni ottimali. É stato inoltre verificato come, in virtù del modello di vetri di spin, fare ciò sia totalmente analogo ad identificare il ground state dell'hamiltoniana (2.8). Supponiamo quindi di voler inviare un messaggio $\boldsymbol{\xi}$ contenente N bit (± 1) attraverso un canale rumoroso di tipo BSC: ciò significa che ogni bit ξ_i del messaggio ha una probabilità p di essere mutato in $-\xi_i$. Ridefiniamo dunque la funzione hamiltoniana (2.8) con l'aggiunta di un campo magnetico esterno:

$$H(\boldsymbol{\sigma}) := -\sum_{i_1 \cdots_K}^{N} J_{i_1,\ldots,i_K}\sigma_{i_1}\ldots\sigma_{i_K} - \sum_{l}^{N} h_l\sigma_l \tag{3.1}$$

dove K indica il numero di corpi interagenti, mentre $\boldsymbol{\sigma}$ è il vettore di spin (± 1, di dimensione N) da cui dipende H. La definizione di $J_{i_1,\dots,i_K}$ è la seguente:

$$J_{i_1,\dots,i_K} := \begin{cases} -\xi_{i_1}\dots\xi_{i_K} & \text{con probabilità } p \\ +\xi_{i_1}\dots\xi_{i_K} & \text{con probabilità 1 - p} \\ 0 & \text{se } \nexists \text{ l'interazione tra } \xi_{i_1}\dots\xi_{i_K} \end{cases} \tag{3.2}$$

e in modo del tutto analogo per $\boldsymbol{h}$:

$$h_l := \begin{cases} -\xi_l & \text{con probabilità } p \\ +\xi_l & \text{con probabilità } 1-p \end{cases} \tag{3.3}$$

Questa hamiltoniana ha un'interpretazione grafica che può essere ottenuta tramite un *factor graph*.

3.1.2 Rappresentazione di H tramite un factor graph

Dal momento che l'implementazione tramite grafi dell'hamiltoniana di cui sopra è necessaria per la comprensione della simulazione del codice di Sourlas, diamo di seguito alcune definizioni fondamentali che porteranno successivamente all'introduzione del factor graph bipartito [13]:

Definizione 3.1.2.1. *Un factor graph è una coppia ordinata $\mathcal{G} = \mathcal{G}(E, V)$, dove V è l'insieme dei vertici (o nodi) ed E l'insieme dei lati, tale per cui gli elementi di E siano coppie di elementi di V ($E \subseteq V \times V$).*

Definizione 3.1.2.2. *Dato un lato $e \in E$ di un grafo $\mathcal{G}$, ci sono due vertici $v_1, v_2 \in V$ tali per cui $e = (v_1, v_2)$: diciamo che v_1 e v_2 sono adiacenti. Si dice che il grafo non è direzionato se $(v_1, v_2) = (v_2, v_1)$.*

Definizione 3.1.2.3. *Il grado di un vertice $v \in V$ è il numero dei suoi vertici adiacenti.*

Definizione 3.1.2.4. *Un grafo bipartito è un grafo in cui i suoi vertici possono essere suddivisi in due insiemi disgiunti U e V tali per cui ogni vertice $u \in U$ è collegato ad un vertice $v \in V$.*

Note queste definizioni minimali, spostiamo l'attenzione verso i factor graph. Per fare questo, iniziamo col considerare un insieme di variabili $x_1, \dots, x_N$ tali che $x_i \in \mathcal{X}_i$ (dominio finito) e $f(x_1, \dots, x_N)$ sia una funzione reale di dominio $\mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$ (detto *spazio di configurazione*). Supponiamo ora che f sia fattorizzabile in un prodotto di M funzioni locali f_j (dette appunto *fattori*), ciascuna con insiemi di variabili $S_j \subseteq x_1, \dots, x_N$, ossia

$$f(x_1, \dots, x_N) = \prod_{j=1}^{M} f_j(S_j). \tag{3.4}$$

Chiaramente ogni funzione f_j ha il proprio spazio i configurazione. Siamo dunque pronti per dare la seguente definizione:

Definizione 3.1.2.5. *Sia $f(x_1, \dots, x_N)$ una funzione tale che possa essere fattorizzata come in* (3.4)*, un factor graph $\mathcal{G}(E, V)$ corrispondente alla funzione f è un grafo bipartito tale che:*

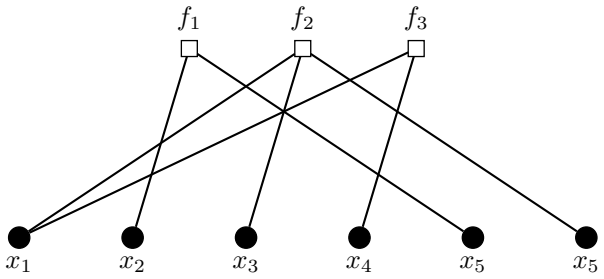


Figura 3.1: Factor graph della funzione (3.5). I fattori f_j sono rappresentati da quadrati bianchi, i vertici x_i da cerchi neri.

- *ogni variabile x_i è associata a un vertice denotato con un cerchio nero;*
- *ogni fattore f_j è associato a un quadrato bianco;*
- *se x_i è nel dominio di f_j, esiste un lato del grafo $e_{ij} = (x_i, f_j)$ che collega il cerchio x_i al quadrato f_j.*

Non è difficile convincersi del fatto che ad ogni funzione f fattorizzata corrisponda un solo factor graph $\mathcal{G}(E, V)$ e viceversa, in virtù del fatto che la funzione che mappa f nel factor graph è una funzione uno-a-uno. Un rapido esempio è il seguente: sia f una funzione di 3 variabili tali che

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = f_1(x_2, x_5) f_2(x_1, x_3, x_6) f_3(x_1, x_4). \tag{3.5}$$

La Figura 3.1 rappresenta il corrispettivo factor graph.

Questa breve introduzione sui factor graph permette, quindi, sia di dare un'interpretazione grafica alla funzione hamiltoniana, sia di poterla riscrivere in modo più semplice in vista di alcuni conti che seguiranno. Riprendendo quanto detto in precedenza, il messaggio di ingresso è un vettore $\boldsymbol{\xi}$ di N bit, mentre con $\boldsymbol{\sigma}$ indichiamo il vettore di N variabili dell'hamiltoniana (3.1). A questo punto dobbiamo imporre alcune condizioni sul factor graph che rappresenta H.

- Assumiamo che il numero di interazioni $\boldsymbol{\mu}_i$ sia pari ad M. Ciò significa che se f è la funzione che descrive il factor graph, questa può essere scritta come il prodotto di M fattori f_j. Nel grafo, dunque, devono comparire M quadrati bianchi.
- Dal momento che M descrive un'interazione di K spin, ogni interazione $\boldsymbol{\mu}_i$ deve essere della forma $\boldsymbol{\mu}_i = (\sigma_{i_1}, \ldots, \sigma_{i_K})$. Nel grafo, dunque, ogni quadrato (ossia ogni interazione $\boldsymbol{\mu}_i$) deve essere collegato con K cerchi neri (ossia con gli spin $\sigma_{i_1}, \ldots, \sigma_{i_K}$).
- Assumiamo che ogni spin σ_i interagisca con un numero di C di spin fissato e uguale per ogni σ_i. Quindi, ogni cerchio nero deve essere collegato con C quadrati bianchi (cioè deve prendere parte a C interazioni $\boldsymbol{\mu}_i$ diverse).

Le condizioni che abbiamo posto implicano la seguente uguaglianza:

$$M \cdot K = N \cdot C. \tag{3.6}$$

Infatti, ogni cerchio nero è collegato con C quadrati bianchi, quindi ai cerchi neri corrispondono in totale $N \cdot C$ lati. Allo stesso modo, ogni quadrato bianco è collegato con K cerchi neri, per un totale di $M \cdot K$ lati. Questi due prodotti devono ovviamente coincidere. in aggiunta, richiediamo che ogni spin σ_i partecipi a interazioni tutte diverse tra loro. Ciò è possibile solo se

$$M > C. \tag{3.7}$$

Dunque, l'hamiltoniana (3.1) può essere riscritta nel seguente modo:

$$H(\boldsymbol{\sigma}) = -\sum_{i}^{M} J_{\boldsymbol{\mu}_i} \sigma_{\boldsymbol{\mu}_i} - \sum_{l}^{N} h_l \sigma_l, \tag{3.8}$$

dove $\sigma_{\boldsymbol{\mu}_i} = \prod_{j=1}^{K} \sigma_{i_j}$. Da ultimo, osserviamo che il rate R di trasmissione è pari a

$$R = \frac{K}{K+C}. \tag{3.9}$$

Infatti, $J_{\boldsymbol{\mu}}$ e $\boldsymbol{h}$ trasmettono rispettivamente M e N bit. A questo punto, tramite l'identità (3.6) si trova l'espressione di R.

3.1.3 Costruzione del factor graph

Una volta definita l'hamiltoniana dei codici LDGM (3.1) e fornita un'interpretazione grafica tramite i factor graph (sezione § 3.1.2). Affrontiamo ora la questione di come costruire un factor graph casuale secondo le restrizioni imposte, che ricordiamo essere:

- $M \cdot K = N \cdot C$,
- $M > C$.

Abbiamo già verificato che un factor graph con queste caratteristiche esiste e rimane ben definito: rimane da capire quale algoritmo possa generarlo.

La strategia che seguiremo per creare un factor graph del tutto casuale consiste fondamentalmente in due step:

- generiamo un particolare factor graph facilmente definibile tramite un algoritmo;
- eseguiamo una permutazione casuale dei lati di questo factor graph.

Riportiamo di seguito i passaggi fondamentali per la costruzione di un factor graph generico. Supponiamo fissati i parametri N,C,M e K. Quindi:

(a) Colleghiamo il primo spin σ_1 con le prime C interazioni (quadratini bianchi). Dal momento che $M > C$, non può ai capitare che σ_1 venga collegato due volte con la stessa interazione.

(b) In modo analogo, colleghiamo il secondo spin σ_2 con le successive C interazioni. Se si giunge all'ultima interazione $\boldsymbol{\mu}_M$, semplicemente si ricomincia a collegare lo spin con la prima di queste, e così via.

(c) Si itera questa operazione per ogni spin σ_i. La prima delle due condizioni che abbiamo posto nella creazione del grafo non garantisce che questa procedura sia sempre fattibile. Tuttavia, una volta definiti K e C in modo arbitrario, ponendo semplicemente N uguale ad un multiplo intero di K e $M = N \cdot C/K$ ci si convince facilmente che la costruzione di questo grafo è sempre possibile.

(d) Scegliamo casualmente due spin diversi σ_i e σ_j, e di seguito due interazioni $\boldsymbol{\mu}_a \neq \boldsymbol{\mu}_b$ tali che σ_i sia collegato con $\boldsymbol{\mu}_a$ e σ_j con $\boldsymbol{\mu}_b$. Supponendo che σ_i non sia collegato a $\boldsymbol{\mu}_b$ e ugualmente σ_j con $\boldsymbol{u}_a$, scambiamo dunque questi due lati collegando σ_i con $\boldsymbol{\mu}_b$ e σ_j con $\boldsymbol{\mu}_a$.

(e) Iteriamo il passaggio precedente, dando luogo a una permutazione del grafo di partenza.

La Figura 3.2 mostra passo passo le fasi di costruzione del grafo.

Rimane soltanto da definire quanti scambi tra due lati si debba eseguire per far sì che la permutazione del grafo iniziale dia origine a un grafo totalmente casuale. Ricordiamo che, per come vengono eseguite queste simulazioni, si ha che $C \ll N$. Da questo segue che il numero di lati, che ricordiamo essere $N \cdot C$, è tale per cui $N \cdot C \ll N^2$. Se assumiamo che la permutazione totale abbia esattamente N^2 scambi, nel limite termodinamico ogni lato del grafo viene scambiato almeno una volta con un altro lato (in realtà, da un punto di vista statistico ogni lato viene scambiato con un altro un numero di volte $\gg 1$). Questo garantisce che il grafo finale sia totalmente casuale.

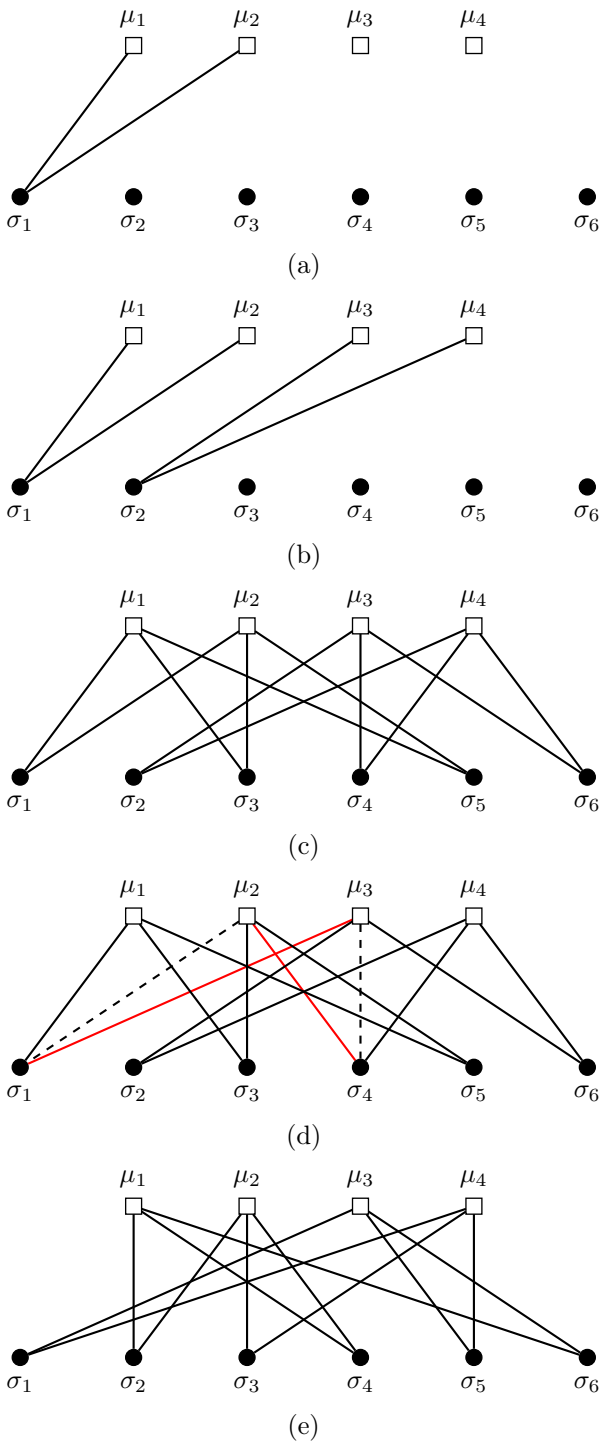


Figura 3.2: Step di costruzione di un grafo casuale con $N = 6$, $M = 4$, $K = 3$ e $C = 2$. Nella sottosezione §3.1.3 sono riportate le descrizioni dei passaggi fondamentali per la costruzione di questa tipologia di factor graph.

3.2 Algoritmo di Belief Propagation

3.2.1 Funzione marginale e proprietà distributiva generalizzata (GLD)

L'algoritmo implementato per la decodifica del messaggio trasmesso $\hat{\boldsymbol{\xi}}$ prende il nome di *belief propagation* (BP) (o *sum-product message- passing alghorithm*, SPA). Questo algoritmo si basa sul calcolo del *marginale* di ciascuna variabile di una funzione f globalmente fattorizzabile. Prima di entrare nello specifico del processo di decodifica è necessaria una breve introduzione su cosa sono i marginali e sull'enorme importanza che rivestono da un punto di vista computazionale.

Definizione 3.2.1.1. *Data una funzione $f = f(X_1, \dots, X_N)$, si definisce marginale di f rispetto alla variabile X_i, una funzione $\mathcal{X}_i \mapsto \mathbb{R}$ denominata $g_{X_i}(x_i)$ ottenuta dalla somma su tutte le altre variabili. In particolare $\forall x_i \in \mathcal{X}_i$ il valore $g_{X_i}(x_i)$ è ottenuto dalla somma dei valori $f(X_1, \dots, X_N)$ su tutte le configurazioni delle variabili di input aventi $X_i = x_i$:*

$$g_{X_i}(x_i) = \sum_{x_1 \in \mathcal{X}} \cdots \sum_{x_{i-1} \in \mathcal{X}_{i-1}} \sum_{x_{i+1} \in \mathcal{X}_{i+1}} \cdots \sum_{x_N} f(X_1 = x_1, \dots, X_i = x_i, \dots, X_N = x_N). \tag{3.10}$$

D'ora in avanti, per semplificare la notazione, sarà comodo indicare non le variabili sommate bensì le variabili non sommate utilizzando la seguente scrittura:

$$g_{X_i}(x_i) = \sum_{\sim\{x_i\}} f(x_1, \dots, x_i, \dots, x_N) = \sum_{x_1, \dots x_{i-1}, x_{i+1}, \dots, x_N} f(x_1, \dots, x_i, \dots, x_N). \tag{3.11}$$

Un modo molto efficiente per la computazione dei marginali è l'applicazione della *proprietà distributiva generalizzata* (GLD):

$$(a + b)c = ac + bc,$$

dove a, b e c sono elementi di un campo arbitrario $\mathbb{F}$. Come si può notare il secondo membro dell'equazione coinvolge tre operazioni aritmetiche (una addizione e due moltiplicazioni), mentre il primo membro solamente due. Il seguente esempio illustrerà in maniera più significativa l'importanza della GLD. Consideriamo la funzione fattorizzata f a sei variabili in un dominio finito $\mathcal{X}_i = \mathcal{X}$, $i = 1, \dots, 6$:

$$f(X_1, X_2, X_3, X_4, X_5, X_6) = f_1(X_1, X_4) f_2(X_1, X_3, X_6) f_3(X_2, X_4, X_5) f_4(X_1). \tag{3.12}$$

Se volessimo calcolare il marginale rispetto alla variabile X_3, $g_{X_3}(x_3)$, secondo la definizione (3.11) avremmo:

$$g_{X_3}(x_3) = \sum_{\sim\{x_3\}} f_1(x_1, x_4) f_2(x_1, x_3, x_6) f_3(x_2, x_4, x_5) f_4(x_1). \tag{3.13}$$

La computazione di questo marginale richiederebbe un algoritmo di complessità $\mathcal{O}(|\mathcal{X}|^6)$, poiché la somma su tutte le possibili variabili X_1, X_2, X_4, X_5, X_6 è di ordine $\mathcal{O}(|\mathcal{X}|^5)$ e $\mathcal{O}(|\mathcal{X}|)$ per la valutazione della funzione $g_{X_3}(x_3)$ per tutti i possibili

valori di X_3, dove $|\mathcal{X}|$ indica la cardinalità del dominio $\mathcal{X}$. Se ora applichiamo la GLD alla (3.13) si ottiene

$$g_{X_3}(x_3) = \sum_{x_1}\Big\{f_4(x_1)\Big(\sum_{x_6} f_2(x_1, x_3, x_6)\Big)\Big[\sum_{x_4}\Big(f_1(x_1, x_4)\sum_{x_2,x_5} f_3(x_2, x_4, x_5)\Big)\Big]\Big\},$$

equivalente a

$$g_{X_3}(x_3) = \sum_{\sim\{x_3\}}\Big[f_4(x_1)\Big(\sum_{\sim\{x_1\}} f_2(x_1, x_3, x_6)\Big)\Big(\sum_{\sim\{x_1\}} f_1(x_1, x_4)f_3(x_2, x_4, x_5)\Big)\Big].$$

L'ultimo termine nella parentesi è di ordine $\mathcal{O}(|\mathcal{X}|^3)$ per la somma sulle variabili X_2, X_5 e X_4. Considerando di seguito la prima somma su X_1 e, infine, la valutazione del marginale $g_{X_3}(x_3)$ per tutti i valori di X_3, porta la complessità ad ordine $\mathcal{O}(|\mathcal{X}|^5)$. Seguendo un procedimento analogo per il calcolo del marginale rispetto alla variabile X_1, $g_{X_1}(x_1)$, secondo la definizione (3.11) e applicando la GLD si ottiene:

$$g_{X_1}(x_1) = \Big[\sum_{\sim\{x_1\}} f_4(x_1)f_1(x_1, x_4)\Big(\sum_{\sim\{x_4\}} f_3(x_2, x_4, x_5)\Big)\Big]\Big(\sum_{\sim\{x_1\}} f_2(x_1, x_3, x_6)\Big).$$

In questo caso la complessità delle operazioni viene ridotta da $\mathcal{O}(|\mathcal{X}|^6)$ a $\mathcal{O}(|\mathcal{X}|^4)$. Si può concludere quindi che la GLD riduce significativamente la complessità computazionale dei marginali.

Quando una funzione fattorizzata ha una struttura ad albero, siamo in grado di calcolare esattamente il marginale rispetto a qualunque variabile in modo esatto. Per prima cosa è necessario costruire una struttura ad albero dell'espressione del marginale da calcolare come quello in Figura 3.3 per l'espressione 3.16. Notiamo che in un grafo di questo tipo i nodi interni (sia variabili che fattori) hanno ≥ 1 nodi figli, mentre ciascun nodo ha un solo nodo genitore. Nel caso di nodo foglia non si alcun nodo figlio e un solo nodo genitore. È quindi possibile creare una procedura di risalita in cui, iniziando dai nodi foglia, vengono inviati i "messaggi" ai nodi genitori. A loro volta questi inviano il messaggio ai propri nodi genitori e così via, finchè il procedimento viene terminato al raggiungimento del nodo radice, ossia un nodo che non abbia nodi genitori, ma solo nodi figli. Con il termine "messaggio" ci si riferisce all'informazione che passa attraverso il lato del factor graph di due nodi adiacenti. Considerando un nodo i con $N-1$ nodi figli e 1 nodo genitore, il calcolo del messaggio di uscita dal nodo i al suo genitore j richiede che siano stati ricevuti tutti messaggi in uscita da nodi figli. Ricordando che la struttura dei factor graph che stiamo considerando è quella descritta nella sezione precedente, sottolineiamo che il calcolo del messaggio di uscita da un nodo variabile differisce da quello di un nodo fattore. Consideriamo un nodo variabile X_i con N nodi fattori adiacenti e un solo f_j nodo genitore ovviamente adiacente ad esso ($f_j \in \mathcal{N}(X_i) = f_1, \ldots, f_N$). Il messaggio di uscita da un nodo variabile X_i al suo nodo genitore f_j, denominato $\mu_{X_i \to f_j}(x_i)$ è dato da

$$\mu_{X_i \to f_j}(x_i) = \prod_{f_k \in \mathcal{N}(X_i)\setminus\{f_j\}} \mu_{f_k \to X_i}(x_i), \tag{3.14}$$

i.e. il prodotto di tutti i messaggi in entrata provenienti dai nodi fattore figli. Consideriamo ora un nodo fattore f_j che ha M nodi variabili adiacenti $X_1, \ldots, X_M$,

e un nodo variabile X_i che è l'unico nodo genitore del nodo fattore f_j e ovviamente adiacente ad esso ($X_i \in \mathcal{N}(f_j) = \{X_1, \ldots, X_M\}$). Allora il messaggio di uscita dal nodo fattore f_j al suo nodo genitore X_i, denominato $\mu_{f_j \to X_i}(x_i)$, è dato da

$$\sum_{\sim\{x_i\}} f_j(x_1, \ldots, x_i, \ldots, x_M) \prod_{X_l \in \mathcal{N}(f_j)\setminus\{X_i\}} \mu_{X_i \to f_j}(x_l), \tag{3.15}$$

i.e. la somma su tutte le configurazioni possibili degli argomenti del nodo fattore f_j con $X_i = x_i$, moltiplicato per il prodotto di tutti i messaggi in entrata proveniente dai nodi variabile figli. Nel caso di nodi foglia, il messaggio in uscita da un nodo variabile foglia al suo nodo fattore genitore è la costante 1, mentre il messaggio da un nodo fattore foglia al suo nodo variabile genitore è pari al valore della sua funzione locale. Questo processo prende il nome di *single sum-product algorithm*, poiché computa il marginale di una singola variabile della funzione fattorizzata. Ricordiamo che il messaggio di uscita da ciascun nodo al suo nodo genitore, può avvenire solo se si hanno a disposizione tutti i messaggi in entrata provenienti dai propri nodi figli.

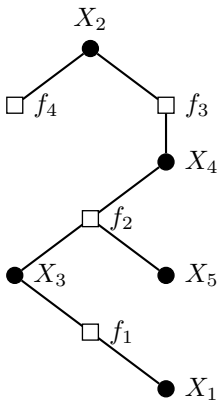


Figura 3.3: Factor graph corrispondente alla funzione (3.16). Questo grafo non contenendo cicli ha una struttura ad albero.

Per evitare confusioni sulla procedura, con il prossimo esempio illustreremo passo passo i passaggi da seguire.
Consideriamo la funzione fattorizzabile $f(X_1, X_2, X_3, X_4, X_5)$ a cinque variabili X_1,X_2,X_3,X_4,X_5 in un dominio finito $\mathcal{X}_i = \mathcal{X}$, $\forall i = 1, \ldots, 5$:

$$f(X_1, X_2, X_3, X_4, X_5) = f_1(X_1, X_3) f_2(X_3, X_4, X_5) f_3(X_2, X_4) f_4(X_2), \tag{3.16}$$

il cui grafo corrispondente è illustrato in Figura 3.3. Il marginale rispetto alla variabile X_2 dopo l'applicazione della GLD è il seguente

$$g_{X_2}(x_2) = \sum_{x_1, x_3, x_4} \left[f_4(x_2) f_1(x_1, x_3) f_3(x_2, x_4) \left(\sum_{x_5} f_2(x_3, x_4, x_5) \right) \right]. \tag{3.17}$$

Nella figura 3.5(a) il processo ha inizio dai nodi foglia f_4, X_1 e X_5 i quali inviano ciascuno il proprio messaggio corrispettivo nodo genitore X_2, f_1 e f_2. I

nodi variabile X_1 e X_4 inviano ai loro nodi fattore genitori (f_1, f_2) la costante 1, mentre il nodo foglia f_4 invia alla suo nodo genitore X_2 il valore del suo fattore $f_4(x_2)$. Il nodo variabile X_2 rimane in attesa finché il messaggio dal suo nodo fattore figlio f_3 è disponibile. Di seguito, il nodo fattore f_1 è pronto per inviare il suo messaggio al suo nodo genitore X_3. Ora seguendo le equazioni (3.14) e (3.15) per i nodi variabile e i nodi fattore rispettivamente, vengono calcolati i messaggi attraverso tutti i nodi del grafo. Il processo termina quando il nodo radice X_2 ha a disposizione tutti i messaggi dai suoi nodi figli f_4 e f_3. Il marginale rispetto alla variabile X_2 è raffigurato in Figura 3.4. I lati attraverso cui è stato inviato il messaggio sono raffigurati di rosso. Ogni messaggio corrispondente ad un diverso step è illustrato con una freccia e un indice associato a quello step.

Questo esempio dettagliato ha mostrato come il calcolo dei messaggi tramite il processo di trasmissione dell'informazione, ha inizio dai nodi foglia e risale gradualmente i lati del factor graph fino a raggiungere il nodo variabile di destinazione. Quando i messaggi di tutti i nodi figli arrivano a un nodo genitore, quest ultimo è in grado di generare il messaggio da inviare al suo corrispettivo nodo padre. Raggiunto il nodo variabile radice di destinazione, attraverso una semplice operazione di moltiplicazione dei messaggi ricevuti viene calcolato il marginale desiderato.

Il processo descritto nell'esempio precedente viene utilizzato per il calcolo della funzione marginale rispetto a una singola variabile. Noi siamo interessati a trovare tutte le funzioni marginali per tutte le variabili della funzione globale. Nella prossima sottosezione verrà descritto l'algoritmo sum-product, attraverso cui è possibile eseguire calcolo simultaneo di tutti i marginali delle variabili della funzione globale.

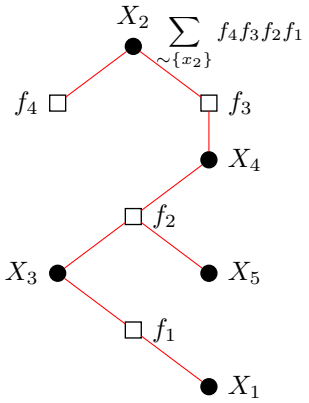


Figura 3.4: In seguito al passaggio dell'informazione attraverso tutti i lati del factor graph (in rosso), il prodotto dei messaggi in ingresso nel nodo variabile X_2 dai nodi figli equivale al marginale relativo alla variabile X_2.

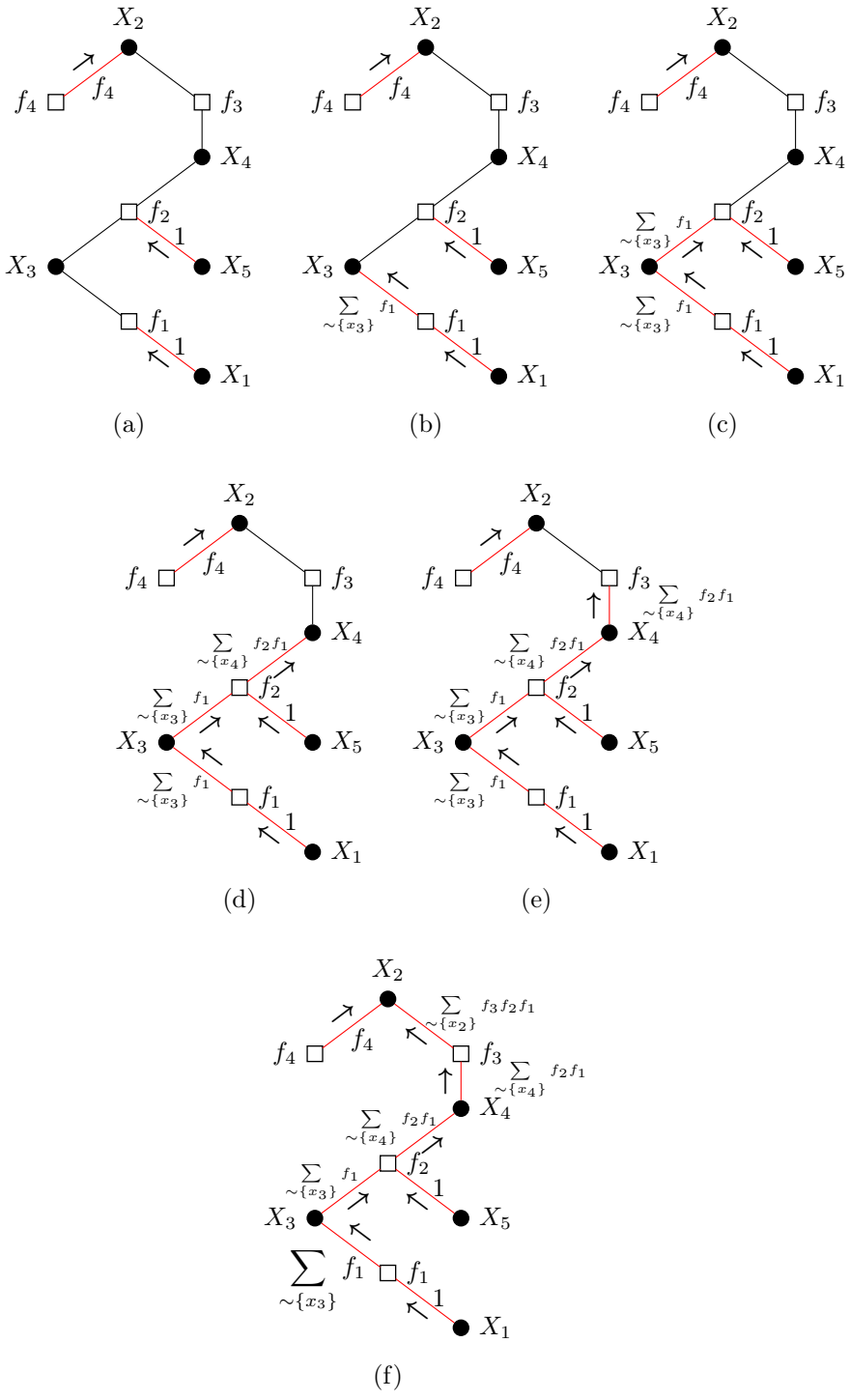


Figura 3.5: Schema del passaggio dell'informazione attraverso il grafo ad albero dell'espressione (3.17). Vengono riportati i messaggi effettivamente trasmessi, in accordo con le relazioni (3.14) e (3.15), accompagnati da una freccia che ne indica la direzione di propagazione. In rosso si evidenziano i lati del grafo attraverso cui è già stato inviato il messaggio da un nodo figlio al suo nodo genitore.

3.2.2 Sum-product algorithm (SPA)

In questa sottosezione verrà illustrato un potente algoritmo per una efficiente computazione dei marginali di una funzione globalmente fattorizzabile, ossia con variabili appartenenti ad un dominio finito. Questo algoritmo è noto come *sum-product algorithm* (SPA).

In molte circostante, come anticipato nella precedente sottosezione, siamo interessati a calcolare il marginale di più di una variabile. Una possibile soluzione è la creazione di un diverso grafo per ciascuna variabile e applicare il single sum-product algorithm. Questo approccio però, sarebbe molto dispendioso da un punto di vista computazionale, e molto lento per un grande numero di variabili. La computazione simultanea di tutti i marginali della funzione può essere completato se sono stati calcolati tutti i messaggi che attraversano i lati del factor graph. Il calcolo del messaggio uscente per SPA segue la stessa idea del single sum-product algorithm. La differenza sostanziale sta nel fatto che nessun nodo è considerato come un nodo radice, quindi non c'è alcuna relazione padre/figlio tra nodi adiacenti. L'algoritmo sum-product termina quando tutti i messaggi sono passati attraverso i lati del factor graph.

Come menzionato in precedenza, ci sono due tipi di nodi, i nodi fattori e i nodi variabile. Supponiamo di avere una variabile X_i in un dominio finito e una funzione locale f_j.[1] Denotiamo $\mu_{X_i \to f_j}(x_i)$ il messaggio da un nodo variabile X_i al suo nodo fattore adiacente f_j, e con $\mu_{f_j \to X_i}(x_i)$ il messaggio dal nodo fattore f_j al nodo variabile X_i. In accordo con l'algoritmo sum-product le update rules dei messaggi hanno la seguente forma:

- **update rule da nodo variabile a funzione locale:**

$$\mu_{X_i \to f_j}(x_i) = \prod_{f_k \in \mathcal{N}(X_i) \setminus \{f_j\}} \mu_{f_k \to X_i}(x_i), \tag{3.18}$$

- **update rule da funzione locale a variabile:**

$$\mu_{f_j \to X_i}(x_i) = \sum_{\sim \{x_i\}} \left(f_j(S_j = s_j) \prod_{X_l \in \mathcal{N}(f_j) \setminus \{X_i\}} \mu_{X_l \to f_j}(x_l) \right), \tag{3.19}$$

dove con S_j indichiamo il set di variabili argomento della funzione locale f_j i.e. $S_j = \{X_l : X_l \in \mathcal{N}(f_j)\}$. Il simbolo "backslash" indica l'espressione "eccetto", letteralmente l'espressione $X_l \in \mathcal{N}(f_j) \setminus \{X_i\}$ significa "tutte le variabili X_l adiacenti al nodo fattore f_j eccetto il nodo variabile X_i". Allo stesso modo, $f_k \in \mathcal{N}(X_i) \setminus \{f_j\}$ sono tutti i nodi fattori adiacenti a X_i eccetto f_j.

L'inizializzazione della procedura è la stessa del single-SPA, che ricordiamo essere per ogni nodo fattore foglia f_j il messaggio ai sui nodi variabile adiacente ($X_m \in \mathcal{N}(f_j)$) $\mu_{f_j \to X_m}(x_m) = f_j(x_m)$, analogamente il messaggio da ogni nodo variabile foglia X_i ai suoi nodi fattori adiacenti ($f_l \in \mathcal{N}(X_i)$) $\mu_{X_i \to f_l}(x_1) = 1$.

Ogni marginale $g_{X_i}(x_i)$ di una variabile X_i è il prodotto di tutti i messaggi entranti nel nodo variabile X_i (se sono tutti disponibili), i.e.

$$g_{X_i}(x_i) = \prod_{f_k \in \mathcal{N}(X_i)} \mu_{f_k \to X_i}(x_i). \tag{3.20}$$

[1] Il termine funzione locale è equivalente al termine fattore.

Similmente, se vogliamo trovare il marginale rispetto a un set di variabili S_j corrispondenti al fattore f_j, dobbiamo moltiplicare per tutti i messaggi entranti nel nodo fattore f_j con il fattore stesso(f_J), i.e.

$$g_{S_j}(s_j) = f_j(s_j) \prod_{X_l \in \mathcal{N}(f_j)} \mu_{X_l \to f_j}(x_l). \tag{3.21}$$

Di seguito vengono riportati i factor graph che illustrano le update rule per entrambi i nodi e il calcolo del marginale per una singola variabile.

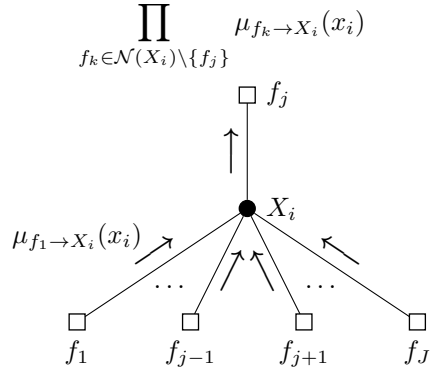


Figura 3.6: Update rule dell'algoritmo sum-product per un nodo variabile. Il nodo variabile ha J nodi adiacenti. Il messaggio in uscita per il nodo fattore f_j equivale al prodotto dei messaggi in ingresso al nodo variabile X_i.

$$\mu_{f_j \to X_i}(x_i) = \sum_{\sim\{x_i\}} \left(f_j(X_1, \ldots, X_i, \ldots, X_I) \prod_{X_l \in \mathcal{N}(f_j)\setminus\{X_i\}} \mu_{X_l \to f_j}(x_l) \right)$$

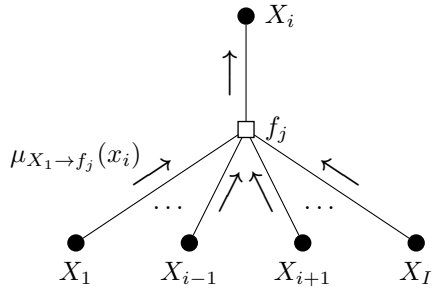


Figura 3.7: Update rule dell'algoritmo sum-product per un nodo fattore. Il nodo fattore ha I nodi adiacenti. Il messaggio uscente dal nodo fattore f_j viene inviato al nodo variabile X_i, quindi la somma non viene effettuata sulla variabile X_i.

Per chiarire bene le regole del SPA e dello schema del passaggio di informazioni verrà di seguito presentato un esempio dettagliato.

Consideriamo la fattorizzazione della funzione dell'esempio precedente:

$$f(X_1, X_2, X_3, X_4, X_5) = f_1(X_1, X_3) f_2(X_3, X_4, X_5) f_3(X_2, X_4) f_4(X_2),$$

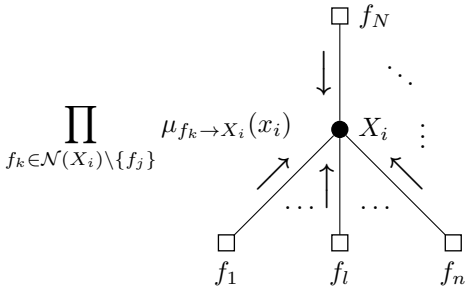


Figura 3.8: Schema della computazione della funzione marginale per un nodo variabile X_i con N nodi fattori adiacenti. Il valore del marginale rispetto alla variabile X_i è equivalente al prodotto di tutti i messaggi uscenti dagli N nodi fattori verso il nodo variabile X_i.

il cui factor graph corrispondente è raffigurato in Figura 3.3. Se ora consideriamo questo grafo come un grafo senza cicli e non ad albero con radici, questo assume la struttura raffigurata nella Figura 3.9.

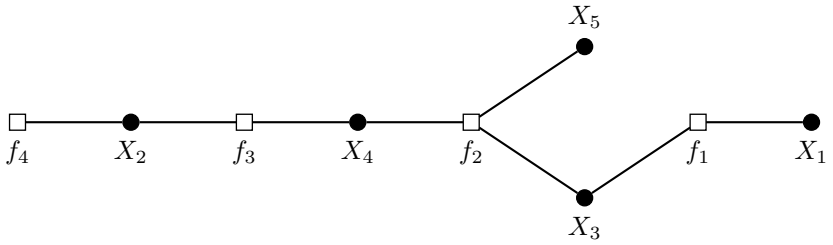


Figura 3.9: Factor graph della funzione globale (3.16). Questo grafo è equivalente a quello in Figura 3.3.

Per semplicità, applicheremo SPA a singoli step, costruendo ogni volta il corrispondente messaggio:

- Step 1:

$$\mu_{f_4 \to X_2}(x_2) = \sum_{\sim\{x_2\}} f_4(x_2) = f_2(x_2),$$
$$\mu_{X_1 \to f_1}(x_1) = 1,$$
$$\mu_{X_5 \to f_2}(x_5) = 1.$$

  I messaggi di questo step sono raffigurato in Figura3.10(a).

- Step 2:

$$\mu_{f_1 \to X_3}(x_3) = \sum_{\sim\{x_3\}} f_1(x_1, x_3)\mu_{X_1 \to f_1}(x_1)$$
$$\mu_{X_2 \to f_3}(x_2) = \mu_{f_4 \to X_2}(x_2).$$

Notiamo che il nodo fattore f_2 non ha ricevuto il messaggio di ingresso dalla nodo variabile X_3 per poter calcolare il messaggio di uscita per i nodi variabile X_5 e X_4. Allo stesso modo, il messaggio in ingresso dal nodo variabile X_4 non è ancora disponibile, quindi, f_2 non può inviare l'informazione ai nodi variabile X_5 e X_3. Per questo motivo il nodo fattore f_2 rimane inattivo. I messaggi di questo step sono mostrati in Figura3.10(b).

- Step 3:

$$\mu_{X_3 \to f_2}(x_3) = \mu_{f_1 \to X_3}(x_3),$$
$$\mu_{f_3 \to X_4}(x_4) = \sum_{\sim\{x_4\}} f_3(x_2, x_4)\mu_{X_2 \to f_3}(x_2).$$

Il nodo fattore f_2 rimane nuovamente inattivo poiché non gli sono ancora giunti tutti i messaggi entranti. La Figura3.10(c) illustra i messaggi di questo step.

- Step 4:

$$\mu_{X_4 \to f_2}(x_4) = \mu_{f_3 \to X_4}(x_4),$$
$$\mu_{f_2 \to X_4}(x_4) = \sum_{\sim\{x_4\}} f_2(x_3, x_4, x_5)\big[\mu_{X_5 \to f_2}(x_5)\mu_{X_3 \to f_2}(x_3)\big].$$

in questo step il nodo fattore f_2 può calcolare il messaggio in uscita per il nodo variabile X_4, poiché ha ricevuto i messaggi in entrata dai nodi variabile X_5, X_3 agli step precedenti. I messaggi in uscita per i nodi variabile X_3 e X_5 non sono ancora pronti pe essere calcolati, quindi devono aspettare un ulteriore step. I messaggi di questo step sono raffigurati in Figura3.10(d).

- Step 5:

$$\mu_{X_4 \to f_3}(x_4) = \mu_{f_2 \to X_4}(x_4),$$
$$\mu_{f_2 \to X_3}(x_3) = \sum_{\sim\{x_3\}} f_2(x_3, x_4, x_5)\big[\mu_{X_5 \to f_2}(x_5)\mu_{X_4 \to f_2}(x_4)\big],$$
$$\mu_{f_2 \to X_5}(x_5) = \sum_{\sim\{x_5\}} f_2(x_3, x_4, x_5)\big[\mu_{X_4 \to f_2}(x_4)\mu_{X_3 \to f_2}(x_3)\big].$$

Tutti i messaggi in uscita dal nodo fattore f_2 sono stati calcolati poiché ha ricevuto tutti i messaggi in ingresso dai nodi variabile adiacenti. I messaggi corrispondenti a questo step sono illustrati in Figura3.10(e).

- Step 6:

$$\mu_{X_3 \to f_1}(x_3) = \mu_{f_2 \to X_3}(x_3),$$
$$\mu_{f_3 \to X_2}(x_2) = \sum_{\sim\{x_2\}} f_3(x_2, x_4)\mu_{X_4 \to f_3}(x_4).$$

I messaggi di questo step sono raffigurati in Figura3.10(f).

- Step 7:

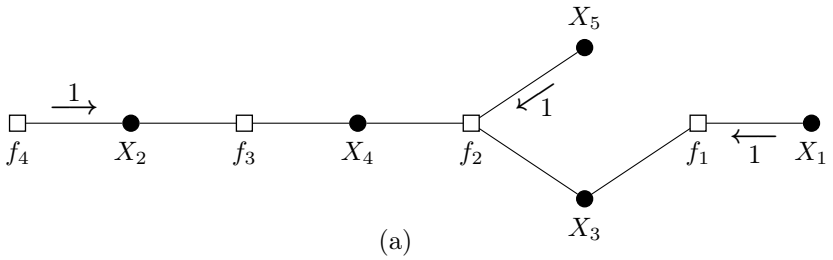
$$\mu_{X_2 \to f_4}(x_2) = \mu_{f_3 \to X_2}(x_2),$$
$$\mu_{f_1 \to X_1}(x_1) = \sum_{\sim\{x_1\}} f_1(x_1, x_3)\mu_{X_3 \to f_1}(x_3).$$

I messaggi dell'ultimo step sono illustrati in Figura3.10(g).

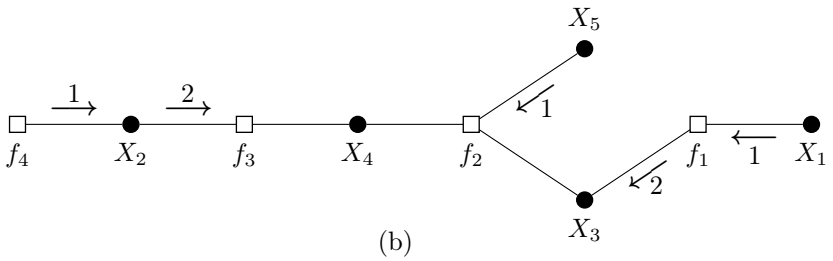
Infine:

$$g_{X_1}(x_1) = \mu_{f_1 \to X_1}(x_1),$$
$$g_{X_2}(x_2) = \mu_{f_4 \to X_2}(x_2)\mu_{f_3 \to X_2}(x_2),$$
$$g_{X_3}(x_3) = \mu_{f_1 \to X_3}(x_3)\mu_{f_2 \to X_3}(x_3),$$
$$g_{X_4}(x_4) = \mu_{f_2 \to X_4}(x_4)\mu_{f_3 \to X_4}(x_4),$$
$$g_{X_5}(x_5) = \mu_{f_2 \to X_5}(x_5).$$

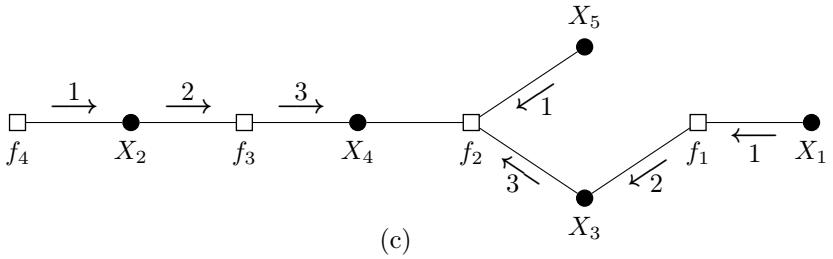
In questo ultimo passaggio calcoliamo contemporaneamente i marginali rispetto a tutte le variabili della funzione f, i quali sono dati dal prodotto di tutti i messaggi in ingresso dai nodi variabile.



(a)



(b)



(c)

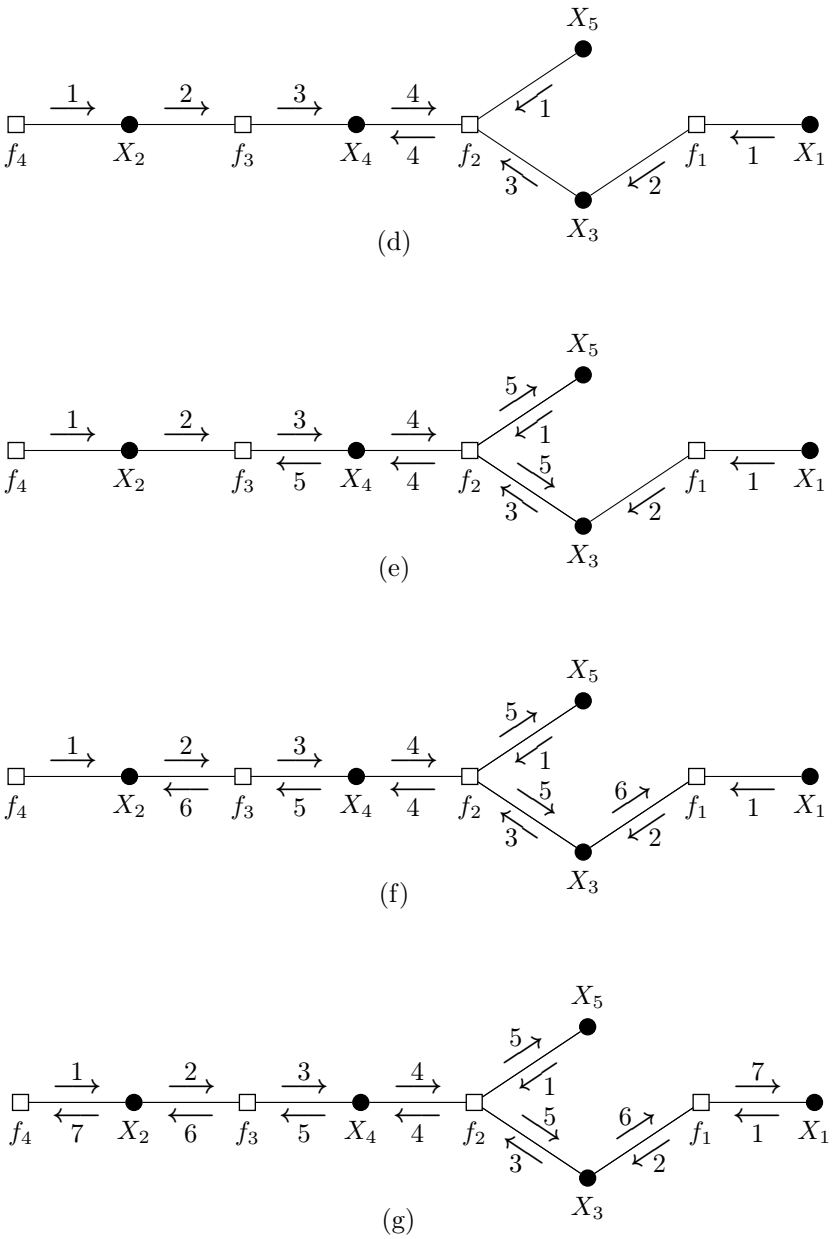


Figura 3.10: Steps dello schema di trasmissione dei messaggi dell'algoritmo SPA. L'indice numerico si riferisce all'ordine in cui questi vengono generati mentre con una freccia viene indicata la direzione di propagazione. Nella sottosezione §(3.2.2) sono riportate le descrizioni dettagliate relative a ciascuno step.

3.3 Algoritmo di decoding

Nelle sezione precedenti è stata definita l'hamiltoniana dei codici LDGM tramite un grafo con numero di connessioni fissato a K ed uniforme.[2] A costo di pedanteria sottolineiamo questo aspetto: la discussione fin qui portata avanti si fonda sull'idea che ottimizzare il codice significa trovare la configurazione di spin $\hat{\boldsymbol{\xi}}$ tale che $\hat{\boldsymbol{\xi}} = \boldsymbol{\xi}$. Come ampiamente descritto nella sezione §3.2.2 la procedura qui implementata è di tipo MPM, ossia la decodifica si concentra sul massimizzare il posterior di ogni singolo bit $\hat{\xi}_i$ e non dell'intero blocco.

3.3.1 Serially concatenated low-density generator-matrix codes (SCLDGM)

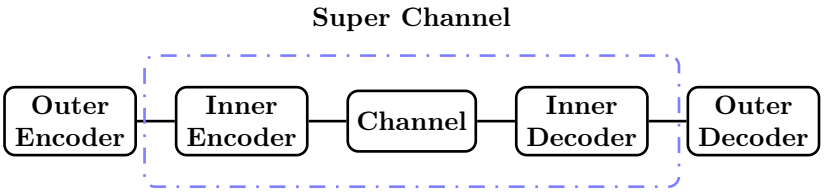
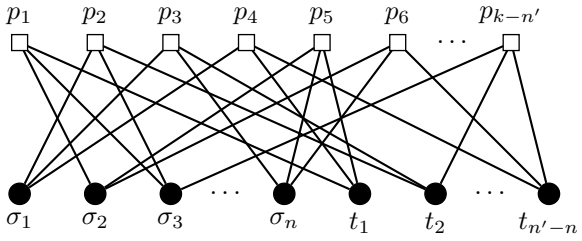


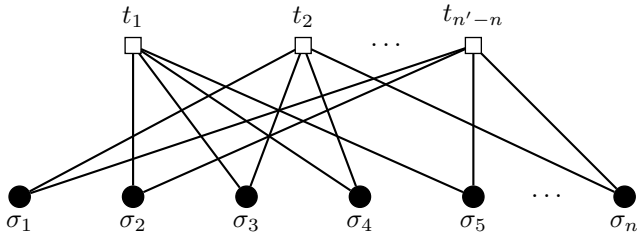
Figura 3.11: Schema della struttura dei codici concatenati SCLDGM. Il messaggio da inviare subisce una prima codifica da un codice esterno e subito una seconda da un codice interno. La sequenza così generata viene inviata all'interno del canale di comunicazione subendo una corruzione dovuta al rumore. Nella fase di decodifica vengono invertite le fasi precedenti, il messaggio subisce una prima decodifica dal codice interno e infine da quello esterno. È possibile riassumere il contributo del codice interno come se il messaggio originale fosse stato inviato in un super-canale con un rumore inferiore rispetto a quello di comunicazione.

Il codice implementato per le simulazioni prende il nome di *serially concatenated low-density generator-matrix codes* (SCLDGM). Mentre i singoli codici LDGM sono asintoticamente "bad", i codici SCLDGM, attraverso l'applicazione dell'algoritmo di Belief Propagation (BP), hanno performance comparabili a quelle dei turbo codici e codici LDPC. La scelta di questi codici, piuttosto che quelli sopra citati, risiede nel fatto che il peso computazionale per l'encoding degli SCLDGM codes è nettamente inferiore. Di fatti i codici LDGM non prevedono l'esecuzione di un algoritmo di eliminazione gaussiana per riscrivere la matrice parity check $\boldsymbol{H}$ in forma sistematica, il quale risulterebbe molto pesante da un punto di vista computazione vista la dimensione di quest'ultima. I codici SCLDGM distinguono due diverse fasi encoding e decoding come mostrato in Figura 3.11. Durante l'encoding, i bit originali vengono prima codificati usando un codice esterno per produrre dei bit intermedi i quali vengono nuovamente codificati da un codice interno per produrre i bit finali di output. Quest'ultimi saranno poi inviati attraverso il canale di comunicazione. Nel processo di decoding, prima viene effettuata la decodifica del codice interno, seguita poi dalla decodifica di quello esterno. Entrambi i codici, interno ed esterno,

[2] É chiaro che, dato un messaggio di ingresso $\boldsymbol{\xi}$ di lunghezza N, la famiglia di grafi qui presa in esame costituisce una sottoclasse di tutti i possibili grafi che è possibile costruire.



(a) Factor graph per la decodifica interna



(b) Factor graph per la decodifica esterna

Figura 3.12: Esempi di grafo bipartito rappresentanti il decoder interno ed esterno.

dei SCLDGM codes utilizzano l'algoritmo belief propagation (BP) come metodo di decodifica. Di seguito presentiamo i passi fondamentali tramite cui ricerchiamo la configurazione $\hat{\boldsymbol{\xi}}$:

- si sceglie una configurazione di spin $\boldsymbol{\sigma}$ generica (ricordiamo $\sigma_i = \pm 1$) lunga n;
- in una prima fase di encoding viene generata una sequenza di bit intermedia $\boldsymbol{m}$, lunga n', tramite un codice esterno ad alto rate ($r_o = n/n' \approx 1$), $\boldsymbol{m} = [\sigma_1, \sigma_2, \ldots, \sigma_n, t_1, \ldots, t_{n'-n}]$;
- nella seconda fase di encoding $\boldsymbol{m}$ viene a sua volta codificato nel messaggio di output $\boldsymbol{o}$, lungo k, tramite un codice interno a rate più basso ($r_i = n'/k$), $\boldsymbol{o} = [\sigma_1, \sigma_2, \ldots, \sigma_n, t_1, \ldots, t_{n'-n}, p_1, \ldots, p_{k-n'}]$;
- si invia il messaggio di output $\boldsymbol{o}$ attraverso un canale rumoroso di tipo BSC con probabilità di errore p, ottenendo il messaggio corrotto $\boldsymbol{h}$;
- si esegue una prima fase di decoding del messaggio corrotto applicando BP al grafo bipartito del decoder interno (Figura 3.12(a)), dove i parity bit interni $\boldsymbol{p} = [p_1, \ldots, p_{k-n'}]$ vengono utilizzati come nodi fattore e i bit intermedi $\boldsymbol{m} = [\sigma_1, \sigma_2, \ldots, \sigma_n, t_1, \ldots, t_{n'-n}]$ come nodi variabile. L'algoritmo viene iterato fino al raggiungimento di una particolare condizione di convergenza (che verrà spiegata successivamente) ottenendo così una configurazione intermedia $\boldsymbol{\sigma}_{\text{out}}$;
- nella seconda fase di decoding, partendo dalla configurazione $\boldsymbol{\sigma}_{\text{out}}$, si applica BP al grafo bipartito del decoder esterno (Figura 3.12(b)), dove i parity bit

esterni $\boldsymbol{t} = [t_1, \ldots, t_{n'-n}]$ vengono utilizzati come nodi fattore e i bit di informazione $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_n]$ come nodi variabile. Si itera l'algoritmo secondo i medesimi criteri della fase precedente, ottenendo la configurazione di spin finale $\boldsymbol{\sigma}_{\text{in}}$.

Riprendendo la notazione utilizzata nella sezione §3.2.2, le update rules dell'algoritmo SP di un LDGM code per l'informazione dal nodo fattore f_j al nodo variabile x_i e dal nodo variabile x_i al nodo fattore f_j, denotate rispettivamente con $\mu_{f_j \to x_i}$ e $\mu_{x_i \to f_j}$, sono:

$$\mu_{f_j \to x_i} = \tanh^{-1}\left[\left(\tanh L(f_j)\right) \prod_{x_l \in \mathcal{N}(f_j)\setminus\{x_i\}} \tanh(\mu_{x_l \to f_j})\right] \tag{3.22}$$

$$\mu_{x_i \to f_j} = L(x_i) + \sum_{f_k \in \mathcal{N}(x_i)\setminus\{f_j\}} \mu_{f_k \to x_i}, \tag{3.23}$$

dove $L(x_i)$ e $L(f_j)$ rappresentano gli estimatori per ciascun corrispondente nodo fattore e nodo variabile. Nel nostro caso, per un canale BSC con probabilità di errore p, gli estimatori assumono i seguenti valori:

- Nodo variabile:
$$L(x_i) = \beta_p \sigma_{\text{in}_i} \quad \text{con} \quad \sigma_{in_i} = h_i; \tag{3.24}$$
- Nodo fattore:
$$L(f_j) = \beta_p J_j; \tag{3.25}$$

dove ricordiamo che β_p è definito dalla relazione (2.3).

Per essere più chiari circa l'implementazione della procedura appena descritta è necessario fare qualche ulteriore precisazione. Al termine delle iterazioni condotte da BP, il valore di ciascun nodo variabile è calcolato attraverso il *marginale* $g_{x_i} = L(x_i) + \sum_{f_k \in \mathcal{N}(x_i)} \mu_{f_k \to x_i}$, il quale somma tutti i messaggi in ingresso dai nodi fattori adiacenti all'estimatore della variabile stessa. **Nel caso del decoder interno, questi stessi valori verranno poi utilizzati come valori iniziali dei messaggi per il decoder esterno.** Dopo aver eseguito la decodifica anche attraverso il codice esterno su un numero predefinito di iterazioni, è possibile prendere la decisione definitiva sul valore da assegnare a ciascun nodo variabile (+1 o −1) utilizzando una *regola di decisione*. È evidente che i valori restituiti dalle espressioni (3.22) appartengono ad un intervallo continuo, mentre abbiamo bisogno di assegnare ad ogni bit del messaggio decodificato un valore discreto; per questo motivo la regola di decisione consiste nel valutare il segno della funzione marginale, $\hat{\xi}_i = \text{sgn}\big(g_{x_i}\big)$:

$$\begin{cases} \hat{\xi}_i = +1 & \text{se } g_{x_i} > 0 \\ \hat{\xi}_i = -1 & \text{se } g_{x_i} < 0 \end{cases}. \tag{3.26}$$

Un'osservazione importante da sottolineare riguarda le condizioni di arresto dell'iterazione dell'algoritmo di BP. Essendo il marginale la funzione che restituisce una stima del segno da assegnare a ciascuna variabile, ed essendo a valori continui, risulta essere il miglior candidato per il controllo sulla convergenza del risultato. Fissando un $\epsilon > 0$ arbitrariamente piccolo, si pone un check su ciascun bit attraverso la condizione $|g_{x_i}^{(f)} - g_{x_i}^{(i)}| < \epsilon$, dove $g_{x_i}^{(f)}$ e $g_{x_i}^{(i)}$ sono i marginali relativi alla

variabile x_i in due iterazioni successive. L'algoritmo SP viene arrestato o se ogni bit del massaggio soddisfa il check per tre iterazioni consecutive o al termine di un numero massimo di iterazioni prefissato.

Parte III

ANALISI PRELIMINARI E STUDIO DELLE PRESTAZIONI DEI CODICI LDGM E SCLDGM

Capitolo 4

Analisi preliminari della simulazione dei codici LDGM

4.1 Definizione del parametro di overlap m

Un primo test preliminare consiste nel verificare le condizioni al contorno della simulazione, ossia stabilire quali valori assegnare a specifici parametri affinché la simulazione restituisca risultati veritieri. L'hamiltoniana (3.1) dipende infatti da quattro parametri (di cui solamente tre sono in realtà indipendenti, a causa della condizione (3.6)) che debbono pertanto essere fissati.

Il teorema di Shannon prevede la possibilità di ottenere una trasmissione priva di errori solo nel limite termodinamico, ossia solo per $N \to \infty$. Chiaramente questa condizione non può essere implementata da un algoritmo, da cui la necessità di fissare N sufficientemente grande da poter essere assunto come limite termodinamico. Infatti è del tutto lecito immaginare l'esistenza di un certo $N_{\text{lim}} > 0$ (il cui modulo dipenderà dall'errore p del canale, dal rate R e dal numero di interazioni K) tale che $\forall N > N_{\text{lim}}$ si possa assumere di lavorare nel limite termodinamico.

Prima di fare questo, dobbiamo definire un funzione che stimi quanto il messaggio di output $\hat{\boldsymbol{\xi}}$ coincida con il messaggio sorgente $\boldsymbol{\xi}$. Quest'ultima è proprio la *funzione di overlap* m che abbiamo definito in (2.12). Per uniformare i risultati di questo elaborato alla letteratura, definiamo m come:

$$m = m(\boldsymbol{\xi}, \hat{\boldsymbol{\xi}}) := \frac{1}{N} \sum_{i=1}^{N} \xi_i \hat{\xi}_i. \tag{4.1}$$

L'overlap m così concepito restituisce un valore compreso tra -1 e 1: tanto più m tende a -1, più $\hat{\boldsymbol{\xi}}$ differisce da $\boldsymbol{\xi}$ (nel limite in cui $m = -1$ i due vettori differiscono in ogni componente), viceversa, tanto più m tende a 1, tanto più $\hat{\boldsymbol{\xi}}$ si può sovrapporre a $\boldsymbol{\xi}$ (nel limite m = 1 i due vettori si sovrappongono in ogni componente).

4.2 Prestazioni della simulazione al variare della lunghezza N del messaggio di ingresso

La prima delle condizioni al contorno che andiamo a studiare è appunto quella relativa alla dimensione del vettore sorgente $\boldsymbol{\xi}$ in quanto è necessario di stimare l'ordine di grandezza di N per assumere che il sistema si trovi nel limite termodinamico.

La Figura 4.1 mostra i risultati della simulazione numerica effettuata con tre diversi valori di N per un grafo con connettività $C = 6$ e con numero di spin a interazione pari a $K = 6$. Prima di analizzare i risultati sottolineamo che ogni istogramma raccoglie i dati di mille simulazioni, dove in ognuna di esse:

- è stato generato un messaggio $\boldsymbol{\xi}$ di N bit;
- è stata generato il grafo rappresentativo dell'hamiltoniana H (3.1);
- si è trovato il vettore $\hat{\boldsymbol{\xi}}$ tramite l'algoritmo di BP;
- si è calcolato il corrispettivo parametro di overlap m tramite la (4.1).

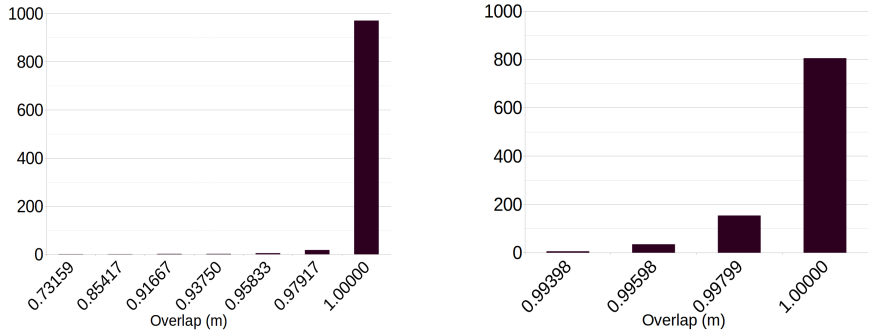
Dai mille valori di overlap ne è stata poi estrapolata la media $\bar{m}$ e la varianza σ^2_m. Andiamo dunque ad analizzare i risultati ottenuti dalla simulazione, riportati di seguito:

N	p	Media $\bar{m}$	Varianza σ^2_m
100	0.05	0.99871	$1.30 \cdot 10^{-4}$
1000	0.05	0.99950	$1.17 \cdot 10^{-6}$
10000	0.05	0.99951	$1.28 \cdot 10^{-7}$

I dati mostrano che il valor medio $\bar{m}$ rimane invariato fino alla terza cifra decimale $\forall N$, mentre per $N = 1000$ e $N = 10000$ non c'è alcuna differenza fino alla quarta cifra decimale. Lo stesso non si può dire della varianza: se N cresce di un fattore 10, la varianza diminuisce circa dello stesso fattore. Questi risultati suggeriscono che l'overlap sia una grandezza *self-averaging*, di cui diamo di seguito la definizione:

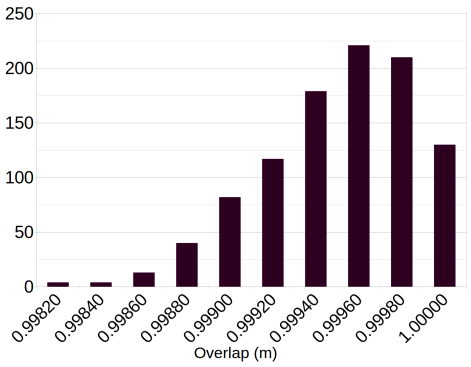
Definizione 4.2.0.1. *Sia X una proprietà fisica di un sistema. Se tale sistema può essere interamente descritto dalla media $\bar{X}$ (intesa come media sui campioni) ed è tale per cui la varianza relativa $R_X = \sigma^2_X/\bar{X} \to 0$ per $N \to \infty$ (dove N è la taglia del sistema e σ^2_X la sua varianza), allora X è una grandezza self-averaging.*

Se chiamiamo $\{\bar{m}_N\}$ la successione dei valori medi di ogni set di dati dipendente da N, ci aspettiamo che $\bar{m}_N \to \bar{m}_\infty$ per $N \to \infty$, dove $\bar{m}_\infty \sim 0.9995$. Questo dimostra che, sebbene per raggiungere il limite termodinamico sarebbe necessario operare con $N \to \infty$, già per $N = 100$ la successione $\{\bar{m}_N\}$ ha saturato la convergenza quasi del tutto. Al contrario, come già detto un aumento di N induce una diminuzione della varianza σ^2_m e non è difficile convincersi del fatto che, definendo in modo analogo una successione $\{\sigma^2_{m_N}\}$, si abbia $\sigma^2_{m_N} \to 0$ per $N \to \infty$. Da



$N = 100$

$N = 1000$



$N = 10000$

Figura 4.1: Confronto tra tre distribuzioni di overlap nei casi in cui $N = 100$, $N = 1000$ e $N = 10000$. I valori medi corrispondono rispettivamente a 0.99871, 0.99950 e 0.99951: questo giustifica come la convergenza sia quasi totalmente saturata già per $N = 100$, sebbene si dovrebbe avere con $N \to \infty$ (parametri: $C = 6$, $K = 6$ e $p = 0.05$).

ciò consegue che la varianza relativa $R_{m_N} = \sigma^2_{m_N}/\bar{m}_N \to 0$ per $N \to \infty$: allora l'overlap è una grandezza self-averaging.

Dal momento che a noi interessa il solo valore medio $\bar{m}$ del set di 1000 dati, sarebbe sufficiente operare con $N = 100$ per avere una buona certezza che il sistema si trovi nel limite termodinamico. Tuttavia, richiedendo anche un controllo sull'errore commesso (fornito dalla varianza σ^2_m), è più opportuno scegliere come valore di lavoro $N = 1000$ o (ancora meglio) $N = 10000$.

Un'ultima osservazione: come appena dimostrato, la scelta di $N = 1000$ o $N = 10000$ non influisce in alcun modo sul valor medio della distribuzione di overlap, quindi (a meno di un miglioramento sul controllo dell'errore) scegliere N di ordine maggiore non ha apparentemente alcun vantaggio. In realtà il problema è più sottile. In Figura 4.1 gli istogrammi con $N = 100$ e $N = 1000$ mostrano un singolo picco molto alto sulla destra: questo comportamento è dovuto al fatto che, in più casi, l'algoritmo è riuscito a correggere perfettamente il messaggio. Poiché al crescere di N questo fenomeno non si ripresenta, si può concludere che sia $N = 100$ sia $N = 1000$ non siano una buona approssimazione del limite termodinamico. Scegliendo valori dei parametri K e C maggiori, il comportamento assunto in questo

caso da $N = 1000$ può essere maggiormente marcato. Tutto questo per dire che la scelta del parametro N dipende dai valori di K e C e pertanto dovrebbe essere scelto ad hoc di volta in volta. Possiamo però affermare che in generale il limite termodinamico viene raggiunto per $N = \mathcal{O}(10000)$ nel range di valori di K qui analizzati, e se ne darà una dimostrazione più dettagliata in una sezione successiva.

4.3 Parametrizzazione delle prestazioni della simulazioni al variare della temperatura β

Una seconda condizione al contorno di vitale importanza è anche quella degli estimatori all'interno dell'algoritmo BP. Di fatto, senza quella informazione, l'iterazione non potrebbe giungere a convergenza della soluzione ottimale $\hat{\boldsymbol{\xi}} = \boldsymbol{\xi}$. Il messaggio da inviare attraverso il canale $\boldsymbol{\xi}$ è ovviamente non noto (altrimenti avrebbe poco senso costruire una procedura che permetta di ritrovarlo), bensì sono noti $J_{i_1,\dots,i_K}$ e $\boldsymbol{h}$. Ricordando le loro definizioni in (3.2) e (3.3), ci si accorge che differiscono dal vettore $\boldsymbol{\xi}$ e dai suoi parity check bit per un numero di bit pari a $p \cdot N$ e $p \cdot M$ rispettivamente, nel limite in cui N e M siano sufficientemente grandi (ricordiamo che p è la probabilità di errore commessa sul singolo bit causata dal rumore del canale). In genere p è un numero piccolo, nell'ordine di 0.01 - 0.1, quindi appare evidente come questi possano essere dei buoni "indicatori" per la ricostruzione del messaggio originale. Questo non è tuttavia sufficiente, in quanto ugualmente importante è il parametro moltiplicativo β. Seguendo lo stesso procedimento descritto nella sezione precedente, in figura 4.2 presentiamo le parametrizzazioni in β delle prestazioni di due codici del tipo (C, K)[1] a $p = 0.05$ fissato.

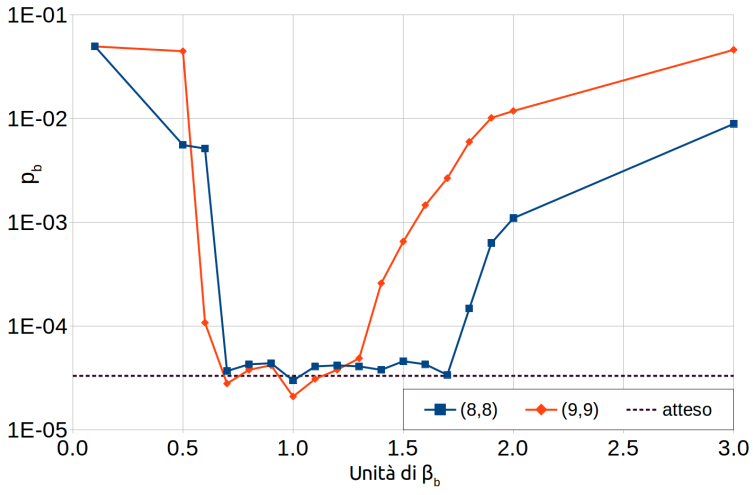


Figura 4.2: Parametrizzazione delle prestazioni dei codici $(8, 8)$ e $(9, 9)$ in funzione di β a $p = 0.05$ nell'intervallo $[0, 3\beta_p]$. Ogni punto è dato dalla media su 1000 simulazioni. L'andamento mostra come per valori che si discostano da β_p l'algoritmo di BP fallisca a convergere alla soluzione prevista dalla previsione teorica.

[1] Con questa notazione indichiamo le proprietà del factor graph coerentemente con quanto descritto nella sezione §3.1.2.

È abbastanza evidente come le prestazioni dell'algoritmo BP falliscano nel convergere alla soluzione prevista dalla previsione teorica (la cui stima verrà illustrata nelle sezioni successive) per valori che si discostino troppo da β_p. Questo risultato è fortemente in accordo con la teoria di meccanica statistica espressa nella sottosezione §2.2.2, in cui si dimostra come per una procedura del tipo MPM, valutare $\beta = \beta_p$ costituisca una massimizzazione dell'overlap. Ricordiamo infatti che il nostro problema è stato modellizzato a partire dalla teoria di Ising di vetri di spin [20]. Da un punto di vista fisico ci troviamo in uno stato a temperatura finita $T_p = \beta_p^{-1}$ e gli estimatori, così come espressi in (3.24) e (3.25), non sono altro che le magnetizzazioni locali del nostro sistema.

Capitolo 5

Prestazioni dei codici LDGM e SCLDGM

Fino a questo momento abbiamo fornito un'analisi delle condizioni al contorno della simulazione dei codici LDGM, ossia lo studio dei parametri da cui dipende l'hamiltoniana (3.1) di tali codici. Ci sono però alcune domande fondamentali a cui si vorrebbe dare risposta. Supponendo di fissare le condizioni al contorno (N, K, ecc.), le prime domande che sorgono spontanee sono per esempio: esiste un grafo privilegiato, tale per cui le sue prestazioni siano migliori rispetto a quelle di qualunque altro grafo? Esiste un K ottimale oppure la connettività infinita è l'unica che soddisfa il limite di Shannon? Un'idea delle risposte a tali domande sarà l'oggetto principale di questo capitolo.

5.1 Dipendenza delle prestazioni dei codici LDGM dal numero di scambi di edge

In un elaborato precedente [26] a questo lavoro è già stato dimostrato come un factor graph con una geometria di tipo *random*, ossia generato in modo casuale, costituisca una soluzione ottimale del problema. Vogliamo qui rimarcare questo aspetto, ponendo l'attenzione sul fatto che per i codici da noi implementati, essa costituisca uno degli aspetti fondamentali per il corretto funzionamento dell'algoritmo di BP.

Ricordando la procedura mostrata nella sottosezione §3.1.3 per ottenere un particolare tipo di factor graph totalmente casuale, ogni n° permutazioni di una coppia di lati calcoliamo il corrispettivo errore commesso sul singolo bit p_b. In questo modo, a partire da un particolare factor graph, possiamo osservare come varia p_b in funzione del numero di scambi effettuati. Avendo chiamato n° tale variabile è ben definita la funzione $p_b(n^\circ)$. È ragionevole aspettarsi che per un numero di scambi sufficientemente grande p_b raggiunga una condizione di equilibrio, essendo una grandezza self averaging.[1] Tuttavia, è meno prevedibile il comportamento prima della stabilizzazione intorno al proprio valore medio. Sarebbe infatti possibile sia un andamento oscillante sia un decrescita monotona.

[1] Per la giustificazione di questa affermazione rimandiamo alla sezione §5.2

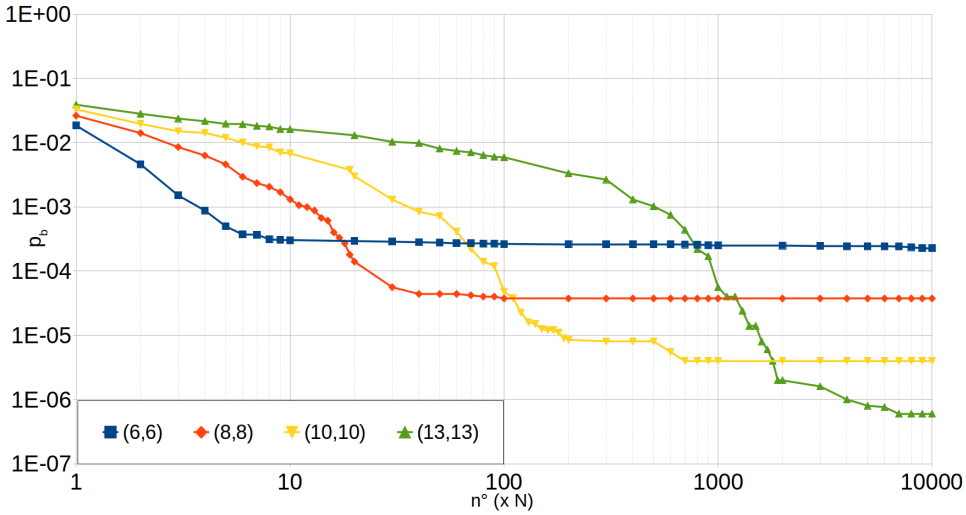


Figura 5.1: Mappatura dell'errore p_b in funzione del numero di scambi n° effettuati, per messaggi di dimensione $N = 10000$ al variare di K. Il grafo di partenza (che corrisponde al valore $n^\circ = 0$) è quello descritto nella sottosezione §3.1.2: dopo un numero scambi opportunamente fissato di una coppia di lati, è stato calcolato il corrispettivo errore per bit (media di un set di mille dati). Sono state mappate quattro curve a rate fissato $R = 1/2$. Si evince come il processo di termalizzazione dipenda fortemente da K, aumentando la dimensione dell'interazione aumenta il numero di scambi n° necessari per raggiungere la condizione di equilibrio.

La Figura 5.1 mostra la mappatura dell'errore p_b in funzione del numero di scambi n° effettuati, per messaggi di dimensione $N = 10000$ con codici a rate fissato $R = 1/2$. Nello specifico, fissato il factor graph di partenza descritto nella sottosezione §3.1.2, dopo ogni n° permutazioni di una coppia di lati è stato calcolato il corrispettivo valore dell'errore p_b (ottenuto come media di un set di mille dati). Mostriamo di seguito i risultati fondamentali:

- Ogni curva mostra un andamento monotono decrescente indipendentemente dal valore di K. Non riscontriamo minimo locali prima della fase di *termalizzazione*, ossia di convergenza di p_b a una condizione di equilibrio.
- Al crescere di n°, ogni curva converge ad un valore costante p_K che dipende da K. Maggiore è K minore è il valore di p_K. Una domanda interessante cui rispondere è se $p_K \to 0$ per $K \to \infty$ oppure se $p_K > 0$ definitivamente.
- Il processo di termalizzazione dipende da K: aumentando la dimensione dell'interazione aumenta il numero di scambi n° necessari per la termalizzazione dell'errore. In particolare al crescere di K può variare anche l'ordine di grandezza degli scambi necessari da effettuare.

L'andamento decrescente dell'errore commesso sul singolo bit p_b è un indice dell'esistenza di geometrie svantaggiose all'interno del grafo rispetto alle altre. Riflettendo in merito alla procedura utilizzata per generare un grafo random, è possibile che a numero di scambi fissato, per valori di K crescenti, questi non siano più in

grado di disperdere il grado di ordine con cui il grafo è stato originariamente creato. Possiamo immaginare, quindi, che all'interno del processo di scambio dell'informazione dell'algoritmo di BP, esistano uno o più sottoinsiemi di nodi variabile (con i corrispettivi nodi fattore) quasi totalmente chiusi in se stessi. Quest'ultimi continuerebbero ad inviare vicendevolmente la medesima informazione, contribuendo solo in minima parte alla decodifica del resto della sequenza. O ancora, ad ogni iterazione, potrebbero avvalorare una decodifica incorretta dei nodi variabile appartenenti a quel sottoinsieme, inficiando la valutazione dei marginali di altri nodi variabile.

Risulta dunque più che necessario allontanarsi il più possibile dalla configurazione ordinata iniziale se non si vogliono compromettere le prestazioni dei codici. Una questione sottile sarebbe quella di stabilire quale sia, fissate le condizioni al contorno N, C e K, il numero di scambi minimo n° tale per cui si è certi di aver raggiunto una configurazione sufficientemente random del grafo. Per l'ordine di grandezza delle interazioni considerate in questo elaborato, possiamo essere sicuri che $n^\circ \sim \mathcal{O}(N^2)$ costituisca una buona stima per raggiungere la fase di termalizzazione dell'errore.

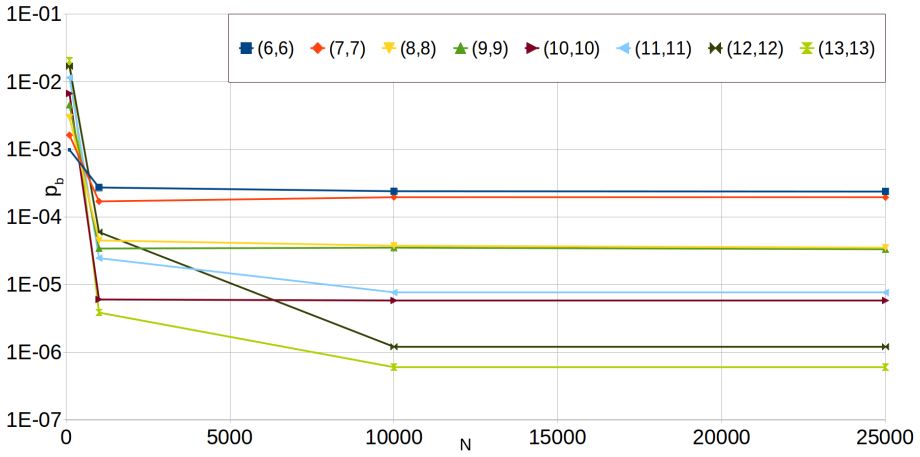
5.2 Dipendenza della termalizzazione dell'errore per bit p_b dalla taglia del sistema

Abbiamo già osservato nella sottosezione §4.2 come il limite termodinamico venga in genere raggiunto per $N = \mathcal{O}(10000)$ (almeno per il numero di corpi K qui preso in considerazione). È stata inoltre data una dimostrazione qualitativa del fatto che l'overlap m sia una grandezza self-averaging.

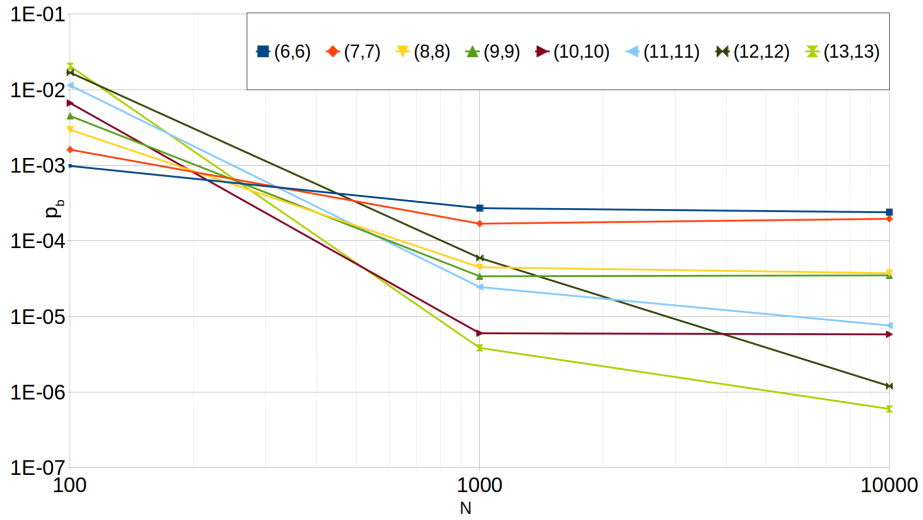
Qui di seguito forniremo ulteriori prove di questa proprietà dell'overlap. Ricordando che la probabilità di errore per ogni bit dopo la codifica del messaggio di output è definita come

$$p_b = 1 - \frac{1+m}{2}, \tag{5.1}$$

si evince una corrispondenza biunivoca tra m e p_b, di conseguenza anche quest'ultima è una grandezza self-averaging. Ci aspettiamo dunque che mappando p_b in funzione di N si raggiunga una fase di termalizzazione. Come si può osservare nella figura 5.2, la velocità di termalizzazione dipende fortemente dal numero K dei corpi interagenti: tanto più cresce K maggiore deve essere la taglia minima N del sistema per cui si possa assumere di lavorare nel limite termodinamico. Prima che il sistema raggiunga una condizione di equilibrio, il comportamento della curva $p_b(N)$ evidenzia delle prestazioni nettamente inferiori.



(a)



(b)

Figura 5.2: Mappatura del parametro p_b (errore commesso sul singolo bit in seguito alla codifica del messaggio di output del canale) in funzione della dimensione N del sistema, eseguita al variare di K per curve del tipo (K, K) per $p = 0.05$. Ogni curva mostra un processo di termalizzazione più o meno rapido a seconda del K considerato: tanto più cresce K, tanto maggiore è la dimensione minima che il sistema deve avere affinché si possa assumere che questo si trovi nel limite termodinamico. La seconda figura è una macro della prima, con una mappatura eseguita sull'intervallo $[10^2, 10^4]$.

Per i risultati ottenuti, risulta dunque ampiamente giustificata quindi la scelta di utilizzare $N = 10000$ come grandezza di riferimento per le simulazioni successive. Tuttavia, per completezza sottolineiamo questo aspetto: la stima dell'errore commesso sulla misura di m nella sottosezione 4.2 è stata valutata tramite la varianza della distribuzione gaussiana dei mille valori calcolati. Nello specifico, quest'ultimi

sono tutti stati ottenuti a partire da uno specifico grafo casuale. La varianza, dunque, tiene conto soltanto dell'errore commesso sul singolo grafo. Potrebbe infatti subire delle fluttuazioni a seconda del grafo su cui viene calcolato m e, pertanto, in generale costituisce una sottostima del reale errore cui è soggetta una misura come la nostra. Ciò nonostante, avendo precedentemente verificato che il grafo random è subottimale, da qui in avanti verrà trascurato questo aspetto e si assumerà la varianza come una buona stima dell'errore commesso.

Ricordiamo che la varianza è la misura di quanto i dati si discostano quadraticamente rispetto al valor atteso (che nel caso di una distribuzione gaussiana coincide con la media aritmetica). L'errore di quest'ultimo è invece dato dalla *deviazione standard della media* e si ricava a partire dalla varianza come

$$\sigma_{\bar{X}} = \sqrt{\frac{\sigma_X^2}{N}}. \tag{5.2}$$

5.3 Prestazioni dei codici LDGM

Nella prima sezione di questo capitolo di siamo occupati di studiare approfonditamente i parametri di accettabilità dei grafi $\mathcal{G}$ qui presi in esame per l'algoritmo di BP. A questo punto dobbiamo verificare le vere e proprie prestazioni dei codici implementati. Di seguito verranno mostrati i risultati delle simulazioni messi a confronto sia con le previsioni teoriche sia con le prestazioni di altri codici della letteratura.

5.3.1 Mappatura dell'errore per bit p_b al variare della dimensione dell'interazione K

Come già anticipato nella sezione precedente, l'overlap m e l'errore p_b sono in corrispondenza biunivoca secondo l'equazione (5.1). Di conseguenza lo studio di p_b permette di determinare le prestazioni del codice al pari dello studio di m: scegliamo il primo per uniformità alla letteratura preesistente. In figura 5.3, 5.4 e 5.5 vengono riportati i risultati delle simulazioni in cui sono state mappate le curve p_b in funzione dell'errore p introdotto sul singolo bit dal canale BSC, al variare del parametro K per rate R fissati. In particolare sono stati presi in esame tre valori di R: 1/2, 1/3 e 1/4. Esattamente come nelle simulazioni preliminari, ogni valore di p_b nel grafo è stato calcolato come media di un set di mille dati. Come detto nella sezione precedente, l'errore sulla singola simulazione è stato valutato attraverso la deviazione standard della media. Nei grafici però le barre d'errore non sono visibili in quanto risultano più piccole della dimensione del punto. Le uniche barre di errore visibili sarebbero relative alle simulazioni fallimentari del codice, dunque non vengono riportate in quanto i dati sono poco significativi.

All'interno dei grafici, per poter effettuare una migliore valutazione sui risultati ottenuti, accanto alle simulazioni vengono riportate anche le curve attese dalla previsione teorica. Specificamente, consideriamo un singolo nodo variabile collegato a C nodi fattore, e un BSC con parametro p. Supponendo che tutti i messaggi da altri nodi variabile in questi C nodi fattore siano corretti e che siano dispari, il metodo di decodifica fallirà e produrrà la stima errata per il nodo variabile se $(C+1)/2$ o più nodi fattore sono in errore. Per il caso in cui è pari, possiamo considerare che il decoder produrrà una stima errata del nodo variabile quando

più di $C/2$ nodi fattore sono in errore o quando $C/2$ nodi fattore più il nodo variabile considerato sono errati. Quindi, considerando una distribuzione binomiale con probabilità p, la probabilità di errore può essere approssimata come:

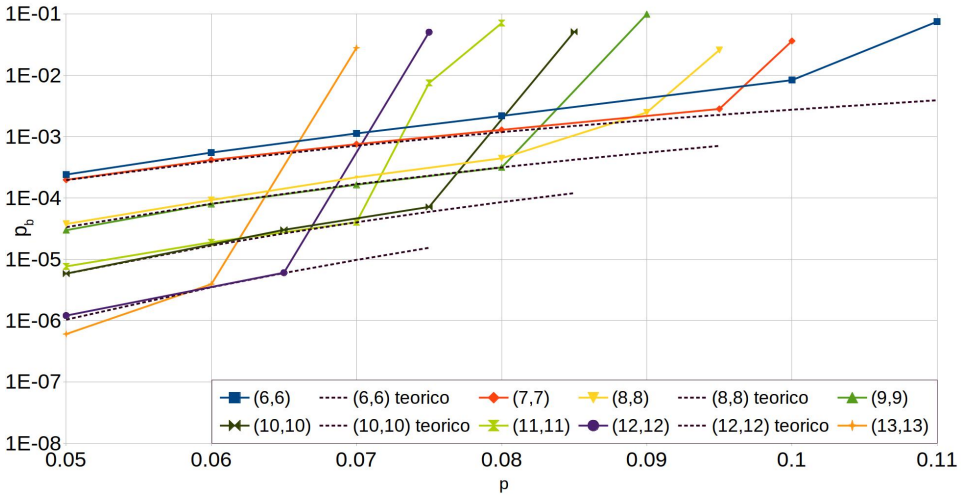
- Per C dispari:

$$p_b \approx \sum_{i=(C+1)/2}^{C} \binom{C}{i} p^i (1-p)^{C-i}. \tag{5.3}$$

- Per C pari:

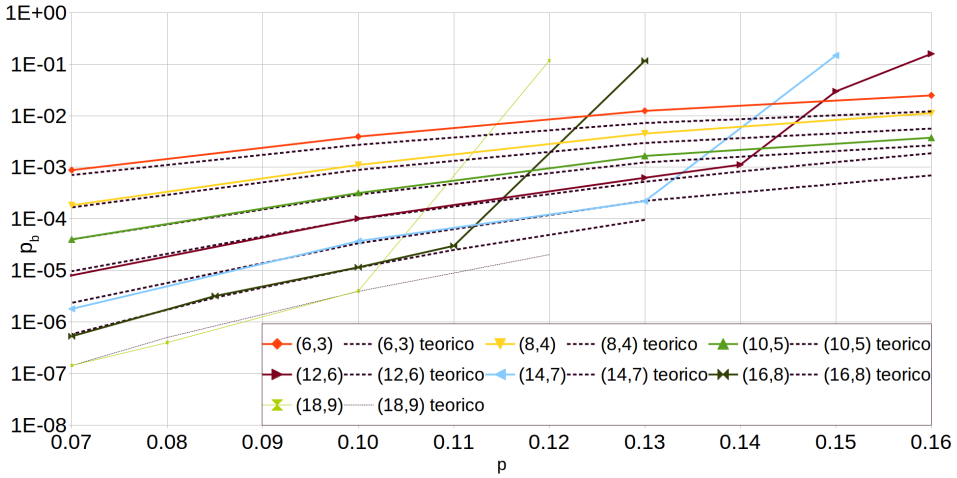
$$p_b \approx p\binom{C}{\frac{C}{2}} p^{\frac{C}{2}} (1-p)^{\frac{C}{2}} + \sum_{i=\frac{C}{2}+1}^{C} \binom{C}{i} p^i (1-p)^{C-i}. \tag{5.4}$$

Nel caso di codici del tipo (C, K) e $(C+1, K)$ con C pari la previsione teorica restituisce i medesimi risultati.



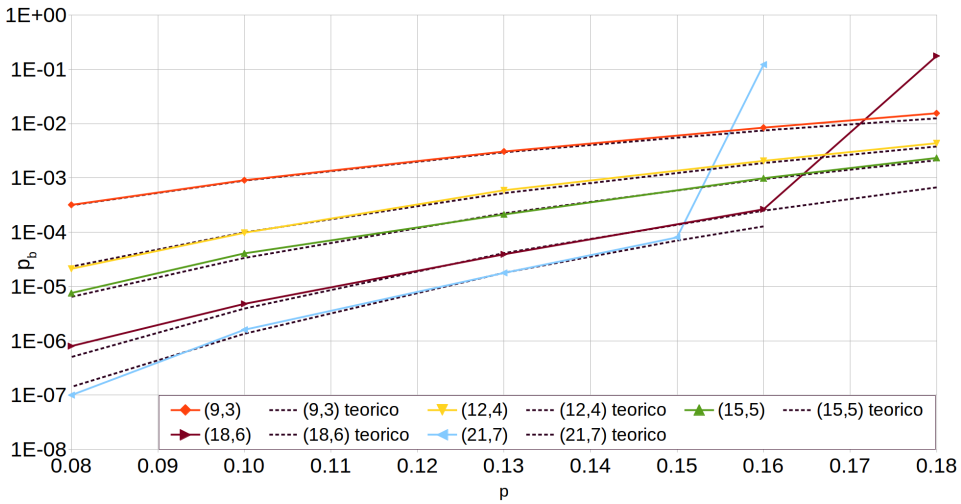
(a) $R = \frac{1}{2}$

Figura 5.3: Il grafico mostra l'andamento del parametro p_b in funzione dell'errore p introdotto dal canale BSC a rate fissato $R = \frac{1}{2}$. Vengono mappate le curve (C, K) del tipo (K, K) al variare di $K = 6, \ldots, 13$.



$(b) \quad R = \frac{1}{3}$

Figura 5.4: Il grafico mostra l'andamento del parametro p_b in funzione dell'errore p a rate fissato $R = \frac{1}{3}$. Vengono mappate le curve (C, K) del tipo $(2K, K)$ al variare di $K = 3, \ldots, 9$.



$(c) \quad R = \frac{1}{4}$

Figura 5.5: Il grafico mostra l'andamento del parametro p_b in funzione dell'errore p a rate fissato $R = \frac{1}{4}$. Vengono mappate le curve (C, K) del tipo $(3K, K)$ al variare di $K = 3, \ldots, 7$.

Il comportamento dei codici LDGM è eterogeneo e dipende fortemente dal numero K di corpi che partecipano all'interazione (da qui in avanti indicheremo con $p_b^{(K)}$ la curva p_b dipendente dal parametro K). Analizziamo di seguito i risultati ottenuti:

- Nel caso di $R = 1/2$ abbiamo ottenuto che le prestazioni dei codici da noi implementati sono in grado di replicare perfettamente le prestazioni dei codici di J Garcia-Frias e W. Zhong [7]. Le curve $p_b^{(K)}$ sono, inoltre, sostanzialmente in accordo con la previsione teorica espressa dalle (5.3) e (5.4). Se nel caso di K dispari[2] le due curve si sovrappongono precisamente, per piccoli valori di K pari, le simulazioni tendono a discostarsi leggermente dalla previsione teorica all'aumentare di p.

- Ogni codice presenta un $p_c^{(K)}$ critico (anch'esso fortemente dipendente da K) oltre il quale non è più in grado di convergere alla soluzione ottimale e rispettare la previsione teorica. Se per $p < p_c^{(K)}$ rispecchiano l'andamento descritto al punto precedente, per $p > p_c^{(K)}$ le simulazioni restituiscono un $p_b \sim p$, come se non fosse applicata quasi alcuna correzione al messaggio in ingresso. Con molta probabilità il sistema sta uscendo dal limite termodinamico, dunque sarebbe necessario aumentare la taglia del sistema per permettere all'algoritmo di BP di giungere a convergenza.

- Per i grafici relativi a $R = 1/3$ e $R = 1/4$ i risultati sono sostanzialmente analoghi a quelli per $R = 1/2$. Nel primo caso, essendo i codici nella forma $(C = 2K, K)$, si ripresenta sistematicamente il comportamento descritto per K pari nel caso $R = 1/2$. Per $R = 1/4$ le curve sembrano invece aderire alla previsione teorica indistintamente sia per C pari che dispari. È evidente anche come al diminuire del rate R, nonostante il notevole aumento del valore dei parametri C e K l'errore critico $p_c^{(K)}$ tende anch'esso ad aumentare.

5.4 Prestazioni dei codici SCLGDM

Presi in esame tutti gli aspetti che concernono alle simulazioni dei singoli codici LDGM, ci apprestiamo ora ad applicare la tecnica della concatenazione come descritta nella sezione §3.3 e presentarne le relative prestazioni.

5.4.1 Parametrizzazione delle prestazioni della concatenazione al variare della temperatura β

Analogamente ai singoli codici LDGM, anche per i codici SCLDGM sorge la problematica di stabilire quali siano i migliori estimatori da inserire nelle equazioni di ricorsione dell'algoritmo di belief propagation per il codice esterno. I ragionamenti fatti nella sezione §4.3 in merito alla scelta di $J_{i_1,\dots,i_K}$ e $\boldsymbol{h}$ rimangono tutt'ora validi, più sottile è invece la questione relativa alla temperatura β. A priori non c'è modo di stabilire se, in seguito al primo processo di decodifica, il valore di β ottimale sia quello relativo all'errore p introdotto dal canale BCS o l'errore p_t restituito dalla

[2] In questo caso, essendo i codici nella forma $(C = K, K)$, possiamo utilizzare indistintamente i due parametri esseno totalmente analoghi.

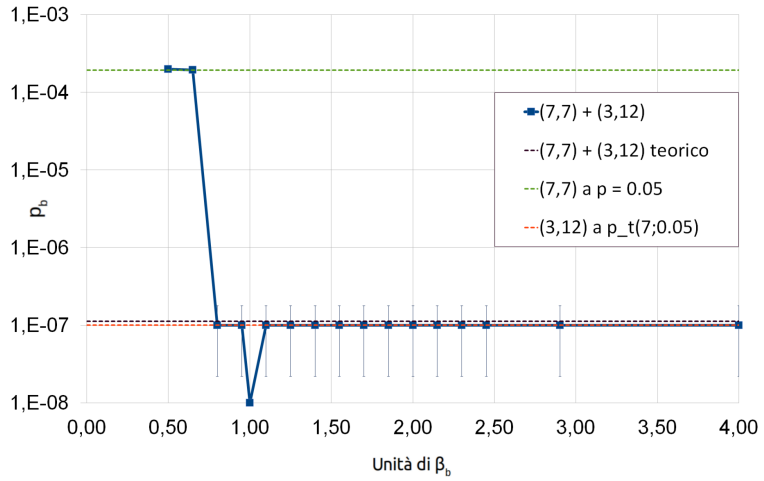


Figura 5.6: Parametrizzazione delle prestazioni della concatenazione di un codice $(3, 12)$ ad un codice $(7, 7)$ a $p = 0.05$ in funzione di β nell'intervallo $[0, 4\beta_p]$. Ogni punto è dato dalla media su 2000 simulazioni. Vengono riportati a confronto i risultati delle simulazioni del codice $(7, 7)$ a $p = 0.05$ (linea tratteggiata verde) e i risultati assieme con la previsione teorica del codice $(3, 12)$ a $p = p_t^{(7;0.05)}$ (rispettivamente linea tratteggiata rossa e nera).

previsione teorica per il codice LDGM interno. In figura 5.6 presentiamo dunque le parametrizzazioni della prestazione di un codice SCLDGM in funzione del valore di β inserito all'interno del codice esterno. Vengono posti a confronto i risultati delle simulazioni del codice $(7, 7)$ a $p = 0.05$ e i risultati assieme con la previsione teorica del codice $(3, 12)$ a $p = p_t^{(7;0.05)}$. [3]

Analizziamo di seguito i risultati ottenuti:

- Al pari dei singoli codici LDGM, le simulazioni evidenziano come le prestazioni vengano ottimizzate per $\beta = \beta_p$. Diverso è invece il comportamento per valori di β che si discostano da questo valore. Nel caso di $\beta > \beta_p$ le prestazioni del codice tendono a convergere a quelle ottenute senza la tecnica della concatenazione, in quest'ottica esse costituiscono dunque una soluzione subottimale. Nel caso $\beta < \beta_p$ l'algoritmo non è in grado di compiere alcuna ulteriore correzione sul messaggio uscente dal codice interno.

- Concentrando l'attenzione sul risultato ottimale, si può notare come le prestazioni in questo punto superino di gran lunga le previsioni teoriche. Specifichiamo che questa stima è stata ottenuta inserendo l'errore teorico $p_t^{(7;0.05)}$ restituito dalla relazione (5.3) per il codice $(7, 7)$ a $p = 0.05$, nuovamente all'interno della (5.3) per il codice $(3, 12)$. Una stima di questo tipo è però associabile solo ad una simulazione condotta sul singolo codice $(3, 12)$ con errore iniziale $p_t^{(7;0.05)}$, come si può evincere chiaramente dai risultati ottenuti. Con ciò vogliamo evidenziare il fatto che la tecnica della concatenazione co-

[3] Identifichiamo con la notazione $p_t^{(C;p)}$ la stima teorica dell'errore data dalle (5.3) e (5.4), per un codice LDGM a connettività C ed errore introdotto dal canale BSC p.

stituisca uno strumento più avanzato della semplice esecuzione di due codici LDGM svincolati che rispettino gli stessi parametri di ingresso. La chiave di volta risiede proprio nell'informazione trasmessa dal codice interno a quello esterno attraverso un primo calcolo dei marginali.

Alla luce di questi risultati risultano chiare le potenzialità dei codici SCLDGM. Partendo da una configurazione iniziale in cui ciascun bit del messaggio inviato attraverso il nostro BSC ha una probabilità $p = 0.05$ di essere stato modificato, applicando il codice LDGM interno $(7, 7)$ di rate $R_i = 1/2$, come espresso dalla (3.9), otteniamo una probabilità di errore per bit $p_b^{(7)} \sim \mathcal{O}(10^{-4})$. Concatenando ora questo codice al codice esterno $(3, 12)$ di rate $R_o = 4/5$ otteniamo un errore $p_b \sim \mathcal{O}(10^{-8})$. Al netto di una diminuzione del rate da $R = 1/2$ a $R = R_i\, R_o = 2/5$, siamo riusciti ad abbassare l'errore commesso sul messaggio in ingresso di quattro ordini di grandezza. Si potrebbe pensare di non fermarsi ad una sola concatenazione, ma compiere più concatenazioni successive riducendo l'errore di altri ordini di grandezza. Questa possibilità svincolerebbe dalla necessità di aumentare enormemente il grado di interazione K del sistema per raggiungere una trasmissione quanto più possibile error-less, il quale andrebbe ad influire sui parametri minimi da rispettare per la simulazione, come la taglia del sistema N e il numero di scambi di edge n°. Lo studio di queste questioni non sarà però oggetto di questa tesi.

Conclusioni

Risultati

Con questo elaborato ci siamo posti l'obiettivo di approfondire alcune proprietà di una particolare classe di codici di Sourlas a connettività finita, i LDGM codes. Abbiamo successivamente rivolto lo sguardo ai SCLDGM code, dal momento che questi ultimi sembrano costituire un valido sostituto ai codici LDPC, in quanto a parità di prestazioni il loro peso computazionale è notevolmente inferiore. Qui è stata fatta un'analisi preliminare in vista delle simulazioni vere e proprie delle prestazioni dei codici LDGM. Nello specifico, ci siamo occupati delle seguenti tre problematiche principali.

(i) Come concretamente costruire un factor graph che sia l'equivalente di un'hamiltoniana di interazione a K corpi: abbiamo fornito un'idea di come costruire un factor graph generico, che consiste sostanzialmente nella randomizzazione dei lati di uno specifico grafo facilmente costruibile, la cui struttura è univocamente determinata dai parametri N, M, K e C in accordo con le condizioni (3.6) e (3.7) (cfr. sottosezione §3.1.3).

(ii) Prestazioni della simulazione al variare della dimensione N del sistema: dopo aver fornito una dimostrazione qualitativa del fatto che il parametro di overlap m sia una grandezza self-averaging, abbiamo verificato che l'ordine di grandezza del sistema (per i valori di C e K da noi considerati) debba essere $N = \mathcal{O}(10000)$ per poter assumere di lavorare nel limite termodinamico (cfr. sottosezioni §4.2).

(iii) Parametrizzazione delle prestazioni al variare del parametro β: abbiamo dimostrato che le prestazioni dell'algoritmo di BP peggiorano considerevolmente se gli estimatori vengono pesati con valori che si discostano da quelli di β_p, coerentemente con la teoria statistica per una procedura MPM. Abbiamo inoltre evidenziato come nella forma espressa dalle (3.24) e (3.25), quest'ultimi equivalgono alle magnetizzazioni locali del sistema fisico. (cfr. sottosezione §4.3).

Definite le ottimali condizioni al contorno, siamo passati ad uno numerico delle prestazioni della classe dei codici LDGM. In primis, abbiamo cercato di capire l'effetto della geometria del factor graph sulle performance dell'algoritmo di BP: la nostra conclusione è che la geometria random fornisce sempre una soluzione ottimale. Come sottolineato nella sezione §5.1, le simulazioni da noi compiute hanno dimostrato qualitativamente l'esistenza di geometrie svantaggiose legate al

grado di ordine con cui viene costruito il grafo iniziale, in grado di compromettere le prestazioni dell'algoritmo di BP.

In una seconda fase abbiamo fornito un'ulteriore prova del fatto che l'overlap sia una grandezza self-averaging, dimostrando che il processo di termalizzazione dipenda fortemente da K (cfr. sezione §5.2). Infatti, i dati riportati in Figura 5.2 verificano che la termalizzazione del sistema è tanto più lenta quanto più grande è il parametro K.

Successivamente abbiamo studiato le performance dei codici LDGM in funzione dei principali parametri in gioco: il grado di interazione K, il rate R e l'errore intrinseco p per ogni bit introdotto nel canale. Più nel dettaglio, per i valori di $R = 1/2, 1/3$ e $1/4$ abbiamo mappato le curve $p_b^{(K)}$ in funzione dell'errore introdotto p al variare del numero di interazioni K, confrontando i risultati ottenuti con quelli restituiti dalla previsione teorica. Abbiamo verificato che le prestazioni migliorano notevolmente all'aumentare del parametro K a scapito però della diminuzione del valore dell'errore critico $p_c^{(K)}$, oltre il quale il codice non riesce più a ricostruire il messaggio in ingresso e non rispetta il valore atteso.

Infine abbiamo ricavato le ottimali condizioni al contorno dei codici SCLDGM e si è fornito un esempio delle loro performance.

Bibliografia

[1] Panagiotis Alevizos. «Factor Graphs: Theory and Applications». Tesi di dott. TECHNICAL UNIVERSITY OF CRETE, 2012.

[2] Claude Berrou, Alain Glavieux e Punya Thitimajshima. «Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1». In: *Communications, 1993. ICC'93 Geneva. Technical Program, Conference Record, IEEE International Conference on*. Vol. 2. IEEE. 1993, pp. 1064–1070.

[3] Raj Chandra Bose e Dwijendra K Ray-Chaudhuri. «On a class of error correcting binary group codes». In: *Information and control* 3.1 (1960), pp. 68–79.

[4] Tommaso Castellani e Andrea Cavagna. «Spin-glass theory for pedestrians». In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.05 (2005), P05012.

[5] Samuel Frederick Edwards e Phil W Anderson. «Theory of spin glasses». In: *Journal of Physics F: Metal Physics* 5.5 (1975), p. 965.

[6] Robert Gallager. «Low-density parity-check codes». In: *IRE Transactions on information theory* 8.1 (1962), pp. 21–28.

[7] Javier Garcia-Frias e Wei Zhong. «Approaching Shannon performance by iterative decoding of linear codes with low-density generator matrix». In: *IEEE Communications Letters* 7.6 (2003), pp. 266–268.

[8] Richard W Hamming. «Error detecting and error correcting codes». In: *Bell System technical journal* 29.2 (1950), pp. 147–160.

[9] Alexis Hocquenghem. «Codes correcteurs d'erreurs». In: *Chiffres* 2.2 (1959), pp. 147–56.

[10] Yoshiyuki Kabashima e David Saad. «Belief propagation vs. TAP for decoding corrupted messages». In: *EPL (Europhysics Letters)* 44.5 (1998), p. 668.

[11] Yoshiyuki Kabashima e David Saad. «Statistical mechanics of error-correcting codes». In: *EPL (Europhysics Letters)* 45.1 (1999), p. 97.

[12] Amrit Kharel e Lei Cao. «Analysis and Design of Serially Concatenated LDGM Codes». In: *arXiv preprint arXiv:1801.08270* (2018).

[13] Hans-Andrea Loeliger et al. «The factor graph approach to model-based signal processing». In: *Proceedings of the IEEE* 95.6 (2007), pp. 1295–1322.

[14] David JC MacKay. «Good error-correcting codes based on very sparse matrices». In: *IEEE transactions on Information Theory* 45.2 (1999), pp. 399–431.

[15] David JC MacKay e David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[16] Marc Mezard e Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[17] Andrea Montanari. «Turbo codes: The phase transition». In: *The European Physical Journal B-Condensed Matter and Complex Systems* 18.1 (2000), pp. 121–136.

[18] Andrea Montanari e Nicolas Sourlas. «The statistical mechanics of turbo codes». In: *The European Physical Journal B-Condensed Matter and Complex Systems* 18.1 (2000), pp. 107–119.

[19] Tatsuto Murayama et al. «Statistical physics of regular low-density parity-check error-correcting codes». In: *Physical Review E* 62.2 (2000), p. 1577.

[20] Hidetoshi Nishimori. *Statistical physics of spin glasses and information processing: an introduction*. Vol. 111. Clarendon Press, 2001.

[21] Giorgio Parisi. «A sequence of approximated solutions to the SK model for spin glasses». In: *Journal of Physics A: Mathematical and General* 13.4 (1980), p. L115.

[22] Claude Elwood Shannon. «A mathematical theory of communication». In: *Bell system technical journal* 27.3 (1948), pp. 379–423.

[23] David Sherrington e Scott Kirkpatrick. «Solvable model of a spin-glass». In: *Physical review letters* 35.26 (1975), p. 1792.

[24] Nicolas Sourlas. «Spin-glass models as error-correcting codes». In: *Nature* 339.6227 (1989), p. 693.

[25] James C Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*. Vol. 65. John Wiley & Sons, 2005.

[26] Davide Maria Tagliabue. «Meccanica statistica dei codici correttori: analisi euristica dei codici di Sourlas a connettività finita». Università degli Studi di Milano, 2018.

[27] Roderick Wong. *Asymptotic approximations of integrals*. Vol. 34. SIAM, 2001.