UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Corso di laurea magistrale in Ingegneria Elettronica

Analisi dei trapping set per codici MDPC con applicazioni in crittografia

Trapping set analysis for MDPC codes with applications in cryptography

Relatore: Prof.
FRANCO CHIARALUCE

Tesi di laurea di:
MICHELE PACENTI

Correlatore: Ing.
MASSIMO BATTAGLIONI

Correlatore: Ing.
PAOLO SANTINI

A.A. 2020/2021

So pensare, so aspettare, so digiunare.

Indice

1 Introduzione

Negli ultimi trent'anni la crittografia è diventata una componente indispensabile in qualsiasi infrastruttura digitale. Poiché le reti moderne supportano una enormità di servizi e applicazioni in pressoché qualsiasi settore produttivo e non, è di vitale importanza preservare la loro sicurezza (anche in vista della diffusione del 5G che sarà il motore dello sviluppo tecnologico dei prossimi anni).

Quella della crittografia può essere considerata a tutti gli effetti un'arte millenaria, dominata da una continua rincorsa tra chi studia nuovi sistemi di difesa per resistere a nuovi attacchi e chi, viceversa, studia nuovi metodi di attacco (*crittanalisi*) per violare nuovi sistemi di difesa: dalla *scitala*, descritta più volte da Plutarco [1], al celebre *cifrario di Cesare*, utilizzato sistematicamente dall'esercito di Giulio Cesare e dei suoi successori (lo sappiamo grazie agli scritti di Svetonio [2]), fino ai ben più recenti cifrari *Playfair* [3] e *ADFGVX* [4], utilizzati durante il primo conflitto mondiale rispettivamente dalle règie truppe inglesi e dall'esercito del II Reich. Appartengono ormai alla cultura popolare le vicende riguardanti Alan Turing e la macchina Enigma, anche grazie ad un ampio spettro di opere letterarie, cinematografiche e di documentari. Dal secondo dopoguerra, grazie allo sviluppo dell'informatica, delle telecomunicazioni, e soprattutto grazie alla nascita di *Arpanet* e poi *Internet*, il ruolo della crittografia diventa cruciale: infatti, mentre fino a prima di quel momento l'unico settore interessato allo sviluppo delle tecniche crittografiche era quello bellico, da lì in poi questa disciplina ha pervaso in maniera sempre più estensiva la vita di ciascun essere umano.

I moderni sistemi di crittografia si suddividono in due categorie: quelli *simmetrici* e quelli *asimmetrici*. Nei sistemi di crittografia simmetrica il testo cifrato viene generato a partire dal testo in chiaro e da una chiave privata; per risalire al testo in chiaro il destinatario deve conoscere la stessa chiave privata usata per la cifratura. Uno dei più importanti (e anche dei più discussi) sistemi a crittografia simmetrica è il *Data Encryption Standard* (DES) [5, 6], oggi sostituito dal più moderno e sicuro *Advanced Encryption Standard* (AES) [7, 8], che ha vinto la competizione indetta dal National Institute of Standard and Technology (NIST) nel 1997 [9] proprio per la sostituzione del DES.

Mentre la crittografia simmetrica è stata storicamente utilizzata sin da tempi antichissimi, ben più recente è la nascita dei sistemi *asimmetrici*. Nella crittografia asimmetrica ciascun utente possiede una coppia di chiavi, una pubblica e nota a tutti, e una privata. Per garantire la confidenzialità, il testo

in chiaro da inviare viene cifrato utilizzando la chiave pubblica del destinatario; in questo modo, il messaggio può essere decifrato soltanto utilizzando la chiave privata, della quale soltanto il destinatario è in possesso. Dualmente, la stessa coppia di chiavi può essere utilizzata per scopi di autenticazione: chi si vuole autenticare cifra un testo noto con la sua chiave privata (produce una *firma digitale*), mentre chi autentica decifra tale firma con la chiave pubblica dell'autore; se il testo decifrato coincide col testo noto (o, più verosimilmente, con una stringa dipendente dal testo noto), l'autenticazione ha successo. Poiché i sistemi asimmetrici sono più lenti delle loro controparti simmetriche, questi vengono utilizzati principalmente per la generazione e lo scambio delle chiavi segrete, che verranno poi utilizzate negli schemi simmetrici. I problemi matematici alla base della sicurezza dei sistemi asimmetrici sono principalmente due: la *fattorizzazione* di grandi numeri primi, e il calcolo del *logaritmo discreto*; questi sono problemi di difficile risoluzione per un calcolatore, nel senso che per arrivare ad una conclusione si impiegherebbe un tempo esageratamente lungo (svariate migliaia di anni); perciò si parla di *sicurezza computazionale*, cioè la protezione che questi sistemi offrono non è data dalla certezza matematica che non esista una soluzione al problema su cui si basano, ma semplicemente dal tempo che richiederebbe la sua risoluzione.

Il primo ad intuire la caratteristica di funzione *trapdoor* (cioè una funzione che non può essere invertita senza la conoscenza di una chiave segreta) del problema della *fattorizzazione* fu William Stanley Jevons nel 1877 [10], quando scrisse nel suo libro *The principles of science*:

> *"Can the reader say what two numbers multiplied together will produce the number 8616460799? I think it unlikely that anyone but myself will ever know."*

Fu però soltanto nel 1976 che Whitfield Diffie e Martin Hellman pubblicarono un articolo [11], in cui per la prima volta venne descritto un sistema di scambio di chiavi segrete su un canale pubblico, destinato a rivoluzionare per sempre l'universo crittografico, anche se a dire il vero già due anni prima, nel 1974, James Henry Ellis, Clifford Cocks e Malcolm John Williamson, membri del Government Communication Headquarters (GCHQ) inglese, crearono un sistema perfettamente analogo a quello di Diffie ed Hellman, ma la scoperta fu mantenuta classificata dal governo inglese fino al 1997 [12]. Ad oggi, oltre allo *scambio di chiavi Diffie-Hellman*, alcuni dei sistemi asimmetrici più importanti sono *Rivest-Shamir-Adleman* (RSA) [13], *ElGamal* [14] e *Elliptic Curve Digital Signature Algorithm* (ECDSA) [15].

Come accennato precedentemente in questo capitolo, la crittografia non può ridursi alla cifratura e alla decifratura di messaggi; oggi crittografia significa risolvere problemi concreti e complessi che riguardano, in maniera generale, la sicurezza dell'informazione [16], quali:

- **Confidenzialità**, ovvero preservare la riservatezza dei messaggi;
- **Integrità**, ovvero evitare l'alterazione dei messaggi trasmessi;
- **Autenticazione**, ossia avere la certezza di chi ha trasmesso un messaggio;
- **Non ripudiabilità**, ovvero l'impossibilità per un utente di negare di aver trasmesso un messaggio (che, in realtà, ha trasmesso proprio lui).

Questi concetti base vengono impiegati in uno stuolo di applicazioni diverse, tra cui: firma digitale [17, 18, 19, 20, 15] (ad esempio nella *blockchain* [21, 22] o nell'*online banking* [23, 24]), protocolli di autenticazione [25, 26, 27, 28, 29, 30, 31], scambio di chiavi [32, 33, 34], *secret sharing* [35, 36], time-stamping certificato [37, 38], ma anche elezioni online [39, 40], solo per nominarne alcune.

Si capisce quindi che, se un giorno la sicurezza delle primitive crittografiche che stanno alla base degli odierni protocolli di cybersecurity dovesse venire meno, le reti di telecomunicazione dell'intero pianeta non sarebbero più al sicuro. Dato che esistono già oggi macchine capaci di violare, con la loro potenza di calcolo tali sistemi, è necessaria una nuova "corsa agli armamenti" per sviluppare una nuova generazione di crittosistemi capaci di resistere a questa nuova tipologia di calcolatori.

1.1 La minaccia quantistica

La ricerca nell'ambito del *quantum computing* ha avuto inizio a partire degli anni 80, quando il fisico Paul Benioff propose un modello quantistico della macchina di Turing [41]. Pochi anni dopo il celebre fisico Richard Feynman ha intuito che per simulare in maniera efficiente i fenomeni quantistici sarebbe servito un computer quantistico [42]. La svolta è però arrivata nel 1985, grazie al lavoro fondamentale di David Deutsch [43], che diede il via allo sviluppo dei primi algoritmi quantistici; particolarmente rivoluzionari furono quelli di Peter Shor [44, 45] e Lov Grover [46, 47], che dimostrarono di riuscire

a risolvere problemi di fattorizzazione in tempo polinomiale, e di ricerca in tempo ridotto (seppur non polinomiale) rispettivamente, allargando così lo spettro delle potenziali applicazioni del quantum computing ad una platea ben più estesa rispetto alla proposta iniziale di Feynman, e incrementando quindi di molto l'interesse (e gli investimenti) verso la ricerca in tale ambito.

Il primo computer quantistico fu sviluppato nel 1999 dalla D-Wave Systems; chiaramente i primi modelli erano piuttosto rudimentali, e potevano contare su una quantità irrisoria di *qubit* (l'analogo quantistico del bit). Oggi esiste una pletora di grandi multinazionali e startup (Google, IBM, Microsoft, Rigetti [48], Xanadu [49], PsiQuantum [50], Honeywell [51], Ionq [52], D-Wave [53], QuTech [54], Quantum Brilliance [55] e altre) che stanno sviluppando, hanno sviluppato e/o stanno già commercializzando il loro prototipo, segno del fatto che il quantum computing è già realtà ed è già interesse strategico di aziende e Paesi.

Un computer quantistico è un calcolatore che sfrutta i principi della meccanica quantistica per eseguire calcoli; il ruolo del *bit* tradizionale (ossia quello di rappresentare l'informazione) è affidato al *quantum bit* (qubit). Mentre il bit tradizionale può assumere solo due stati (0 e 1) ed è implementato a livello fisico dalle tensioni su dei transistor, il qubit può assumere tutti gli infiniti stati compresi tra 0 ed 1 (un fenomeno chiamato *sovrapposizione coerente di stati* [56]); al momento della misurazione il qubit collassa su uno dei due stati "classici". Questo fenomeno viene sfruttato dagli algoritmi quantistici per riuscire a risolvere particolari problemi in tempo polinomiale. I qubit sono quindi rappresentazioni di fenomeni quantistici, che variano a seconda della tecnologia impiegata per la costruzione del computer (a superconduttori [57], a fotoni [58], a trappole ioniche [59], a topologia [60], a stato solido [61]).

Tralasciando il dibattito rispetto al ruolo che i computer quantistici avranno nel prossimo futuro (l'opinione dominante è che questi rimarranno macchine special purpose, ausiliarie ai computer tradizionali e utilizzate per risolvere alcuni problemi specifici; alcune correnti di pensiero però sono convinte che i quantum computer diventeranno macchine general purpose, sostituendo i computer a stato solido [62]), una cosa è certa: la potenza dell'algoritmo di Shor [44, 45], dimostrata anche empiricamente da Arute et al. (Google) [63], ha già avuto un impatto dirompente nella comunità scientifica, soprattutto nel mondo della crittografia. Riuscire a fattorizzare un numero sufficientemente grande in tempo polinomiale significa infatti riuscire a violare sistemi attualmente sicuri come RSA o ECDSA. È stato stimato che tra circa 20 anni esisteranno quantum computer sufficientemente potenti da violare gli attua-

li sistemi a chiave pubblica [64]; considerando che per sviluppare gli schemi odierni è stato impiegato più o meno lo stesso tempo, è presumibile che serviranno altrettanti anni per mettere in atto una migrazione sicura verso le nuove generazioni di primitive crittografiche, chiamate *Post-Quantum*.

1.2 Crittografia Post-Quantum

Nel 2016, durante la settima conferenza internazionale sulla crittografia post-quantum [65, 66], il NIST ha aperto il bando per la selezione dei nuovi standard crittografici: esattamente come avvenne per AES, una serie di algoritmi sviluppati da aziende, università e centri di ricerca competono per diventare i nuovi standard di crittografia a chiave pubblica (i sistemi simmetrici continuano ad essere sicuri anche da attacchi quantistici, a patto che vi sia un aumento della lunghezza della chiave per AES, o un aumento della lunghezza dell'output per algoritmi come SHA-2 e SHA-3 [67]). I nuovi sistemi richiesti dal bando devono prevedere:

- *Public-key encryption*: algoritmi di cifratura e decifratura, oltre che, ovviamente, la generazione delle chiavi;
- *Key encapsulation mechanism* (KEM): algoritmi per la generazione di chiavi e per il loro incapsulamento e decapsulamento;
- Algoritmi di *firma digitale*.

Sono state ricevute 82 candidature, di cui 23 algoritmi di firma digitale e 59 algoritmi di KEM e di cifratura, coinvolgendo 263 ricercatori provenienti da 24 paesi diversi (a maggioranza USA e Francia, ma con un'ottima rappresentanza italiana). Anche l'Università Politecnica delle Marche ha partecipato attivamente proponendo, in collaborazione col Politecnico di Milano, *LEDAcrypt* [68], che è stato ammesso al secondo round di selezione [69] ma non l'ha superato [70]. Attualmente gareggiano, nel terzo round, 7 finalisti (4 algoritmi di cifratura/KEM e 3 di firma digitale) più 8 candidati alternativi.

La sicurezza computazionale delle primitive crittografiche post quantum è data da problemi matematici difficili da risolvere anche per una macchina quantistica. In particolare i sistemi proposti finora possono dividersi in poche famiglie, ognuna delle quali sfrutta un problema differente. Le principali sono:

Crittografia basata su codici Se ne discuterà in maniera approfondita nel capitolo successivo.

Crittografia basata su reticoli Sono diverse le ragioni che rendono questa categoria di crittosistemi interessanti: diverse applicazioni sono rese possibili dalla crittografia basata su reticoli (crittografia omomorfica [71, 72], offuscamento di codici sorgente [73] e crittografia basata su attributi [74, 75]); inoltre questa consente generalmente l'implementazione di algoritmi semplici, efficienti e fortemente parallelizzabili. Tuttavia è difficile stimare con precisione i livelli di sicurezza di questi sistemi rispetto agli attacchi noti. Il problema computazionale più rilevante riguardante i reticoli è lo *Shortest Vector Problem* (SVP), ossia trovare il vettore con la lunghezza euclidea minore all'interno di un reticolo, ovvero l'insieme di tutte le combinazioni lineari di una base definita in $\mathbb{Q}^n$ o $\mathbb{Z}^n$. È ben noto che si tratta di un problema NP-hard[1] [76, 77, 78], anche se esistono molteplici altri problemi altrettanto difficili sempre basati su reticoli (come *Learning With Errors* (LWE) o *Module Learning With Rounding* (MLWR)). I crittosistemi basati su reticoli attualmente in gara nella competizione del NIST sono *CRYSTALS-KYBER* [79], *SABER* [80], *NTRU* [81] (oltre a *NTRU Prime* [82] e *FrodoKEM* [83] come candidati alternativi) per quanto riguarda cifratura e KEM, *CRYSTALS-DILITHIUM* [84] e *FALCON* [85] per quanto riguarda gli schemi di firma digitale.

Crittografia basata su polinomi multivariabili Questi schemi sono basati sulla difficoltà di risolvere sistemi di polinomi a più variabili su campi finiti. Dal lavoro iniziale di Imai e Matsumoto [86], passando all'introduzione delle *funzioni a campi nascosti* da parte di Jacques Patarin [87], gli schemi moderni si prestano maggiormente per la firma digitale, tanto che l' unico finalista nella competizione del NIST è proprio un algoritmo di digital signature, ovvero *Rainbow* [88], oltre a *GeMSS* [89] come candidato alternativo.

Firme basate su funzioni Hash Le funzioni Hash non costituiscono una novità nel mondo della cybersecurity, infatti sono già ampiamente utilizzate soprattutto in schemi di firma digitale [90, 91] e di autenticazione [92, 93, 94]. Sono funzioni cosiddette *one way*, cioè funzioni irreversibili: ricevono in ingresso uno o più input (stringhe di bit) e restituiscono in uscita un output (detto *digest*) più piccolo e completamente scorrelato dall'ingresso, al quale (almeno in linea teorica) è impossibile risalire. Alcune delle funzioni hash più importanti sono SHA-1, SHA-2 e MD5. Sono piuttosto sicure, anche contro attacchi

[1] NP sta per nondeterministic polynomial (time); evitando di entrare nei dettagli della *teoria della complessità*, è sufficiente dire che i problemi appartenenti classi NP-hard e NP-complete sono sufficientemente difficili da non poter essere risolti in tempo polinomiale.

quantistici, tuttavia presentano alcune insicurezze: ad esempio, possono produrre solo un numero limitato di firme, e l'unico modo per aumentare tale numero è incrementare la lunghezza della firma stessa. Attualmente gli unici schemi di firma digitale basati su hash sono entrambi candidati alternativi, ossia *Picnic* [95] e *SPHINCS+* [96].

Isogenie su curve ellittiche supersingolari Questa tipologia di algoritmi, di cui il capostipite è sicuramente l'algoritmo di Jao e De Feo [97] che per primi hanno introdotto le isogenie nello scambio di chiavi Diffie-Hellmann, è basata sempre su curve ellittiche e sfrutta appunto il problema delle isogenie piuttosto che quello del logaritmo discreto (come si fa in ECDSA); infatti, mentre l'algoritmo di Shor riesce a risolvere il problema del logaritmo discreto su curve ellittiche, il problema delle isogenie su curve supersingolari non sembra ancora risolubile da un computer quantistico. Ad oggi soltanto uno schema basato su isogenie è ancora in gara nella competizione NIST, cioè *SIKE* [98], come candidato alternativo.

Tra tutte le categorie di sistemi post quantum, quella basata sui codici è probabilmente, assieme a quella basata su reticoli, la più promettente; ad oggi, nella competizione NIST sono in gara tre di questi sistemi: il *McEliece classico* come finalista, *BIKE* [99] e *HQC* [100] come candidati alternativi. In particolare BIKE (acronimo di BIt flipping Key Encapsulation), rispetto al McEliece classico, utilizza codici *Moderate Density Parity Check* (MDPC), una classe di codici che, a differenza dei codici *Goppa* utilizzati in origine da McEliece, e similarmente ai ben noti codici *Low Density Parity Check* (LDPC), non hanno una sfera di decodifica deterministica; in altre parole, mentre la decodifica dei codici Goppa (identicamente a quella di qualsiasi codice algebrico, come i *Bose-Chaudhuri-Hocquenghem* (BCH) o i *Reed-Solomon* (RS)) garantisce il successo fino ad un certo numero massimo di errori, quella di codici come gli LDPC o gli MDPC è sempre affetta da una certa probabilità di fallire; questa probabilità è identificata col nome di *Decoding Failure Rate* (DFR), ed è cruciale in quanto è stato dimostrato che questa natura probabilistica del decoder espone il sistema ad attacchi basati sull'osservazione della fase di decodifica. Queste tecniche di crittanalisi vengono solitamente denominate *reaction attacks* quando si basano proprio sui fallimenti del decoder, o *side-channel attacks* quando si basano su altre informazioni relative alla decodifica come, ad esempio, la durata o altre quantità. Un sistema di crittografia basato su codici LDPC o MDPC è in grado di resistere a tali attacchi solo

se il DFR è sufficientemente piccolo e se l'algoritmo è implementato a tempo costante (ossia successo e fallimento della decodifica impiegano lo stesso tempo). Il DFR è intrinsecamente legato alla capacità correttiva del codice, e viene solitamente stimato mediante simulazioni Monte Carlo, in quanto difficile da modellare teoricamente. D'altra parte, in un contesto crittografico (e non solo), proprio per i motivi citati poc'anzi, il DFR dovrebbe essere talmente basso (2^{-80} o meno) che è impossibile da "fotografare" mediante simulazioni numeriche; argomento attuale di ricerca è quindi lo sviluppo di modelli analitici che consentano una stima accurata del DFR senza l'ausilio di simulazioni numeriche. Per quanto riguarda i codici LDPC, è ben nota l'esistenza di una regione di *error floor* nella curva del DFR, ossia una regione in cui la curva riduce di molto la sua pendenza, costituendo quindi un limite inferiore ad esso. La regione di error floor (che non può essere analizzata mediante simulazioni per le motivazioni descritte prima) viene caratterizzata teoricamente mediante lo studio di particolari strutture all'interno del codice chiamate genericamente *trapping set*, ossia particolari pattern di errore a basso peso che il decoder iterativo non riesce a correggere. Mentre lo studio dei trapping set nei codici LDPC è argomento ben consolidato in letteratura, totalmente inesplorato rimane per quanto riguarda i codici MDPC: la maggiore densità della matrice di parità di questi codici, infatti, comporta in primis una complessità eccessiva che rende inutilizzabili praticamente tutti gli algoritmi di ricerca di trapping set presenti in letteratura, e in secondo luogo crea delle nuove topologie all'interno del codice, che probabilmente rendono necessarie una nuova caratterizzazione grafica e strutturale dei trapping set più pericolosi per questa categoria di codici. L'obiettivo di questa tesi è perciò quello di esplorare, indagare e caratterizzare i trapping set più pericolosi all'interno di codici MDPC (in particolar modo in quelli utilizzati in BIKE), e di stabilire un punto di partenza per studi futuri, proponendo alcune possibili tecniche mai impiegate prima in questo ambito (come la ricerca di *clique*) e allo stesso tempo sfruttando la teoria già nota in letteratura sugli LDPC.

2 Crittografia basata su codici

La crittografia basata su codici nasce nel 1978 grazie al contributo di Robert J. McEliece [101], che ebbe per primo l'intuizione di impiegare i codici a correzione d'errore nell'ambito crittografico. La sicurezza del crittosistema di McEliece risiede nella difficoltà di decodificare un codice di grandi dimensioni senza conoscerne la struttura, il che è notoriamente un problema NP-complete [102], tanto che il sistema non è ancora stato violato (precisamente non esiste ancora un algoritmo, tradizionale o quantistico che sia, che riesce a risolvere tale problema in tempo polinomiale) ed è tuttora in gara come finalista nella competizione del NIST; oltretutto è diversi ordini di grandezza più veloce rispetto ad altre soluzioni a chiave pubblica che in passato ne hanno ostacolato l'utilizzo su larga scala, come RSA. Tuttavia, la proposta originale di McEliece presenta due problemi principali: l'eccessiva lunghezza delle chiavi pubbliche e il basso rate trasmissivo.

Prima di proseguire con la descrizione del crittosistema di McEliece, è opportuno un richiamo alla notazione che si utilizzerà di seguito. Un codice C binario è descritto dalla terna

$$(n, k, d)$$

dove:

- n è la lunghezza del codice;
- k è la dimensione del codice;
- d è la distanza minima.

Il rate del codice è definito come $R = k/n$. Un codice è detto *lineare* quando la somma di due parole di codice è ancora una parola di codice. Un codice lineare può essere rappresentato in forma matriciale da una *matrice generatrice* $\mathbf{G}$, di dimensione $k \times n$ e rango k, o dualmente da una *matrice di parità* $\mathbf{H}$, di dimensione $(n-k) \times n$ e rango $(n-k)$. Sia $\mathbf{u}$ una sequenza di informazione di lunghezza k; la corrispondente parola di codice $\mathbf{c} \in \mathrm{C}$, di lunghezza n, è ottenuta dalla moltiplicazione:

$$\mathbf{c} = \mathbf{u} \cdot \mathbf{G}. \tag{1}$$

Come ben noto, condizione necessaria e sufficiente affinché $\mathbf{c} \in \mathrm{C}$ è che la sua moltiplicazione per la trasposta della matrice di parità $\mathbf{H}$ del codice dia come

risultato un vettore nullo:

$$\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0}. \tag{2}$$

Una sequenza di lunghezza n affetta da errori può essere considerata come la somma tra una parola di codice ed un vettore di errore:

$$\tilde{\mathbf{c}} = \mathbf{c} + \mathbf{e} \tag{3}$$

Moltiplicando $\tilde{\mathbf{c}}$ per la matrice di parità, considerando anche la (2), si ottiene la *sindrome* $\mathbf{s}$:

$$\mathbf{s} = \tilde{\mathbf{c}} \cdot \mathbf{H}^T = \mathbf{c} \cdot \mathbf{H}^T + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T \tag{4}$$

che, nel caso binario, non è altro che la somma delle colonne di $\mathbf{H}$ posizionate in corrispondenza degli elementi non nulli di $\mathbf{e}$; nel caso non binario tali colonne saranno anche moltiplicate per il valore di $\mathbf{e}$ in tali posizioni. Tuttavia, di seguito, si considererà esclusivamente il caso binario.

Tornando al sistema di McEliece, la proposta originale sfrutta codici *Goppa* (vale a dire una particolare famiglia di codici algebrici), con $n = 1024$, $k = 524$ che correggono fino a $t = 50$ errori. I codici Goppa, impiegati nello schema di McEliece, presentano tutte le caratteristiche che un codice dovrebbe avere per poter essere impiegato in schemi di questo tipo [103]:

- Data la terna (n, k, d), il codice deve essere grande abbastanza da evitare enumerazioni (enumerazione delle parole di codice, ad esempio);
- l'algoritmo di decodifica deve essere sufficientemente efficiente, ossia avere una sufficiente capacità correttiva ed essere oltremodo rapido;
- la matrice generatrice $\mathbf{G}$ (o la matrice di parità $\mathbf{H}$), di un codice equivalente a quello scelto non deve fornire informazioni riguardo la struttura del codice originale.

La chiave pubblica e la chiave privata, nel crittosistema di McEliece, sono costituite entrambe dalla matrice generatrice di un codice lineare. La coppia di chiavi viene generata scegliendo casualmente un polinomio irriducibile di grado t in $GF(n)$ (*campo di Galois*), dal quale si genera la matrice generatrice $\mathbf{G}$, che costituisce la chiave privata. Dopo aver generato una matrice random, densa, non singolare, $\mathbf{S}$ detta di *scrambling*, di dimensione $k \times k$ e una matrice

di permutazione[2] $n \times n$ $\mathbf{P}$, la chiave pubblica si ottiene come

$$\mathbf{G}' = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P}. \tag{5}$$

La chiave pubblica $\mathbf{G}'$ è la matrice generatrice di un codice lineare con lo stesso rate e la stessa distanza minima del codice originale.

Come sovente si usa fare in crittografia, si descrive ora il processo di cifratura e decifratura supponendo che due utenti, Alice e Bob, vogliano comunicare. Si ipotizzi che Alice voglia inviare un messaggio a Bob; Alice divide il testo in chiaro in blocchi di n bit, e recupera la chiave pubblica di Bob, ovvero $\mathbf{G}'$. Per ciascun blocco $\mathbf{u}$ Alice esegue l'operazione:

$$\mathbf{x} = \mathbf{u} \cdot \mathbf{G}' + \mathbf{e} = \mathbf{c} + \mathbf{e} \tag{6}$$

dove $\mathbf{e}$ è un vettore casuale generato da Alice, di lunghezza n e peso t, che può essere visto come un "vettore d'errore intenzionale", e il cui peso non deve superare la capacità correttiva del codice.

Bob, che riceve il messaggio, calcola:

$$\mathbf{x}' = \mathbf{x} \cdot \mathbf{P}^{-1} = (\mathbf{u} \cdot \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P} + \mathbf{e}) \cdot \mathbf{P}^{-1} = \mathbf{u} \cdot \mathbf{S} \cdot \mathbf{G} + \mathbf{e} \cdot \mathbf{P^{-1}}. \tag{7}$$

$\mathbf{x}'$ è quindi una parola del codice segreto di Bob (corrispondente al vettore di informazione $\mathbf{u}' = \mathbf{u} \cdot \mathbf{S}$), affetta da un vettore di errore $\mathbf{e} \cdot \mathbf{P^{-1}}$ di peso t. Una volta che il decoder di Bob corregge gli errori, l'informazione originale $\mathbf{u}$ viene trivialmente recuperata calcolando $\mathbf{u}' \cdot \mathbf{S^{-1}}$.

La sicurezza contro attacchi a forza bruta è garantita dal numero enorme di possibili matrici $\mathbf{G}$, $\mathbf{S}$ e $\mathbf{P}$: considerando i parametri proposti da McEliece, ovvero $n = 1024$, $t = 50$, $k = 524$, il numero di polinomi irriducibili di grado 50 in $GF(1024)$ è circa 10^{149}, il numero di matrici di scrambling $k \times k$ è 2^{524^2} e quello delle possibili matrici di permutazione è 1024!; questi numeri sono grandi abbastanza da rendere il sistema sicuro contro attacchi a recupero della chiave. Un altro possibile attacco a forza bruta consiste nel cercare di recuperare $\mathbf{u}$ partendo da $\mathbf{x}$ senza calcolare la chiave, cioè sostanzialmente cercare di decodificare un codice incognito di lunghezza n e dimensione k in presenza di t errori, che è un problema NP-hard [102].

[2]Una matrice di permutazione è una matrice quadrata ottenuta scambiando alcune righe o colonne della matrice identità. Una matrice $\mathbf{A}$ moltiplicata per una matrice di permutazione permette di ottenere una matrice $\mathbf{A}'$, identica ad $\mathbf{A}$ ma con alcune righe o colonne (a seconda del verso della moltiplicazione) scambiate; allo stesso modo, un vettore $\mathbf{v}$ moltiplicato per una matrice di permutazione genera un vettore $\mathbf{v}'$ identico a $\mathbf{v}$ ma con un diverso ordine degli elementi.

La principale famiglia di attacchi al crittosistema di McEliece (e, più in generale, ai crittosistemi basati su codici), sono quelli basati su *Information Set Decoding* (ISD). Un information set per un codice lineare è definito come un set di k valori $\in [1; n]$ [3] tali che due parole di codice qualsiasi differiscano per almeno una delle posizioni corrispondenti [104]; i bit delle parole di codice in tali posizioni possono in tal modo essere considerati bit di informazione (i.e. le colonne della matrice $\mathbf{G}$ in quelle posizioni sono linearmente indipendenti). Un algoritmo ISD è quindi un algoritmo di decodifica che sfrutta le informazioni date da un information set per trovare una parola di codice partendo dalla versione affetta da errore della stessa (fintanto che l'errore è a basso peso) o per trovare una parola di codice a basso peso; questi attacchi si dividono in *classici* [104, 105, 106, 107] e *moderni* [108, 109, 103, 110, 111], a seconda delle tecniche sfruttate dall'algoritmo ISD; in particolar modo, gli attacchi ISD moderni sfruttano tecniche come la ricerca di collisioni basata sul paradosso del compleanno e uso di rappresentazioni.

Un primo esempio di algoritmo ISD è stato proposto dallo stesso McEliece in [101]: essendo $\mathbf{e}$ generato casualmente, può accadere che nessuno dei suoi simboli non nulli si trovi in corrispondenza di cifre di informazione della parola di codice; in tal caso è facile ottenere il testo in chiaro partendo dal corrispondente testo cifrato. Il codice ha dimensione k, cioè la sequenza di informazione è lunga k e la matrice generatrice (sia la privata che la pubblica) hanno k righe. Si possono quindi considerare soltanto k simboli del testo cifrato $\mathbf{x}$ e di $\mathbf{e}$ (basandosi proprio sull'information set), assieme alle k colonne corrispondenti (nelle stesse posizioni) della chiave pubblica $\mathbf{G}'$, e la (6) può essere riscritta come:

$$\mathbf{x}_k = \mathbf{u} \cdot \mathbf{G'}_k + \mathbf{e}_k. \tag{8}$$

Se il vettore d'errore $\mathbf{e}$ presenta solo zeri nelle k posizioni scelte si avrà che $\mathbf{e}_k = \mathbf{0}$ e quindi che $\mathbf{x}_k = \mathbf{u} \cdot \mathbf{G'}_k$. Sarà quindi facile per l'attaccante risalire a $\mathbf{u}$ calcolando $\mathbf{u} = \mathbf{x}_k \cdot \mathbf{G'}_k^{-1}$ (assumendo che $\mathbf{G'}_k$ sia invertibile). Inevitabilmente, per condurre questo tipo di attacco si deve essere in grado di riconoscere quando $\mathbf{e}_k = \mathbf{0}$ e, soprattutto, che le k posizioni scelte corrispondano proprio ai simboli di informazione; esistono quindi diversi algoritmi per assicurarsi che il risultato di $\mathbf{x}_k \cdot \mathbf{G'}_k^{-1}$ sia proprio $\mathbf{u}$.

Indubbiamente di maggiore interesse per questo lavoro di tesi sono gli attacchi cosiddetti *message resend*: il McEliece classico non riesce infatti a proteggere messaggi trasmessi più di una volta, o più in generale, messaggi

[3] Per $[a, b]$ si intende l'insieme degli interi compresi tra a e b, estremi compresi.

legati tra loro da relazioni lineari note [112]; non è quindi per forza necessario conoscere le caratteristiche del codice segreto per riuscire a decifrare un testo, ma è talvolta sufficiente partire da un insieme di testi cifrati. Questo mette in evidenza il fatto che un crittosistema debba rispettare delle caratteristiche di sicurezza ben precise in relazione all'informazione che un testo cifrato può fornire ad un potenziale attaccante. Queste caratteristiche vengono identificate da definizioni precise: *indistinguishability under chosen plaintext attack* (IND-CPA), *indistinguishability under (non-adaptive) chosen ciphertext attack* (IND-CCA) e *indistinguishability under adaptive chosen ciphertext attack* (IND-CCA2). La condizione di IND-CCA2 implica le altre due, quella di IND-CCA1 implica IND-CPA ma non IND-CCA2, mentre IND-CPA è la condizione più "debole". Il McEliece originale non è sicuro contro attacchi a testo cifrato scelto adattivi (soddisfa quindi IND-CCA), anche se la condizione di IND-CCA2 può essere ottenuta mediante opportune conversioni [113, 114, 115].

2.1 IND-CCA nel McEliece basato su codici random

Questa categoria di attacchi rappresentano la ragion d'essere di questa tesi, considerando però, in luogo del McEliece classico, la sua variante che utilizza codici QC-MDPC (al posto dei tradizionali Goppa) [116], sulla quale si basa BIKE. Particolarmente rilevante è il *reaction attack* (o attacco a reazione) illustrato in [117]. Un attacco a reazione è una versione più debole di un attacco a testo cifrato scelto. L'attaccante invia versioni modificate di testi cifrati intercettati e osserva la reazione del ricevitore; così facendo è possibile in certi casi trovare in maniera efficiente il messaggio corrispondente. Nello specifico si tratta di un attacco adattivo a recupero della chiave che sfrutta la probabilità di fallimento del decoder iterativo in codici QC-MDPC. Nello schema di McEliece basato su QC-MDPC la chiave segreta è costituita dalla matrice di parità stessa, che assume la forma

$$\mathbf{H} = [\mathbf{H}_0 \mid \mathbf{H}_1 \mid ... \mid \mathbf{H}_{n_0-1}] \tag{9}$$

dove ciascun $\mathbf{H}_i$ è un blocco circolante di dimensione $p \times p$ con peso di riga w_i. Poiché questi blocchi sono circolanti, possono essere rappresentati dalla loro prima riga (o prima colonna), consentendo quindi la considerevole riduzione della dimensione delle chiavi rispetto al McEliece classico, dove queste

sono costituite dall'intera matrice $\mathbf{G}$. Se ciascun blocco cicolante $\mathbf{H}_i$ viene rappresentato dalla prima riga $\mathbf{h}_i$, la chiave segreta sarà costituita dall'insieme $(\mathbf{h}_0, ..., \mathbf{h}_{n_0-1})$. In [117] si sfrutta la correlazione che c'è tra la probabilità di fallimento del decoder rispetto a vettori di errori appositamente scelti e l'esistenza di una distanza δ (i.e. la differenza) tra due elementi di $\mathbf{h}_0$.

È possibile resistere a questa tipologia di attacco se il DFR del codice è sufficientemente basso: in particolare è stato stabilito che per avere λ bit di sicurezza, si debba avere un DFR$\leq 2^{-\lambda}$. Il problema è come dimostrare che il DFR sia sufficientemente basso: lo stato attuale dell'arte fornisce più che altro stime, basate su simulazioni dalle quali si estrapolano limiti superiori, risultati asintotici [118] o modelli Markoviani del decoder [119]. Questi risultati studiano il comportamento tipico del decoder, ma probabilmente non prendono in considerazione due fenomeni che influenzano negativamente la capacità correttiva del decoder, che sono di due tipi: *chiavi deboli* e, come già accennato nella Sezione 1, error floors. Si noti che è certo che questi oggetti esistano, tuttavia la questione è definire quanto essi contribuiscano al DFR.

Chiavi deboli Come detto in precedenza, la chiave privata nel sistema considerato è determinata dalla struttura dell'intera matrice $\mathbf{H}$ e, conseguentemente, questa avrà un certo impatto sulla fase di decodifica. In [120] sono illustrate alcune chiavi deboli, che hanno una bassa densità ma allo stesso tempo un forte impatto sul decoder, anche se dai risultati illustrati in [121] si intuisce che queste particolari chiavi siano molto poche e abbiano un impatto trascurabile sul DFR.

Error floors Il DFR di codici LDPC subisce un fenomeno noto come error floor: al diminuire del tasso d'errore la curva del DFR passa da una fase a "cascata" ad una fase a bassa pendenza, formando un plateau. Esiste una letteratura ben consolidata su questo fenomeno: è dovuto infatti ad una serie di strutture piccole all'interno di $\mathbf{H}$ che inducono il decoder all'errore. Per strutture si intendono delle sottomatrici della $\mathbf{H}$ caratterizzate da una particolare distribuzione delle cifre non nulle o, analogamente, a dei sottografi nel grafo di Tanner (che verrà introdotto nella Sezione successiva) caratterizzati da particolari topologie; si rimanda alla Sezione 4 per i dettagli. A seconda del canale e del tipo di decoder utilizzato le strutture considerate pericolose variano, e pertanto assumono anche diverse denominazioni: *near-codewords*, *pseudo-codewords*, *stopping set* o *trapping set*; è accertato che fenomeni del

genere si verificano anche nei codici QC-MDPC utilizzati in BIKE. L'obiettivo di questa tesi è appunto indagare a tal riguardo.

3 Codici Low Density Parity Check

Un codice LDPC C(n, k) è solitamente definito dalla sua matrice di parità $\mathbf{H}$ i cui elementi, nel caso binario, potranno assumere solo i valori 0 e 1. Questa matrice può essere rappresentata sotto forma di un grafo bipartito, chiamato grafo di Tanner [122], il quale è costituito da due gruppi di nodi: i nodi "variabile", che sono tanti quanti il numero di bit n della parola di codice, e i nodi "check" (o di controllo), che sono tanti quanti il numero di righe nella matrice di parità $r = n - k$. Essendo il grafo di Tanner bipartito, ciascun nodo variabile è collegato esclusivamente ad uno o più nodi check, così come ciascun nodo check è collegato soltanto ad uno o più nodi variabile; un nodo variabile v_j è collegato ad un nodo check c_i se e solo se l' elemento $h_{ij} \in \mathbf{H}$ è pari a 1, cioè solo se il j-esimo bit della parola di codice partecipa all'i-esima equazione di parità. Il grado di un nodo è il numero di *rami* ad esso connessi (i.e., il numero di nodi collegati): ne segue che il grado dell'i-esimo nodo check corrisponde al peso [4] dell'i-esima riga di $\mathbf{H}$, mentre il grado del j-esimo nodo variabile corrisponde al peso della j-esima colonna di $\mathbf{H}$; se il codice è regolare, lo sarà anche il grafo di Tanner.

Un *ciclo* nel grafo di Tanner è un percorso chiuso che parte da un nodo variabile (o un nodo check) e torna allo stesso nodo senza passare più di una volta sullo stesso ramo. Poiché il grafo è bipartito, un ciclo conterrà un numero pari di nodi. Dal punto di vista della matrice di parità, un ciclo è formato da un insieme di colonne che presentano elementi non nulli in alcune posizioni in comune. I cicli più significativi sono quelli più corti, di lunghezza 4, 6 e 8, poiché essi hanno un maggiore impatto sulle prestazioni.

Esempio 1. *A sinistra un esempio di ciclo da 4, a destra un ciclo da 6.*

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

[4] Il peso di Hamming di un vettore corrisponde al numero di elemementi non nulli.

In altre parole, un ciclo corrisponde ad un poligono nella matrice di parità, i cui vertici sono elementi non nulli e i cui lati sono allineati alle righe e alle colonne di **H**; il numero di lati corrisponde alla lunghezza del ciclo. La lunghezza del ciclo più corto presente all'interno di **H** è chiamata *girth*.

La decodifica dei codici LDPC avviene mediante una classe di algoritmi noti come *belief propagation* che sfruttano il grafo di Tanner. Negli LDPC la matrice di parità **H** è molto sparsa: in genere il peso di riga è dell'ordine di $\log_2(n)$; in questo modo i nodi nel grafo di Tanner, se ben progettato, mantengono un grado relativamente basso, riducendo quindi il numero di cicli corti e permettendo agli algoritmi di decodifica di mantenere prestazioni elevate, oltre che una complessità molto bassa.

3.1 Codici QC-LDPC

Un codice LDPC è ben progettato se riesce a garantire una buona capacità correttiva e allo stesso tempo una bassa complessità di codifica e decodifica. Una particolare soluzione per raggiungere tali caratteristiche è quella di sfruttare le proprietà dei codici quasi ciclici (QC). Tali codici sono stati studiati per la prima volta da Townsend e Weldon in [123], dove un codice quasi ciclico è definito come un codice a blocco lineare con dimensione $k = p \cdot k_0$, lunghezza $n = p \cdot n_0$ e ridondanza $r = p \cdot r_0$ (con $r_0 = n_0 - k_0$), e ha la caratteristica che ciascuna permutazione ciclica di p posizioni di una parola di codice è anch'essa una parola di codice. La matrice di parità **H** di un codice quasi ciclico è costituita da blocchi circolanti. Un blocco circolante **A** è una matrice quadrata $p \times p$ definita come:

$$\mathbf{A} = \begin{pmatrix} a_0 & a_1 & a_2 & \dots & a_{p-1} \\ a_{p-1} & a_0 & a_2 & \dots & a_{p-2} \\ a_{p-2} & a_{p-1} & a_0 & \dots & a_{p-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & a_3 & \dots & a_0 \end{pmatrix} \tag{10}$$

dove ogni elemento $a_i \in F = GF(q)$ ($q = 2$ nel caso binario) e $i \in [0, p-1]$. Un blocco circolante è regolare (cioè tutte le righe e le colonne hanno lo stesso peso), dato che ogni riga (e ogni colonna) è una permutazione ciclica della prima. L'insieme delle matrici circolanti $p \times p$ forma un anello chiuso nella somma e nel prodotto modulo 2; l'elemento nullo coincide con la matrice nulla, e l'elemento unitario è la matrice identità di dimensione $p \times p$. È possibile quindi definire un isomorfismo tra tale anello e l'algebra dei polinomi tale che:

$$\mathbf{A} \leftrightarrow a(x) = \sum_{i=0}^{p-1} a_i \cdot x^i. \tag{11}$$

In altre parole una matrice circolante è associata univocamente ad un polinomio nella variabile x con coefficienti in F dati dagli elementi della prima riga della matrice (che sono poi gli stessi delle righe successive):

$$a(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_{p-1} x^{p-1}. \tag{12}$$

Chiaramente la matrice nulla corrisponde al polinomio nullo e la matrice identità corrisponde alla costante a_0. L'operazione di trasposizione viene convertita in forma polinomiale dalla seguente formula:

$$\mathbf{A}^T \leftrightarrow a(x)^T = \sum_{i=0}^{p-1} a_{p-1-i} \cdot x^i. \tag{13}$$

La possibilità di rappresentare le matrici di parità tramite polinomi permette di utilizzare algoritmi di cifratura e decifratura estremamente più efficienti. Esistono infatti alcune tecniche efficienti di moltiplicazione di polinomi come il *Fast Polynomial Product* [124, 125, 126] o l'*Optimized Vector-Circulant Matrix Product* [127], con quest'ultimo che permette di ottenere una complessità di codifica (effettuando l'operazione $\mathbf{u} \cdot \mathbf{G}$) pari a $\mathcal{O}((k_0-1) \cdot n_0 \cdot p)$, a fronte della complessità standard $\mathcal{O}(p^2/2)$.

Una particolare famiglia di codici quasi ciclici è quella con $r_0 = 1$; in tal caso la matrice di parità è composta da un solo strato di circolanti:

$$\mathbf{H} = [\mathbf{H}_0 \ \mathbf{H}_1 \ ... \ \mathbf{H}_{n_0-1}] \tag{14}$$

dove ciascun $\mathbf{H}_i$ è un blocco circolante, mentre il rate del codice è $R = (n_0 - 1)/n_0$. Come precedentemente accennato, sovente vengono utilizzati in crittografia codici costruiti in tale maniera, con $n_0 = 2$ e quindi $R = 1/2$,

costituiti quindi da due blocchi circolanti. Nonostante tali vincoli sulla struttura, sulla lunghezza e sulla rate si hanno comunque sufficienti gradi di libertà per modellare codici per un ampia gamma di applicazioni. Inoltre, più in generale, i QC-LDPC permettono l'utilizzo di tecniche di design studiate ad hoc per evitare la formazione di cicli brevi [128, 129, 130, 131].

3.2 Decodifica a decisione Hard in codici LDPC

Notoriamente i codici LDPC riescono ad ottenere un'ottima capacità correttiva su canali sui quali può essere usata l'informazione soft, come l'*Additive White Gaussian Noise* (AWGN), grazie agli algoritmi di decodifica belief propagation a decisione soft. Tuttavia nello scenario del sistema di McEliece non esiste informazione soft relativa ai bit ricevuti: come già illustrato nella Sezione 2, ad ogni parola di codice viene aggiunto un numero fisso di errori (corrispondente al peso del vettore $\mathbf{e}$), e poiché un bit errato subisce una transizione da 0 a 1 o viceversa, l'errore stesso è da considerarsi hard; piuttosto che il canale AWGN quindi, il sistema viene modellato dal *Binary Symmetric Channel* (BSC): nel canale BSC ogni bit ha una probabilità p di subire una transizione e, conseguentemente, ha una probabilità pari a $1 - p$ di rimanere invariato.

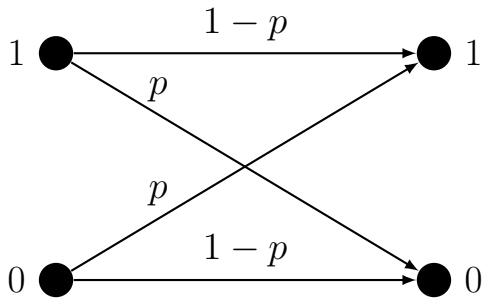


Figura 1: Schema di BSC con probabilità di crossover p.

Se nel crittosistema di McEliece vengono aggiunti t errori distribuiti casualmente, la probabilità di crossover sarà esattamente $p = t/n$.

Sebbene algoritmi a decisione soft come il classico *Log-Likelihood Ratios Sum-Product Algoritm* (LLR-SPA) siano impiegabili anche nei canali BSC con ottime prestazioni, si preferisce utilizzare algoritmi di decodifica a decisione hard con performance peggiori ma di gran lunga meno complessi e più rapidi.

Uno degli algoritmi di decodifica a decisione hard più importanti è l'algoritmo *Bit Flipping* (BF). Il principio dell'algoritmo BF è stato introdotto dallo stesso ideatore dei codici LDPC, Robert G. Gallager [132], ed è definito come segue. Si consideri la matrice di parità $\mathbf{H}$ di un codice LDPC che (per semplicità) ha un peso di colonna costante d_v. Ad ogni nodo variabile viene assegnato inizialmente il valore del corrispondente bit della parola ricevuta; ad ogni iterazione ogni nodo check c_i invia un messaggio ad ogni nodo variabile v_j cui è collegato, contenente la somma binaria dei valori di tutti i nodi variabile connessi ad esso. Ogni nodo variabile v_j riceve d_v valori di controllo ed effettua un semplice conteggio del numero delle equazioni di parità non soddisfatte eccetto quella corrispondente a c_i (in altre parole, se il messaggio ricevuto da un nodo check è 0 l'equazione di parità è soddisfatta, viceversa se è stato ricevuto un 1). Il risultato di questo conteggio, ossia il numero di equazioni di parità non soddisfatte a cui partecipa ciascun nodo variabile v_j, viene confrontato con una soglia $b \leq d_v - 1$, opportunamente scelta; se tale numero è $\geq b$ allora il bit corrispondente a v_j viene "flippato" (effettua una transizione da 0 a 1 o viceversa), altrimenti rimane invariato. All'iterazione successiva, ogni nodo variabile v_j invierà ad ogni nodo check a cui è collegato il valore aggiornato del bit corrispondente. È immediato intuire che le prestazioni del decoder dipendono dalla scelta della soglia b: lo stesso Gallager propose due algoritmi, *Algorithm A* e *Algorithm B*, nei quali la soglia b è fissata a $d_v - 1$ o varia tra $\lceil d_v/2 \rceil$ e $d_v - 1$ in maniera adattiva, o semplicemente viene fissata in tale intervallo.

L'implementazione del decoder BF è descritta dall'Algoritmo 1; il BF, sebbene ispirtato all'algoritmo di decodifica Gallager A/B, differisce leggermente. Nell'Algoritmo 1,$\mathbf{s}$ è la sindrome, $\mathbf{b}$ è un vettore contenente le soglie b_i per ciascun bit, i_{max} è il numero massimo di iterazioni, F è il vettore delle posizioni dei bit stimati errati e σ_i è il contatore delle equazioni di parità non soddisfatte a cui partecipa l'i-esimo bit. L'operatore $S(\mathbf{x})$ indica il supporto del vettore $\mathbf{x}$, mentre con $\mathbf{h}_i$ si intende l'i-esima colonna di $\mathbf{H}$. Dalla riga 1 alla 7 vengono sostanzialmente inizializzate le variabili; il loop nelle righe 8-10 aggiorna il valore del contatore per l'i-esimo bit: s'_l indica l'l-esimo elemento della sindrome, dove l'indice l percorre tutte le equazioni di parità a cui partecipa l'i-esimo bit. In sostanza, se un bit partecipa ad una equazione di parità non soddisfatta, il corrispondente elemento della sindrome sarà 1, e il contatore verrà incrementato; viceversa, se l'equazione è soddisfatta, il corrispondente elemento della sindrome sarà 0 e il contatore rimarrà costante. Nelle righe 11-12 il contatore viene confrontato con la soglia e viene aggiorna-

ta la stima del vettore d'errore. Nelle righe 15-18 il vettore d'errore stimato $\mathbf{e}'$ viene aggiornato così come la sindrome. Se la sindrome $\mathbf{s}'$ è nulla, allora necessariamente il vettore d'errore stimato corrisponde a quello reale, a meno di avere un vettore d'errore tale da originare un'altra parola di codice. Se entro i_{max} iterazioni non si riesce ad ottenere una sindrome nulla il decoder termina la procedura con un fallimento.

Algorithm 1 BF Decoder

Input: $\mathbf{H} \in \mathbb{F}_2^{r \times n}$, $\mathbf{s} \in \mathbb{F}_2^r$, $i_{max} \in \mathbb{N}$, $\mathbf{b} = [b_0, ..., b_{n-1}]$, $b_i \in [1, v_i]$, $\forall i$
Output: $\mathbf{e}' \in \mathbb{F}_2^n$

1: $\mathbf{e}' \leftarrow \mathbf{0}_n$
2: $\mathbf{s}' \leftarrow \mathbf{s}$
3: NumIter $\leftarrow 0$
4: **do**
5: $\quad F \leftarrow \varnothing$
6: $\quad$ **for** $i \leftarrow 0$ **to** $n-1$ **do**
7: $\quad\quad \sigma_i \leftarrow 0$
8: $\quad\quad$ **for** $l \in S(\mathbf{h}_i)$ **do**
9: $\quad\quad\quad \sigma_i \leftarrow \sigma_i + s'_l$
10: $\quad\quad$ **end for**
11: $\quad\quad$ **if** $\sigma_i \geq b_i$ **then**
12: $\quad\quad\quad F \leftarrow F \cup i$ ▷ La i-esima posizione è stimata come affetta da errore
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad$ **for** $i \in F$ **do**
16: $\quad\quad e'_i \leftarrow e'_i \oplus 1$ ▷ Aggiornamento vettore errore
17: $\quad\quad \mathbf{s}' \leftarrow \mathbf{s}' \oplus \mathbf{h}_i$ ▷ Aggiornamento sindrome
18: $\quad$ **end for**
19: $\quad$ NumIter $\leftarrow$ NumIter $+ 1$
20: **while** NumIter $\leq i_{max}$ **and** $\mathbf{s}' \neq \mathbf{0}_r$
21: **return** $\{\mathbf{e}'\}$

Il raggio di decodifica dei codici LDPC non può essere determinato analiticamente, sebbene diverse ricerche siano state condotte in questo senso [133, 134]; inoltre le prestazioni del decoder BF sono influenzate anche dalla scelta della soglia. Per questo la capacità correttiva viene solitamente valuta-

ta mediante simulazioni numeriche; nondimeno esistono dei modelli asintotici che consentono di predire il comportamento del decoder con una buona affidabilità, determinando limiti inferiori e superiori della capacità correttiva del codice [135, 136, 137, 138, 139, 140].

4 Strutture che inducono a fallimenti di decodifica in codici LDPC

4.1 Definizioni preliminari

Si consideri un grafo *non orientato*, cioè un grafo i cui collegamenti tra i nodi non sono direzionati, $\mathcal{G} = (\mathcal{F}, \mathcal{E})$; i due insiemi $\mathcal{F} = \{f_1, ..., f_k\}$ e $\mathcal{E} = \{e_1, ..., e_m\}$ sono rispettivamente i *nodi* e i *rami* di $\mathcal{G}$. Un ramo e è detto *incidente* ad un nodo f se e è connesso con f. Se esiste un ramo e_k incidente a due nodi distinti f_i ed f_j, tali nodi sono detti *adiacenti*, e si dice che f_i è un *vicino* di f_j e viceversa; in tal caso e_k può essere rappresentato dalla notazione $f_i f_j$ o $f_j f_i$. L'*intorno* di un nodo f, indicato con $\mathcal{N}(f)$, è l'insieme dei nodi adiacenti a f. Il *grado* di un nodo f è indicato con $deg(f)$, ed è il numero di rami incidenti a f. Il *grado massimo* e il *grado minimo* di un grafo $\mathcal{G}$, indicati con $\Delta(\mathcal{G})$ e $\delta(\mathcal{G})$, indicano il grado massimo e minimo, rispettivamente.

Dato un grafo non orientato $\mathcal{G} = (\mathcal{F}, \mathcal{E})$, un *cammino* tra due nodi f_1 e f_{k+1} è una sequenza di nodi e rami $f_1, e_1, f_2, e_2, ..., f_k, e_k, f_k + 1$ dove $e_i = f_i f_{i+1}, \forall i \in [1, k]$; in questa definizione, i nodi $f_1, f_2, ..., f_{k+1}$ non sono necessariamente distinti. Se il primo e l'ultimo nodo sono distinti, allora si tratta di un *percorso aperto*, altrimenti si parla di *ciclo*. La *lunghezza* di un cammino, un percorso, o un ciclo corrisponde al numero di rami che lo compongono. La *corda* di un ciclo è un ramo che non fa parte del ciclo ma è incidente a due nodi distinti appartenenti ad esso. Un *ciclo semplice* è un ciclo che non ha nessuna corda.

Un grafo è detto *connesso* quando esiste un percorso tra ogni coppia di nodi. Un *albero* è un grafo connesso che non contiene cicli. Un *albero con radice* è un albero contenente uno specifico nodo chiamato *radice*, ovvero un nodo connesso a tutti gli altri. La *profondità di nodo* in riferimento ad un nodo in un albero con radice è la lunghezza del percorso che va dalla radice a tale nodo. La *profondità di albero* è la profondità del nodo che ha profondità massima. Un *albero a profondità unitaria* è un albero la cui profondità è pari a uno. Un nodo f è detto *foglia* se $deg(f) = 1$; un grafo connesso $\mathcal{G}$ è detto *senza foglie* o *leafless* se $\delta(\mathcal{G}) \geq 2$.

Due grafi $\mathcal{G}_1 = (\mathcal{F}_1, \mathcal{E}_1)$ e $\mathcal{G}_2 = (\mathcal{F}_2, \mathcal{E}_2)$ sono *isomorfi* se esiste una corrispondenza biunivoca $p : \mathcal{F}_1 \to \mathcal{F}_2$ tale che due nodi $f_1, f_2 \in \mathcal{F}_1$ sono collegati da un ramo se e solo se $p(f_1)$ e $p(f_2)$ sono collegati da un ramo; in caso contrario, i grafi sono *non isomorfi*.

Ogni matrice di parità $\mathbf{H}$ di un codice LDPC C(n, k), di dimensione $(n-k) \times n$, può essere rappresentata dal suo grafo di Tanner $\mathcal{G} = (\mathcal{V} \cup \mathcal{C}, \mathcal{E})$, dove $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ è l'insieme dei nodi variabile e $\mathcal{C} = \{c_1, c_2, ..., c_r\}$ (con $r = n-k$) è l'insieme dei nodi check. Il grado di un nodo variabile v_i è indicato con d_{v_i}, mentre il grado di un nodo check c_i è indicato con d_{c_i}. Un grafo di Tanner è detto *variabile-regolare* con grado d_v se $d_{v_i} = d_v, \forall\ v_i \in \mathcal{V}$. Un grafo di Tanner (d_v, d_c)-regolare è un grafo variabile-regolare in cui $d_{c_i} = d_c, \forall\ c_i \in \mathcal{C}$. Dato un sottoinsieme $\mathcal{S} \subset \mathcal{V}$, il sottoinsieme $\Gamma(\mathcal{S}) \subset \mathcal{C}$ indica l'insieme dei nodi check vicini dei nodi in $\mathcal{S}$, all'interno di $\mathcal{G}$. Il *sottografo indotto* da $\mathcal{S} \subset \mathcal{G}$, indicato con $\mathcal{G}(\mathcal{S})$, è un grafo in cui l'insieme dei nodi è $\mathcal{S} \cup \Gamma(\mathcal{S})$, mentre l'insieme dei rami è $\{f_i f_j \in \mathcal{E} : f_i \in \mathcal{S}, f_j \in \Gamma(\mathcal{S})\}$. L'insieme dei nodi check con grado pari (detti anche nodi *soddisfatti*) in $\mathcal{G}(\mathcal{S})$ è indicato con $\Gamma_e(\mathcal{S})$, mentre quello dei nodi check di grado dispari (*non soddisfatti*) è indicato con $\Gamma_o(\mathcal{S})$. La *dimensione* di un sottografo indotto è data dal numero dei suoi nodi variabile. La lunghezza del ciclo più corto in un grafo di Tanner è chiamata *girth*, ed è indicata con g.

4.2 Stopping set in canali BEC

Come già affermato nella Sezione 3, esistono delle ben note tecniche di analisi della capacità correttiva dei codici LDPC considerando diversi tipi di decodifica. Queste analisi sono principalmente svolte a ridosso del limite asintotico della lunghezza del codice ($n \to \infty$) [141, 142, 143]; tuttavia tali analisi presentano diverse limitazioni quando vengono applicate a codici di lunghezza finita.

L'analisi a lunghezza finita dei codici LDPC è stata storicamente particolarmente proficua innanzitutto riguardo alla decodifica sul *Binary Erasure Channel* (BEC), di cui uno schema è rappresentato in Figura 2: a differenza del canale BSC, nel BEC ciascun bit ha una certa probabilità di erasure (o cancellazione) p_e, che fa sì che alcuni bit arrivino al decoder in uno stato indeterminato.

Un semplice esempio di decodifica iterativa sul BEC è rappresentato dall'algoritmo *Edge Removal* (ER) [144], applicato nel *Peeling decoder*:

Definizione 1. *Sia* $\mathbf{c} \in \mathrm{C} \subseteq \{0,1\}^n$ *una parola di un codice binario trasmessa sul BEC, e sia* $\mathbf{v} \in \{0,1,?\}^n$ *il vettore ricevuto, affetto da cancellazioni. L'algoritmo ER procede come segue:*

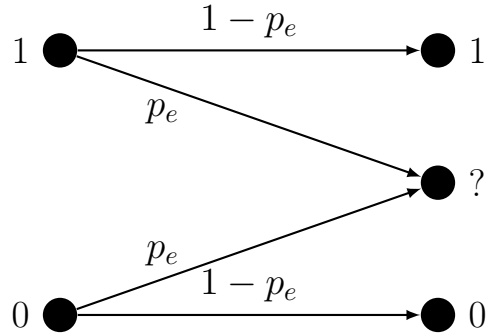


Figura 2: Schema di canale BEC con probabilità di erasure p_e.

1. *Step iniziale: Il valore di ciascun bit ricevuto v_i viene assegnato all'i-esimo nodo variabile del grafo di Tanner;*

2. *i nodi check $c_i \in \mathcal{C}$ contano il numero dei vicini nodi variabile affetti da cancellazione;*

3. *se il nodo check c_i ha un solo vicino v_j affetto da cancellazione, il valore del j-esimo bit è univocamente determinato dall'equazione di parità che coinvolge c_i e tutti i suoi vicini nodi variabile (cioè, se oltre al nodo affetto da cancellazione v_j, c_i ha come vicini un numero pari di nodi con valore 1, allora il valore di v_j sarà 0, e sarà 1 se il numero di nodi con valore 1 è dispari);*

4. *gli step 2 e 3 vengono ripetuti finché non sono state eliminate tutte le cancellazioni o finché non ci sono più nodi check con meno di due vicini affetti da cancellazione; in quest'ultimo caso il decoder fallisce a causa di uno stopping set (vedi sotto).*

Un esempio della procedura di decodifica ER è illustrato in Figura 3.

In [145], Di et al. utilizzano per la prima volta una caratterizzazione combinatoria del fallimento della decodifica di codici LDPC su canali BEC, che sfrutta un oggetto, definito sul grafo di Tanner, denominato *stopping set*:

Definizione 2. *Uno stopping set $\mathcal{S} \subset \mathcal{V}$ è un insieme di nodi variabile che forma un sottografo indotto $\mathcal{G}(\mathcal{S})$, tale che $\delta(\mathcal{G}) \geq 2$.*

L'insieme vuoto è anch'esso uno stopping set; lo spazio degli stopping set è chiuso nell'unione: se $\mathcal{S}_1$ e $\mathcal{S}_2$ sono due stopping set, lo sarà anche $\mathcal{S}_1 \cup \mathcal{S}_2$. Il ruolo cruciale di questi oggetti nella decodifica iterativa di codici LDPC su canali BEC viene descritto dal seguente Lemma [145]:

a)

b)

c)

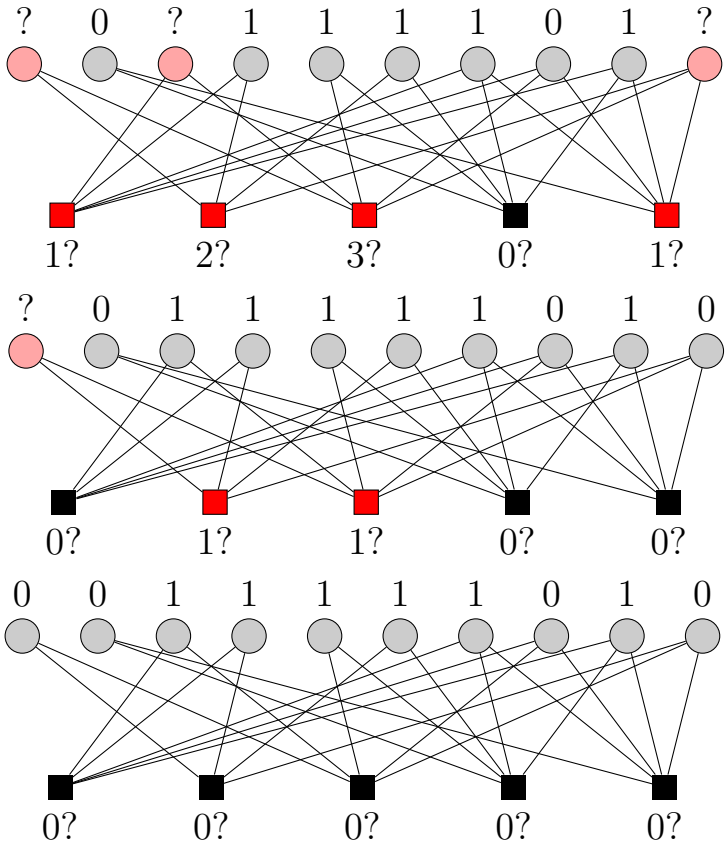


Figura 3: Procedura iterativa di decodifica ER sul grafo di Tanner di un codice $(10, 5)$ irregolare (relativo alla matrice di parità dell'Esempio 2). In (a) vengono evidenziati gli step 1 e 2 dell'algoritmo EF: il vettore ricevuto $\mathbf{v} = [?, 0, ?, 1, 1, 1, 1, 0, 1, ?]$ viene assegnato ai nodi variabile, e i nodi check effettuano il conteggio dello step 2. In (b), dopo aver determinato il valore di due dei bit cancellati, viene effettuato nuovamente il conteggio dello step 2, per una seconda iterazione di decodifica. In (c) la decodifica termina (step 4).

Lemma 1. *Sia $\mathcal{G}(\mathcal{S})$ il sottografo indotto da un insieme $\mathcal{S}$ di nodi variabile, di cui $E \subset \mathcal{S}$ sono "cancellati" dal canale. Alla fine della decodifica iterativa, cioè quando viene recuperata la parola di codice o quando il decoder non riesce più ad eliminare cancellazioni (a prescindere dal numero di iterazioni), il numero di cancellazioni che rimangono corrisponde al massimo stopping set di E.*

Esempio 2. *Sia* $\mathbf{H}$ *la seguente matrice di parità di un codice* $(10, 5)$*, irrego-*

lare:

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

L'insieme $\{v_7, v_9\}$ costituisce uno stopping set, evidenziato nel grafo di Tanner a sinistra (e nelle colonne 7 e 9 nella matrice sopra); a destra, invece, il sottografo indotto dallo stopping set:

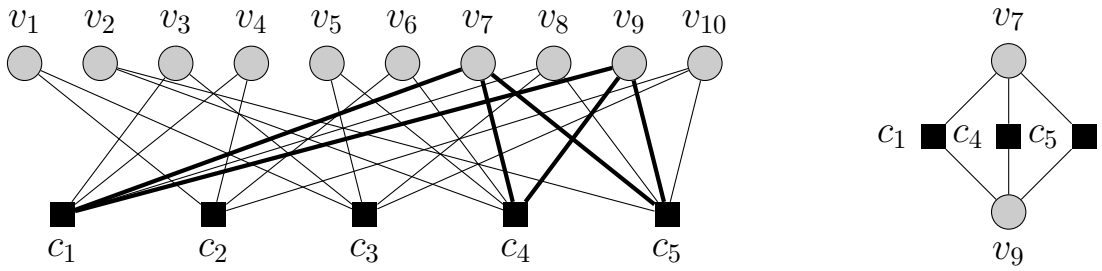


Figura 4: Esempio di stopping set.

In Figura 5 è illustrato un esempio di fallimento di decodifica ER a causa di uno stopping set. Partendo dal Lemma 1 è "sufficiente" studiare la probabilità che un insieme random di cancellazioni causate dal canale contenga uno stopping set, e questo può essere fatto in maniera esatta. Più precisamente, la capacità correttiva del codice è determinata dall'esistenza di uno stopping set, e dalla dimensione dello stopping set più grande, nei nodi affetti da cancellazione. In [145] viene derivata una formula ricorsiva per calcolare la probabilità di errore media di codici LDPC regolari; in [146] tale risultato viene esteso e semplificato. Poiché gli stopping set hanno una caratterizzazione combinatoria, la loro distribuzione nel grafo di Tanner può essere analizzata rigorosamente [147].

Un parametro fondamentale nelle analisi delle prestazioni dei codici LDPC in canali BEC è la cosiddetta *stopping distance* o *stopping number*:

Definizione 3. *La stopping distance (o stopping number) $s(\mathbf{H})$ è la dimensione del più piccolo stopping set non vuoto nella matrice di parità di un codice.*

Questa gioca un ruolo importante nella comprensione delle prestazioni di un codice con decodifica iterativa sul canale BEC, similarmente a quanto fa

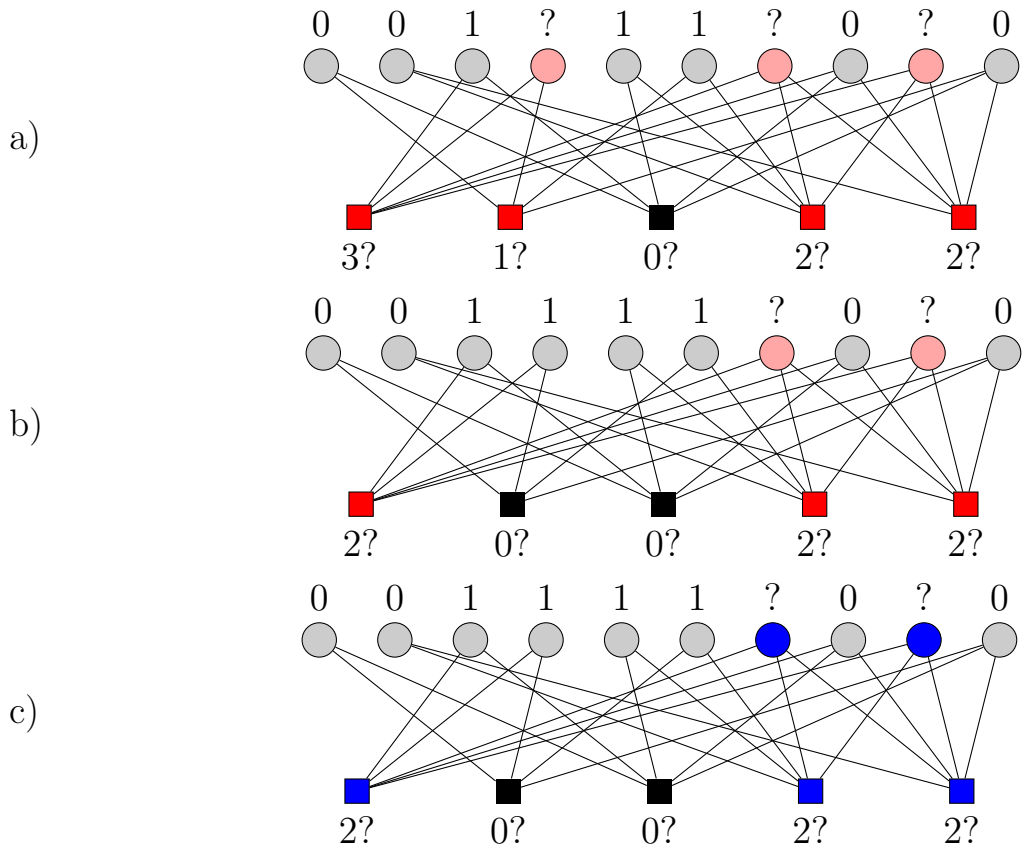


Figura 5: Fallimento di decodifica iterativa ER sul grafo di Tanner di un codice $(10, 5)$ irregolare (relativo alla matrice di parità dell'Esempio 2). In (a) il vettore ricevuto $\mathbf{v} = [0, 0, 1, ?, 1, 1, ?, 0, ?, 0]$ viene assegnato ai nodi variabile, e i nodi check effettuano il conteggio dello step 2. In (b), dopo aver determinato il valore di un bit cancellato, viene effettuato nuovamente il conteggio dello step 2, per una seconda iterazione di decodifica. In (c) la decodifica termina con un fallimento (step 4), convergendo in uno stopping set.

la *distanza minima* di Hamming d per quanto riguarda i codici algebrici: proprio come si vorrebbe normalmente avere la distanza minima di Hamming più grande possibile, lo stesso vale per la stopping distance. La differenza tra le due è che, mentre la distanza minima di Hamming dipende dal codice (per questo si preferisce indicare $d = d(\mathrm{C})$), la stopping distance dipende esclusivamente dal grafo di Tanner relativo alla matrice di parità del codice (e per questo si indica $s = s(\mathbf{H})$). Ne segue che aggiungere righe linearmente dipendenti alla matrice $\mathbf{H}$ può aumentare la stopping distance. In generale,

infatti, vale la seguente, frutto di semplici considerazioni combinatorie:

$$s(\mathbf{H}) \leq d(\mathrm{C}) \tag{15}$$

per cui la stopping distance costituisce un limite inferiore alla distanza minima del codice.

Il numero minimo $r_m(\mathrm{C})$ di righe della matrice di parità tale che $s(\mathbf{H}) = d(\mathrm{C})$ è chiamato *stopping redundancy* [148]. In alcuni casi la stopping distance aumenta linearmente col numero di nodi variabile del grafo di Tanner; per questo si definisce un'ulteriore grandezza, chiamata *stopping ratio*:

Definizione 4. *Sia $\mathcal{G}$ un grafo di Tanner con n nodi variabile e stopping distance s. Lo stopping ratio è definito come:*

$$\rho = \frac{s}{n}. \tag{16}$$

Le principali soluzioni al problema degli stopping set consistono nel costruire codici LDPC in modo da evitare la creazione di stopping set piccoli nel grafo di Tanner [149, 150, 151].

4.3 Trapping set in canali BSC e AWGN

I trapping set, proprio come gli stopping set, sono definiti da insiemi di nodi variabile e nodi check che ostacolano la corretta decodifica da parte del decoder di codici LDPC. Storicamente la definizione di trapping set è successiva a quella di stopping set, ed è comunemente accettata come volontariamente generica: nel nome di trapping set si annidano infatti diversi tipi di strutture, tanto che si potrebbe assumere che lo stopping set sia un particolare tipo di trapping set, piuttosto che un analogo, per il BEC. Generalmente, la nozione di trapping set identifica le strutture "error-prone" sul grafo di Tanner su canali BSC o AWGN. Di seguito si farà riferimento esclusivamente al BSC, anche considerando che esso è d'interesse per quanto riguarda la crittografia basata su codici. Come già illustrato nella Sezione 3, nel BSC un nodo variabile è corretto se il suo valore ricevuto corrisponde a quello trasmesso, altrimenti si dice *corrotto*, o affetto da errore. Senza perdita di generalità, stante la linearità del codice, si può assumere la trasmissione della parola nulla: in altri termini, si considera la sequenza trasmessa composta solo da zeri (parola di

codice nulla), di modo che i soli nodi variabile pari a 1 siano quelli affetti da errore; questa assunzione può essere adottata soltanto quando l'algoritmo di decodifica soddisfa le condizioni di simmetria descritte in [141], il che vale per tutti gli algoritmi di decodifica iterativi (compreso BF) [152].

Definizione 5. *Sia* $\mathbf{y} = (y_1, ..., y_n)$ *la sequenza binaria in ingresso al decoder, e sia* $\mathbf{x}^l = (x_1^l, ..., x_n^l)$ *l'output del decoder alla l-esima iterazione, con* $l \leq i_{max}$*. Un nodo variabile* v *è detto "eventualmente corretto" se esiste* $q \in \mathbb{N}$ *tale che* $\forall\ l \geq q$*,* $v \notin S(\mathbf{x}_l)$*. Un decoder fallisce se non esiste* $l \leq i_{max}$ *tale che* $S(\mathbf{x}^l) = \emptyset$*.*

Definizione 6. *Sia* $\mathbf{T}(\mathbf{y})$ *l'insieme dei nodi variabile* ***non*** *eventualmente corretti. Se* $\mathbf{T}(\mathbf{y}) \neq \emptyset$*, sia* $a = |\mathbf{T}(\mathbf{y})|$ *(per* $|.|$ *si intende il numero di elementi non nulli dell'insieme) e b il numero di nodi check non soddisfatti appartenenti al sottografo indotto da* $\mathbf{T}(\mathbf{y})$*;* $\mathbf{T}(\mathbf{y})$ *è un trapping set appartenente alla classe* (a, b)*.*

Si noti che $\mathbf{T}(\mathbf{y})$ rappresenta lo stato finale del decoder; si dice infatti che il decoder "converge" nel trapping set, partendo però da un pattern di errore iniziale $\mathbf{y}$ (o, equivalentemente, il suo supporto $S(\mathbf{y})$) che contiene $\mathbf{T}(\mathbf{y})$ o un suo sottoinsieme.

Definizione 7. *Sia* $\mathcal{T}$ *un trapping set. Se* $\mathbf{T}(\mathbf{y}) = \mathcal{T}$*, allora* $S(\mathbf{y})$ *è un inducing set di* $\mathcal{T}$*.*

Un *inducing set* è quindi un pattern di errore che porta il decoder a convergere in un trapping set.

Definizione 8. *Sia* $\mathcal{T}$ *un trapping set. Il numero critico* $\mu(\mathcal{T})$ *di un trapping set* $\mathcal{T}$ *è la dimensione del suo più piccolo inducing set.*

Definizione 9. *La forza* $\nu(\mathcal{T})$ *di un trapping set* $\mathcal{T}$ *è il numero degli inducing set di dimensione* $\mu(\mathcal{T})$*.*

L'effetto di un trapping set sul DFR è quindi determinato dal suo numero critico e dalla sua forza: una volta noti tutti i trapping set presenti nel grafo di Tanner di un codice, assieme al loro numero critico e alla loro forza, è possibile determinare analiticamente il DFR nella regione di error floor tramite la seguente formula [153], che vale per il canale BSC:

$$\mathrm{DFR} = \sum_{\mu} \sum_{\nu} \nu \Gamma_{\mu,\nu} \alpha^{\mu} (1-\alpha)^{(n-\mu)} \tag{17}$$

dove:

- α è la probabilità di crossover del canale BSC;
- $\Gamma_{\mu,\nu}$ è il numero di tutti i trapping set con numero critico μ e forza ν;
- n è la lunghezza del codice.

Il comportamento della curva di DFR per valori bassi di α è dominato da $\log(\text{DFR}) \approx \mu_{\min} \log(\alpha) + \log(\sum_{\nu} \Gamma_{\mu_{\min},\nu})$, che graficamente è "quasi" una linea retta con una pendenza pari a μ_{min}. È evidente quindi che il trapping set con numero critico più basso è quello dominante nella curva di error floor. Anche se la (17) è apparentemente calcolabile con estrema semplicità, determinare l'esatto numero critico e la forza di ogni trapping set presente in un grafo di Tanner non è affatto banale.

In Figura 6 è mostrato il sottografo indotto da un trapping set $(4, 4)$, $\mathcal{T} = \{v_1, v_2, v_3, v_4\}$ in un LDPC con peso di colonna pari a 3. Si può osservare come l'insieme dei nodi variabile del trapping set siano coinvolti in un ciclo da otto. Si noti che i nodi check soddisfatti sono colorati in bianco, mentre quelli non soddisfatti in nero. Il numero critico del trapping set è 4 se si considerano gli algoritmi di decodifica Gallager A/B [132], cioè tutti e quattro i nodi variabile di $\mathcal{T}$ devono essere affetti da errore (in altre parole, $\{v_1, v_2, v_3, v_4\} \subseteq \mathbf{y}$); se invece si considera l'algoritmo BF, si può dimostrare che $\{v_1, v_3\}$ e $\{v_2, v_4\}$ sono entrambi inducing set di $\mathcal{T}$, con il risultato che il numero critico, in quest'ultimo caso, è pari a 2. Si noti che l'insieme $\{v_1, v_2, v_3, v_4\}$ costituisce un inducing set anche per il decoder BF; tuttavia, ciò non risulta essere particolarmente rilevante, in quanto esistono inducing set più piccoli.

Sulla destra, sempre in Figura 6, viene illustrata una rappresentazione grafica alternativa del trapping set, basata sulla struttura di incidenza di linee e punti. Una struttura di incidenza è definita da una terna (P, L, I), dove P è l'insieme dei punti, L è l'insieme delle linee, e $I \subseteq P \times L$ è la relazione d'incidenza. Gli elementi di I sono chiamati *flags*. Se $(p, l) \in I$, si dice che il punto p "poggia" sulla linea l. In questa rappresentazione, i nodi variabile corrispondono a linee, e i nodi check corrispondono a punti. Un punto è colorato in nero se ha un numero dispari di linee che vi incidono, altrimenti è colorato in bianco. Un trapping set della classe (a, b) è dunque una struttura di incidenza con a linee e b punti colorati in nero. Mediante questa rappresentazione è possibile applicare le tecniche di *colorazione di grafi*, che sono ben note in letteratura [154], al problema dei trapping set, al fine di creare una lista di possibili candidati sfruttando le relazioni topologiche tra le diverse strutture [155].

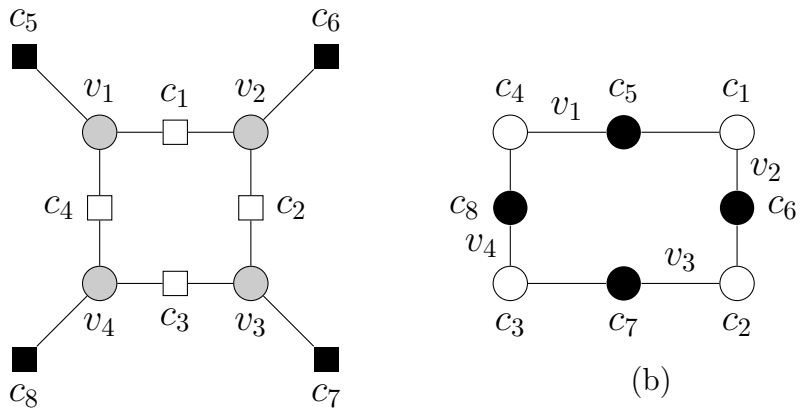


Figura 6: Rappresentazione di un trapping set di classe $(4, 4)$; (a) nel grafo di Tanner, (b) nella rappresentazione tramite struttura di incidenza.

In Figura 7 è rappresentato un trapping set della classe $(5, 3)$ che è l'unione di tre cicli da otto. Il suo numero critico è 3, dato che l'inducing set più piccolo è $\{v_2, v_4, v_5\}$. Come si evince da un primo sguardo ai trapping set di Figura 6 e Figura 7, esiste un qualche tipo di relazione tra i due: il trapping set $(5, 3)$ può essere infatti ottenuto aggiungendo al $(4, 4)$ il nodo variabile v_5, tale che $\mathcal{N}(v_5) = \{c_5, c_7, c_9\}$; in tal modo, c_5 e c_7, che già appartengono al $(4, 4)$, vengono soddisfatti, e al contempo c_9, che non appartiene al $(4, 4)$, viene aggiunto al trapping set come nodo non soddisfatto. Esiste quindi una sorta di relazione genitore-figlio tra i due, basata sulla proprietà di isomorfismo dei sottografi indotti:

Definizione 10. *Siano $\mathcal{T}_1$ e $\mathcal{T}_2$ due trapping set, che generano due sottografi indotti $\mathcal{G}(\mathcal{T}_1)$ e $\mathcal{G}(\mathcal{T}_2)$ rispettivamente. Allora, $\mathcal{T}_1$ è genitore di $\mathcal{T}_2$ (o, analogamente, $\mathcal{T}_2$ è figlio di $\mathcal{T}_1$) se $\mathcal{T}_2$ contiene un sottoinsieme il cui sottografo indotto è isomorfo a $\mathcal{G}(\mathcal{T}_1)$.*

Dal punto di vista della struttura d'incidenza, viene semplicemente aggiunta una linea passante per c_5 e c_7, che in accordo con quanto detto sopra, cambiano colore, mentre c_9 viene aggiunto colorato di nero, poiché per esso passa una sola linea. Questo suggerisce un possibile vincolo per la generazione di strutture figlie partendo da un genitore: le nuove linee devono passare soltanto attraverso i nodi neri, con l'obiettivo di ridurne il numero; questo vincolo viene adottato sia perché permette di ridurre la complessità della ricerca di strutture figlie, sia perché le strutture così ottenute costituiscono quelle più pericolose nei codici LDPC, ovvero gli *Elementary Trapping Set* [156]:

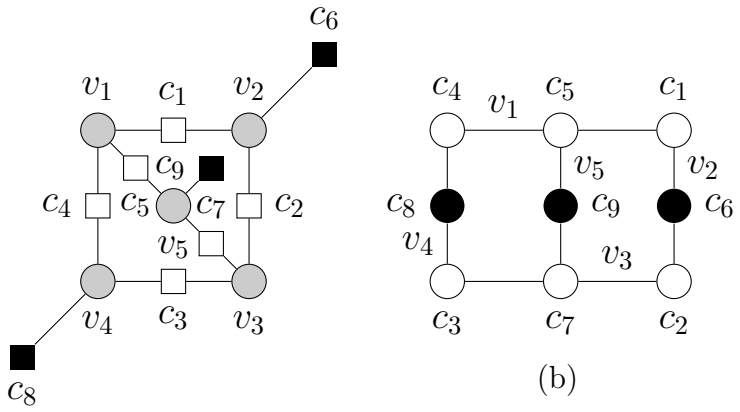


Figura 7: Rappresentazione di un trapping set di classe $(5, 3)$; (a) nel grafo di Tanner, (b) nella rappresentazione tramite struttura d'incidenza.

Definizione 11. *Un trapping set $\mathcal{T}$ è detto Elementary Trapping Set (ETS) se contiene solo nodi check di grado 1 o 2 (ovviamente quelli di grado 1 sono non-soddisfatti, quelli di grado 2 sono soddisfatti).*

Un ulteriore vincolo da porre per la ricerca di strutture figlie è che il *girth* del grafo genitore deve essere preservato.

Si noti che il trapping set $(4, 4)$ di Figura 6 è un ciclo da 8 in un codice regolare con grado 3. Il $(5, 3)$ di Figura 7 è figlio di prima generazione del precedente, mentre in Figura 8 si vedono due strutture a loro volta figlie del $(5, 3)$ menzionato prima; anche i due $(6, 4)$ di Figura 9 sono figlie del $(5, 3)$, il quale è l'unione di tre cicli da 8, il $(6, 4)$ (a) è l'unione di un ciclo da 8 e uno da 10, mentre il $(6, 4)$ (b) è l'unione di due cicli da 8.

La pericolosità di un trapping set $\mathcal{T}$ dipende dal suo numero critico $\mu(\mathcal{T})$ e dalla sua forza ν. Più è piccolo il numero critico, più è pericoloso un trapping set, poiché sono necessari meno errori per "attivarlo"; allo stesso modo, più è grande la sua forza e più il trapping set è pericoloso, in quanto esistono più pattern di errore che possono portare il decoder a convergere sul trapping set. Generalmente, al progredire delle generazioni i trapping set diventano più pericolosi, in quanto l'aggiunta di nodi variabile al sottografo indotto può introdurre nuovi inducing set, eventualmente anche più piccoli.

Dati quindi un codice LDPC e un algoritmo di decodifica, è possibile definire una *ontologia* dei trapping set [155], ossia una sorta di albero genealogico di strutture pericolose, che parte dai cicli semplici di diverse lunghezze che vengono espanse di volta in volta, generando diverse generazioni successive.

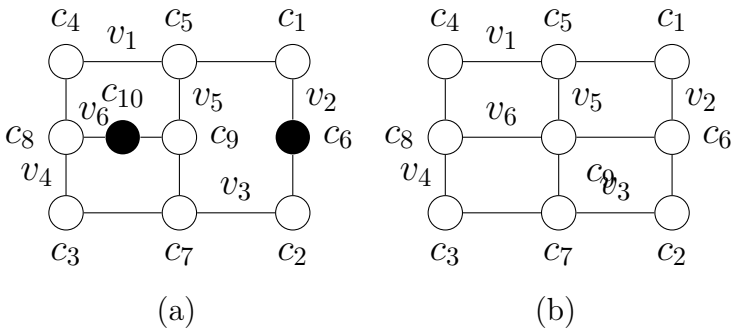


Figura 8: Due strutture figlie del trapping set $(5, 3)$ di Figura 7: nel primo caso, in (a), il nodo v_6 è tale che $\mathcal{N}(v_6) = \{c_8, c_9, c_{10}\}$ e origina una trapping set $(6, 2)$; in (b) si ha che $\mathcal{N}(v_6) = \{c_6, c_8, c_9\}$, e l'aggiunta di tale nodo origina un trapping set $(6, 0)$. Si noti che, in quest'ultimo caso, il trapping set corrisponde ad una parola di codice.

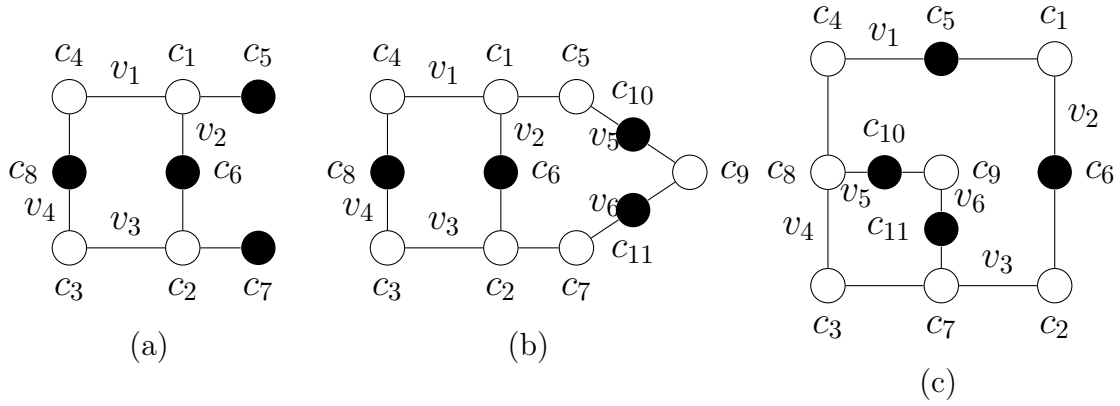


Figura 9: In (a) una rappresentazione isomorfa a quella di Figura 6, utile per raffigurare un'ulteriore struttura sua figlia (b); in (c) un altra struttura figlia del trapping set di Figura 6. In questo caso, si dice che (b) e (c) appartengono alle stessa generazione; in particolare, (b) e (c) appartengono alla prima generazione, in quanto ottenute entrambe aggiungendo un solo nodo alla stessa struttura di partenza.

Nelle Figure 6-9 sono illustrati alcuni elementi dell'ontologia per un codice LDPC regolare di grado 3 e decodifica con algoritmo Gallager A.

Sebbene la creazione di un database di trapping set, sfruttando le relazioni genitore-figlio di cui sopra, presenti diverse limitazioni (tale tecnica è stata infatti impiegata soltanto per codici regolari di grado 3, e non consente una caratterizzazione esaustiva), il principio di costruire trapping set partendo da cicli semplici è stato fondamentale nella ricerca in tale ambito. L'ontologia dei trapping set può essere usata per semplificare la loro enumerazione sfruttando le relazioni di parentela piuttosto che procedere con un approccio a forza bruta. Su questo presupposto su basano gran parte degli algoritmi di ricerca di trapping set, oltre che gran parte delle loro caratterizzazioni [157, 158, 159, 160, 161]. Il primo passo, per questi algoritmi, è l'enumerazione dei cicli di lunghezza g, ossia i più pericolosi; questi sono considerati trapping set di generazione zero, e vengono espansi tenendo conto delle possibili strutture presenti nella prima generazione, e così via fino ad arrivare a trapping set della dimensione desiderata.

Particolarmente rilevante è il lavoro di Banihashemi et al. [156, 157, 159], che hanno studiato a fondo gli ETS, in particolare i *Leafless* ETS (LETS):

Definizione 12. *Un Elementary Trapping Set $\mathcal{T}$ è detto leafless se forma un sottografo indotto $\mathcal{G}(\mathcal{T})$ tale che $\delta(\mathcal{G}) \geq 2$.*

In particolare, in [159], viene illustrata la proprietà *Layered Superset* (LSS):

Definizione 13. *Sia $\mathcal{T}$ un ETS (a, b). Sia $\mathcal{I} \subset \mathcal{T}$ un ETS di dimensione $\alpha < a$. $\mathcal{T}$ è un Layered Superset (LSS) di $\mathcal{I}$ se esiste una sequenza di ETS tale che: $\mathcal{I} := \mathcal{T}^{(0)} \subset \mathcal{T}^{(1)} \subset ... \subset \mathcal{T}^{(a-\alpha)} := \mathcal{T}$, tale che $\mathcal{T}^{(i)} \in \mathcal{T}$ ha dimensione $\alpha + i$ per $i = 0, ..., a - \alpha$.*

In Figura 10 a sinistra è illustrato un ETS $\mathcal{T} = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ su un codice LDPC regolare con grado 4. Si consideri il sottoinsieme $\mathcal{I}_1 = \{v_2, v_2, v_3\}$. È facile verificare che $\mathcal{T}$ e $\mathcal{I}_1$ sono due trapping set $(6, 6)$ e $(3, 6)$, rispettivamente. Si può verificare, anche graficamente, che $\mathcal{T}$ non è LSS di $\mathcal{I}_1$: partendo da $\mathcal{I}_1$ è possibile ottenere $\mathcal{I}_2 = \mathcal{I}_1 \cup \{v_4\}$, che è un suo LSS; tuttavia, aggiungendo a $\mathcal{I}_2$ v_5 o v_6, non si aggiunge alcun ciclo alla struttura di partenza, ovvero non si crea un nuovo ETS (per chiarezza, se in ingresso al decoder si ha $\mathbf{y} = \{\mathcal{I}_2 \cup v_5\}$, questo convergerà comunque su $\mathcal{I}_2$). Per ottenere $\mathcal{T}$ si dovrebbero aggiungere v_5 e v_6 contemporaneamente, violando però in tal modo la definizione; tuttavia, $\mathcal{T}$ è LSS di $\mathcal{I}_3 = \{v_3, v_4, v_5, v_6\}$. Sempre in Figura 10, a destra, si può osservare un ETS $(7, 1)$ in un codice regolare

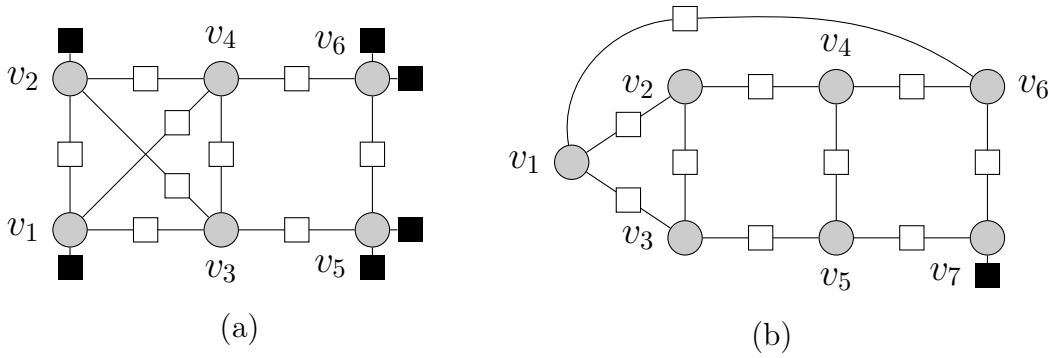


Figura 10: In (a) un ETS $(6, 6)$ in un grafo regolare con grado 4; in (b) un ETS $(7, 1)$ in un grafo regolare con grado 3.

di grado 3. In questo caso il trapping set è un LSS di $\mathcal{I}_1 = \{v_2, v_3, v_4, v_5\}$ mediante la sequenza: $\mathcal{I}_1 \subset \{v_1, v_2, v_3, v_4, v_5\} \subset \{v_1, v_2, v_3, v_4, v_5, v_6\} \subset \mathcal{T}$.

La proprietà LSS viene sfruttata per sviluppare una semplice tecnica di espansione del grafo, chiamata *depth-one-tree* (*dot*). In pratica, espandendo i cicli semplici di diverse lunghezze mediante l'espansione *dot*, si ottengono tutti i successivi LSS. In tal modo si costruisce una sorta di ontologia, strutturata come una foresta (ossia un insieme di grafi ad albero), in cui ogni albero ha come radice un ciclo di diversa lunghezza, dalla quale si diramano le varie generazioni di LETS che costituiscono un LSS del ciclo. Le strutture non isomorfe da ricercare vengono caratterizzate mediante il software *Nauty* [162]. La notazione dot_m viene utilizzata per indicare un espansione *dot* con m rami, come descritto in Figura 11. Si noti che in tale figura viene utilizzata un'ulteriore rappresentazione grafica, alternativa al grafo di Tanner e alle strutture d'incidenza illustrate sopra. Si tratta del cosiddetto *grafo normale*, e si ottiene a partire dal grafo di Tanner, eliminando i nodi check di grado 1, e rappresentando quelli di grado 2 con una linea che, evidentemente, collega i due nodi variabile vicini. È facile verificare che, anche in questo caso, esiste una corrispondenza biunivoca tra le due rappresentazioni, che sono quindi equivalenti; tuttavia, il grafo normale permette di semplificare la trattazione quando si parla di ETS.

Esempio 3. *Si consideri il LETS di Figura 11a in un grafo di Tanner regolare con grado 4 e* $g = 6$. *La struttura appartiene alla classe* $(4, 8)$. *L'applicazione dell'espansione* dot_2 *genera due strutture non isomorfe, illustrate in Figure 11b-11c. In Figura 11d è illustrata l'unica struttura non isomorfa ottenuta applicando un'espansione* dot_3 *al* $(4, 8)$ *originale, mentre in Figura 11e è rappresentata l'unica struttura ottenibile da una* dot_4. *Nelle figure, i nodi bianchi*

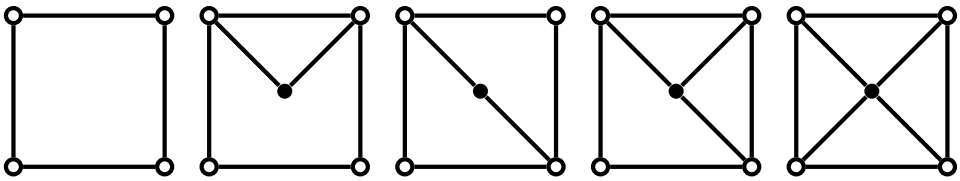


Figura 11: Struttura LETS $(4, 8)$ e sue strutture LSS, ottenute mediante espansioni dot_2, dot_3 e dot_4.

rappresentano i nodi in comune tra la struttura genitore e le sue figlie, mentre i nodi neri sono i nodi aggiunti dall'espansione.

Tuttavia, in [156], gli stessi autori evidenziano l'esistenza di alcune strutture chiamate *prime*, che non possono essere ottenute tramite espansioni *dot*. Pertanto vengono introdotte ulteriori espansioni, leggermente più complesse (chiamate espansioni *path* e *lollipop*), che aggiungono più nodi alla volta; in tal modo si riesce ad ottenere esaustività (oltre che una maggiore efficienza) nell'algoritmo di ricerca di LETS, che viene appunto denominato *Dot-Path-Lollipop* (DPL).

4.4 Absorbing Set e Fully Absorbing Set

In [163] gli autori hanno isolato un insieme di strutture nel grafo di Tanner che davano luogo alla regione di error floor dei codici LDPC analizzati. Tali strutture sono state denominate *absorbing set*, e possono essere viste come una particolare classe di trapping set. Proprio come gli stopping set, gli absorbing set hanno una definizione puramente combinatoria.

In [164] gli absorbing set vengono descritti come i contributori dominanti alla regione di error floor; il loro studio è quindi fondamentale sia dal punto di vista teorico che dal punto di vista pratico, quando si tratta di progettare codici per evitare la loro formazione e decoder in grado di gestirli. Gli absorbing set sono inoltre una componente fondamentale nella stima dell'error floor basata su *importance-sampling* [165], e nel calcolo dei limiti teorici della capacità correttiva dei codici LDPC [166].

Definizione 14. *Un trapping set $\mathcal{T}$ della classe (a, b) è chiamato Absorbing Set se ogni nodo variabile in $\mathcal{G}(\mathcal{T})$ è connesso a più nodi check in $\Gamma_e(\mathcal{T})$ che in $\Gamma_o(\mathcal{T})$.*

In altre parole, si ha una absorbing set (AS) quando ogni nodo variabile del sottografo indotto è collegato a (strettamente) più nodi check soddisfatti che non soddisfatti. Ad esempio, i trapping set di Figura 6 e Figura 7 sono anche AS (nondimeno, essi sono anche ETS; in tal caso si possono definire *Elementary Absorbing Set* (EAS)); anche l'ETS di Figura 9b è un AS, mentre non lo è l'ETS di Figura 9a, in quanto v_5 e v_6 hanno lo stesso numero di nodi soddisfatti e non soddisfatti.

Il nome "absorbing set" indica la natura stessa di questi oggetti, nel senso che essi si comportano come degli stati stabili del decoder BF: quando quest'ultimo converge in una di queste strutture, ne rimane "assorbito". Un caso particolare di AS è il *Fully Absorbing Set* (FAS):

Definizione 15. *Se un trapping set $\mathcal{T}$ è un Absorbing set, e se ogni nodo in $\mathcal{V} \setminus \mathcal{T}$ è connesso a più nodi check in $\mathcal{C} \setminus \Gamma_o(\mathcal{T})$ che in $\Gamma_o(\mathcal{T})$, allora è un Fully Absorbing Set.*

Si consideri quindi un AS $\mathcal{T}$ e il suo sottografo indotto $\mathcal{G}(\mathcal{T})$; si consideri poi il sottografo complementare a quello indotto $\mathcal{G} \setminus \mathcal{G}(\mathcal{T})$, costituito dal grafo di Tanner del codice ad esclusione dei nodi variabile di $\mathcal{G}(\mathcal{T})$. Se tutti i nodi variabile nel grafo complementare hanno più vicini che non appartengono ai nodi non soddisfatti del trapping set, rispetto a quanti ne appartengono, allora si ha un FAS. È evidente come, in un algoritmo message passing, la caratteristica del FAS è quella di includere nodi check non soddisfatti che sono isolati dal resto del grafo di Tanner, e che quindi non ricevono messaggi significativi; questo rende tale struttura uno stato stabile del decoder.

Questa categoria di trapping set è stata studiata principalmente per quanto concerne codici LDPC strutturati, come i codici *Array LDPC* [167, 163, 164, 167, 168]; con l'ipotesi di un codice strutturato, infatti, si è riuscito in alcuni casi a determinare la dimensione dell'absorbing set più piccolo, che altrimenti costituisce un problema NP-hard [169]. Per la caratterizzazione degli AS e FAS in codici non strutturati, o semplicemente più complessi, è necessario l'impiego di un algoritmo di ricerca. In [170, 171] vengono proposti due algoritmi di ricerca di FAS piccoli, basati sulla tecnica *branch-&-bound* che, per quanto efficienti, diventano rapidamente impraticabili all'aumentare della lunghezza e del peso del codice. Recentemente McMillon et. al [172] hanno sviluppato un algoritmo di ricerca basato sui *coset* e su alcune relazioni tra gli AS e la loro sindrome.

4.5 Pseudo-Codewords e Near-Codewords

Mentre le Pseudo-Codewords sono fenomeni relegati all'ambito della decodifica basata su *Linear Programming* (LP) [173, 174], e quindi non riguardano direttamente questo lavoro di tesi, che vuole focalizzarsi sulla decodifica mediante algoritmi message-passing e, nello specifico, mediante algoritmo BF, le Near-Codewords sono indubbiamente di maggiore interesse.

Definizione 16. *Sia* $\mathbf{H}$ *una matrice di parità di dimensioni* $r \times n$*; sia* $\mathbf{x}$ *un vettore binario di lunghezza* n *e di peso* w*. Sia* $\mathbf{s}$ *la sindrome di peso* v *generata da* $\mathbf{x}$*, tale che*

$$\mathbf{x} \cdot \mathbf{H}^T = \mathbf{s}.$$

Si dice allora che $\mathbf{x}$ *è una* (w, v) *near-codeword [175].*

È evidente, quindi, che le definizioni di near-codeword e di trapping set sono perfettamente coincidenti, e che tali nozioni descrivono lo stesso fenomeno.

5 Trapping set in codici MDPC

In questa sezione verrà illustrato il lavoro svolto e i risultati ottenuti. È evidente che, visto la complessità dell'analisi, l'obiettivo di questa tesi non può essere quello di trarre conclusioni a riguardo dell'argomento proposto, ma è piuttosto quello di illustrare dei primi progressi utili nella definizione di un percorso verso l'analisi a lunghezza finita di codici MDPC. Per questo i risultati verranno accompagnati da considerazioni qualitative ed ipotesi sviluppate nel corso della tesi, che il lettore potrà tenere in considerazione o meno qualora avesse in mente di proseguire lo studio in tale ambito.

Se l'analisi a lunghezza finita dei codici LDPC è notoriamente argomento ostico, la trasposizione di tale analisi ai codici MDPC si preannuncia altrettanto, se non più, difficoltosa; si prenda ad esempio l'algoritmo di ricerca DPL proposto da Banihashemi et al. [156], che è attualmente il più efficiente per quanto riguarda la ricerca di LETS: la complessità per la ricerca di cicli semplici di lunghezza k è $\mathcal{O}(n(d_v d_c)^k)$, mentre quella delle espansioni è $\mathcal{O}(N_{a,b} b^2 d_c^2)$ per quanto riguarda la *dot* e $\mathcal{O}(N_{a,b} b^2 (d_v d_c)^m)$ per quanto riguarda la *path* (per $N_{a,b}$ si intende la molteplicità di strutture di partenza appartenenti alla classe (a, b), mentre m è la lunghezza dei percorsi da aggiungere). È evidente come tali complessità siano proibitive nel contesto degli MDPC, perlomeno quelli utilizzati in crittografia, dove il peso di colonna e di riga sono proporzionali a $\sqrt{n}$, e dove n arriva anche ad essere nell'ordine di 10^4.

Nondimeno, sebbene siano stati in particolar modo studiati codici MDPC quasi ciclici come quelli utilizzati in BIKE ($p = 12323$, $n_0 = 2$, $k_0 = 1$, $d_v = 71$, $d_c = 142$), una caratterizzazione teorica della dimensione del minimo absorbing set sfruttando la geometria della matrice di parità, analogamente a quanto fatto in [164] per codici array, sembra pressoché infattibile: l'estrema lunghezza del codice e l'elevato peso di colonna rendono assai complesso l'approccio combinatorio; inoltre, a complicare ulteriormente tali analisi contribuisce il fatto che nei codici MDPC non viene rispettato il cosiddetto *RC-constraint*: nei codici LDPC strutturati, solitamente, il numero massimo di intersezioni che due colonne della matrice di parità possono avere è uno, evidentemente per evitare la formazione di cicli da 4; è chiaro che questa ipotesi semplifica una eventuale trattazione combinatoria come quella proposta in [164]. Viceversa, nei codici MDPC, tale vincolo viene rimosso, di modo che le colonne della matrice di parità possono intersecarsi anche in più di un elemento. Questo comunque influisce sulla capacità correttiva [134], che

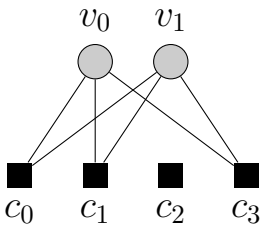


Figura 12: Esempio di grafo di Tanner di due colonne che si intersecano di tre elementi.

però, per l'algoritmo di decodifica scelto, non risente significativamente della massiccia presenza di cicli da 4. Quest'ultima caratteristica non è da sottovalutare: già semplicemente se si volesse applicare un algoritmo di ricerca di trapping set tradizionale si sbaglierebbe probabilmente strada; bisogna infatti chiedersi se, per la costruzione di trapping set mediante espansione, sia più opportuno partire dai cicli semplici, cioè quelli da 4, o dalle coppie di colonne a maggiore intersezione, che costituiscono a loro volta dei cicli che possono essere visti come "sovrapposti". Ad esempio, se due colonne si intersecano in tre elementi, come in Figura 12, si può notare come esistano ben tre cicli da 4 tra di essi. In generale, se due colonne di $\mathbf{H}$ si intersecano di m elementi, tra i corrispondenti nodi variabile si formeranno $\binom{m}{2}$ cicli da 4. In questo senso, è possibile che la densità della matrice di parità dei codici MDPC, la quale introduce giocoforza nuove topologie all'interno del grafo di Tanner, invalidi i risultati ottenuti per i codici LDPC riguardo ai metodi di costruzione dei trapping set partendo da cicli semplici. Nondimeno, dato che si è considerato il caso di codici QC-MDPC utilizzati in BIKE, dove il decoder è una variante del decoder BF, si suppone che le strutture pericolose siano Absorbing Set e Fully Absorbing Set e, pertanto, la ricerca si è concentrata su tali oggetti.

Ad ogni modo, il lavoro può dirsi suddiviso in due parti sostanziali: nella prima verrà analizzato un particolare trapping set, già noto agli autori di BIKE come near-codeword, che è intrinseco nella struttura quasi ciclica del codice; nella seconda parte verranno invece esplorate nuove possibili tecniche di ricerca di trapping set applicate a questa categoria di codici. Prima però, verrà esplorata la relazione che esiste tra i δ-connected set, un oggetto definito sulla *matrice delle adiacenze*, e gli AS.

5.1 δ-connected set e Absorbing Set

I δ-connected set sono degli oggetti presenti nella matrice di parità, emersi in [134], che intaccano le prestazioni del decoder BF. Queste strutture vengono caratterizzate sfruttando la *matrice delle adiacenze*:

Definizione 17. *Sia* $\mathbf{H}$ *una matrice di parità di dimensione* $r \times n$*. Si definisce la sua matrice delle adiacenze* $\mathbf{\Gamma}$ *come una matrice di dimensione* $n \times n$ *il cui elemento appartenente all'*i*-esima riga e alla* j*-esima colonna* $\gamma_{i,j}$ *è pari al numero di intersezioni tra l'*i*-esima e la* j*-esima colonna di* $\mathbf{H}$ *quando* $i \neq j$*, e pari a 0 quando* $i = j$*.*

Esempio 4. *Esempio di matrice delle adiacenze relativa alla matrice di parità dell'Esempio 2.*

$$\mathbf{\Gamma} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 2 & 1 & 2 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 2 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 2 & 1 & 1 & 1 & 1 & 0 & 2 & 3 & 1 \\ 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 2 & 2 \\ 0 & 2 & 1 & 1 & 1 & 1 & 3 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 & 0 & 1 & 1 & 2 & 1 & 0 \end{pmatrix}$$

La matrice delle adiacenze così costruita è sempre simmetrica, e corrisponde univocamente ad un grafo non orientato, costituito da n vertici, uno per ogni riga (o colonna); l'i-esimo vertice è collegato al j-esimo se $\gamma_{i,j} > 0$, e al ramo che li collega è assegnato un peso, pari proprio a $\gamma_{i,j}$. Il grafo delle adiacenze può essere utilizzato per individuare vettori di errore che, con alta probabilità, causano un fallimento del BF decoder.

Un δ-connected set viene quindi individuato su tale matrice dalla seguente definizione:

Definizione 18. *Sia* $\mathbf{H}$ *una matrice di parità,* $\mathbf{\Gamma}$ *la sua matrice delle adiacenze e* $\mathbf{G}$ *il corrispondente grafo. Un* δ*-connected set di* $\mathbf{G}$ *è definito come un set di indici* $\mathcal{D} \subseteq \{1, 2, ..., n\}$ *tale che*

1. *l'intersezione tra* $\mathcal{D}$ *e il supporto di ciasuna riga di* $\mathbf{H}$ *è inferiore o uguale a 2;*

2. *per ogni indice $i \in \mathcal{D}$, la somma dei pesi dei rami connessi al nodo i è maggiore o uguale a δ, ed esiste almeno un nodo per il quale questa somma è esattamente uguale a δ.*

Si noti che vale la seguente relazione:

$$\min_{j\in\mathcal{D}} \left\{ \sum_{l\in\mathcal{D}\setminus\{j\}} \gamma_{j,l} \right\} = \delta \tag{18}$$

cioè il valore di δ è il valore minimo tra le somme degli elementi delle righe di $\mathbf{\Gamma}$ che appartengono al δ-connected set. Ragionando in termini di matrice di parità $\mathbf{H}$, δ si può ottenere come:

$$\delta = \min_{i\in\mathcal{D}} \left\{ \sum_{j\in\mathcal{D}\setminus\{i\}} |S(\mathbf{h}_i) \cap S(\mathbf{h}_j)| \right\} \tag{19}$$

e cioè come il minimo numero di intersezioni tra le colonne di $\mathbf{H}$ incluse in $\mathcal{D}$. Si può inoltre dimostrare il seguente lemma, che esplicita la relazione tra un δ-connected set e i contatori delle sue equazioni di parità:

Lemma 2. *Sia $\mathbf{H}$ una matrice di parità, e sia $\mathrm{wt}(\mathbf{h}_i)$ il peso dell'i-esima colonna di $\mathbf{H}$. Sia $\mathcal{D}$ un δ-connected set, e sia $\mathbf{e}$ un vettore di errore il cui supporto sia uguale a $\mathcal{D}$. Allora si ha:*

$$\sigma_i \leq \mathrm{wt}(\mathbf{h}_i) - \delta \quad \forall\ i \in \mathcal{D} \tag{20}$$

Il contatore σ_i indica il numero di equazioni di parità non soddisfatte a cui partecipa l'i-esimo bit del δ-connected set o, equivalentemente, il numero di nodi check non soddisfatti collegati all'i-esimo nodo variabile del sottografo indotto $\mathcal{G}(\mathcal{D})$. Il parametro δ rappresenta quindi la riduzione minima dei contatori per ciascun bit appartenente a $\mathcal{D}$. Come è stato già illustrato, un decoder BF riesce a riconoscere i bit affetti da errore se i loro contatori si trovano oltre una certa soglia: ecco che allora δ diventa significativo quando riduce sufficientemente i contatori, in modo da mantenerli sotto la soglia.

Sebbene la definizione di δ-connected set sia piuttosto generica, tale oggetto presenta una stretta analogia con l'Absorbing Set, che viene esplicitata dalla seguente proposizione:

Proposizione 1. *Sia* $\mathbf{H}$ *una matrice di parità; sia* $\mathrm{wt}(\mathbf{h}_i)$ *il peso dell'i-esima colonna di* $\mathbf{H}$*. Sia* $v = \max_{i \in \mathcal{D}} \{\mathrm{wt}(\mathbf{h}_i)\}$ *il peso della colonna con peso maggiore della matrice di parità che appartiene al* δ*-connected set. Sia* $\mathcal{D}$ *un* δ*-connected set con* $\delta \geq v/2 - 1$ *se* v *è pari, o* $\delta \geq (v-1)/2$ *se* v *è dispari. Allora* $\mathcal{D}$ *è un Absorbing Set.*

Dimostrazione. Secondo il Lemma 2 ogni bit del δ-connected set ha un numero di contatori limitato superiormente:

$$\sigma_i \leq \mathrm{wt}(\mathbf{h}_i) - \delta \ .$$

Ammettendo che v sia pari, che j sia l'indice di $\mathcal{D}$ che ha peso v nella matrice di parità e che $\delta = v/2 - 1$, il numero dei suoi contatori sarà limitato da:

$$\sigma_j \leq v - v/2 - 1$$

e quindi

$$\sigma_j \leq v/2 - 1$$

ossia, tale nodo variabile partecipa a più equazioni di parità soddisfatte che non soddisfatte. Se questo è vero per l'elemento di $\mathcal{D}$ a peso maggiore in $\mathbf{H}$, è vero anche per tutti gli altri a peso minore, rispettando quindi la Definizione 14, che rimane verificata anche per tutti i valori di δ maggiori. Analogamente, si può dimostrare facilmente che la stessa conclusione vale nel caso di v dispari. □

Alla luce di questa corrispondenza, si può vedere il δ-connected set come una sorta di Absorbing Set *generalizzato*: a seconda del valore di δ, questi possono coincidere o meno con la definizione di Absorbing Set; la limitazione al numero dei contatori può essere sia più stringente ($\delta > d_v/2$), sia più lasca ($\delta < d_v/2$), permettendo quindi una maggiore flessibilità nella definizione di strutture che portano un certo tipo di decoder al fallimento (si pensi, ad esempio, che un decoder BF con una soglia più bassa di $d_v/2$ riesce a correggere un Absorbing Set, se questo non ha una riduzione dei contatori tale da mantenere tutti i suoi nodi variabile sotto la soglia, mentre non riesce a correggere un δ-connected con un δ sufficientemente elevato).

5.2 Fully Absorbing Set della classe (d_v, d_v)

Gli stessi autori di BIKE dichiarano l'esistenza di una near-codeword (d_v, d_v) costituita da $S(\mathbf{h}_0)$ [5], ossia il supporto della prima colonna del primo blocco circolante che forma la matrice di parità; si evidenza anche che la sindrome che tale vettore d'errore genera coincide con $(\mathbf{h}_0)^2$. Di seguito si cercherà di caratterizzare con più precisione ciò che accade mediante una proposizione che, seppur con delle ipotesi semplificative, aiuta a descrivere il processo di formazione di questo trapping set.

Proposizione 2. *Sia* $\mathbf{H}_0$ *un blocco circolante di dimensione* $p \times p$*, con* $p \geq 2 \cdot \max\{S(\mathrm{h}_0^T)\}$*, di un codice* (d_v)*-regolare con* $d_v \geq 2$*, e che rispetta il vincolo RC; allora* $\mathbf{e} = \mathbf{H}_{0_{:,i}}$ *con* $1 \leq i \leq p$*, è un Absorbing Set appartenente alla classe* (d_v, d_v)*.*

Dimostrazione. Sia $h(x)$ il polinomio associato alla prima colonna del blocco circolante, corrispondente a $\mathbf{H}_{0_{:,1}}$:

$$h_0(x) = x^{a_0} + x^{a_1} + x^{a_2} + ... + x^{a_{d_v-1}} \ .$$

La k-esima colonna di $\mathbf{H}_0$, essendo circolante, sarà una versione opportunamente shiftata della prima colonna:

$$h_k(x) = x^{a_0+k} + x^{a_1+k} + x^{a_2+k} + ... + x^{a_{v-1}+k}|_{\text{mod } p} \ .$$

Di seguito si ometterà la notazione di operazione in modulo, in virtù dell'ipotesi posta su p. Si assuma un vettore di errore tale che

$$S(\mathbf{e}) = \{a_0, a_1, a_2, ..., a_{d_v-1}\} \ .$$

La sottomatrice indotta da questo vettore di errore sarà composta da una serie di colonne shiftate di $\mathbf{h}_0$, di cui la prima sarà rappresentata da

$$h_{a_0}(x) = x^{2a_0} + x^{a_1+a_0} + x^{a_2+a_0} + ... + x^{a_{d_v-1}+a_0}$$

Poiché il codice è regolare per ipotesi esso rispetta il vincolo RC, cioè le coppie di colonne di $\mathbf{H}$ si intersecano al massimo in una sola posizione, per

[5] Si ricorda che la matrice di parità nel crittosistema BIKE è costituita da due blocchi circolanti tali che $\mathbf{H} = (\mathbf{H}_0|\mathbf{H}_1)$; se $\mathbf{h}_0$ rappresenta la prima colonna di $\mathbf{H}_0$.

cui ciascuna nuova colonna che si aggiunge ha uno ed un solo elemento in comune con la prima:

$$h_{a_1}(x) = x^{a_0+a_1} + x^{2a_1} + x^{a_2+a_1} + ... + x^{a_{d_v-1}+a_1}$$

$$h_{a_2}(x) = x^{a_0+a_2} + x^{a_1+a_2} + x^{2a_2} + ... + x^{a_{d_v-1}+a_2}$$

$$h_{a_{d_v-1}}(x) = x^{a_0+a_{d_v-1}} + x^{a_1+a_{d_v-1}} + x^{a_2+a_{d_v}} + ... + x^{2a_{d_v-1}} \ .$$

Poiché viene rispettato il vincolo RC, ogni polinomio ha in comune un solo elemento con gli altri; ciò significa che gli elementi del tipo:

$$x^{a_i+a_j}$$

si elidono (e quindi i relativi nodi check vengono soddisfatti), mentre rimangono soltanto i termini del tipo

$$x^{2a_i}$$

che sono esattamente i rimamenti d_v nodi check non soddisfatti, che sono di grado 1, ciasuno connesso all'i-esimo nodo variabile. Ciò significa che il numero di equazioni non soddisfatte a cui partecipa ciascun nodo variabile è tale che:

$$\sigma_i = 1 < \frac{d_v}{2} \ \forall \ i$$

il che dimostra che $\mathbf{e}$ è un *Absorbing Set*. È inoltre evidente che il polinomio rappresentativo dei nodi check non soddisfatti

$$c(x) = \sum_{i=a_1}^{a_{d_v-1}} h_i(x) = x^{2a_0} + x^{2a_1} + ... + x^{2a_{d_v-1}}$$

coincide proprio con $h_0^2(x)$. □

Nei codici QC-MDPC il vincolo RC non viene rispettato, inoltre l'ipotesi che $p \geq 2 \cdot \max\{S(\mathrm{h}_0^T)\}$ non è verificata; il trapping set che si viene a formare non è quindi *elementary* (com'è, invece, quello della Proposizione 1, dato che i nodi non soddisfatti sono di grado 1, e quelli soddisfatti di grado 2). I nodi non soddisfatti possono essere anche di grado 3, 5 o più, mentre i nodi soddisfatti sono di grado maggiore o uguale di 2. Nondimeno, la sindrome generata coincide comunque con $\mathbf{h}_0^2$. Inoltre questi Absorbing Set, intrinseci

nella struttura della matrice di parità, hanno anche la caratteristica di essere *Fully* se d_v si mantiene oltre una certa soglia, al di sotto della quale diventano semplici Absorbing Set, sebbene anche questo non sia dimostrato analicamente nella Proposizione 1.

In ogni caso, per $d_v = 71$, $\mathbf{h}_0$ costituisce un Fully Absorbing Set; a causa della caratteristica circolante, inoltre, ciascuna delle p possibili permutazioni di $\mathbf{h}_0$ mantiene le medesime caratteristiche. Pertanto, ogni blocco circolante contiene p FAS, e poiché la matrice di parità è composta da due blocchi, $\mathbf{H}$ ha ben $2p$ FAS della classe (d_v, d_v). Nello specifico caso preso in esame, si hanno quindi 24646 FAS di classe $(71, 71)$.

È chiaramente di fondamentale importanza capire il contributo che questi danno alla regione di error floor; per farlo bisognerebbe conoscere il valore del numero critico μ e della forza ν per ciascuno di essi. Questi vengono, nel caso degli LDPC, calcolati mediante un approccio a ricerca esaustiva: data la dimensione a del trapping set, si testano tutti i possibili $\binom{a}{t}$ inducing set di dimensione t, per $1 \leq t \leq a$. Ovviamente, tale procedura non può essere seguita per trapping set di dimensioni così elevate. Servirebbe quindi un'analisi teorica, che tenga ovviamente conto delle caratteristiche del decoder, che riesca a predire (o a stimare) la dimensione del minimo inducing set, o le caratteristiche che un sottografo indotto da un inducing set deve avere per portare il decoder a convergere al FAS. Sebbene tale analisi non sarebbe affatto triviale, un'idea è, nel caso del decoder BF, quella di partire dalle analisi *density evolution* [133, 176, 177] che stimano, mediante considerazioni combinatorie, la capacità correttiva del decoder. Tuttavia, un possibile lower bound per il numero critico può essere rappresentato dal seguente Teorema, dimostrato in [134]:

Teorema 1. *Si consideri un codice definito da una matrice di parità* $\mathbf{H}$ *in cui ogni colonna ha peso maggiore o uguale di* v^**. Sia* t *un intero tale che* $t \leq t_M$*, dove* t_M *è l'intero più grande tale che*

$$v^* > \Theta(\mathbf{H}, t_M) + \Theta(\mathbf{H}, t_M - 1), \tag{21}$$

dove $\Theta(\mathbf{H}, t_M)$ *è definito come il massimo tra le somme dei* t_M *elementi più grandi in ciascuna riga della matrice delle adiacenze* $\mathbf{\Gamma}$*. Allora un decoder BF con soglie tali che:*

$$b_i \in [\Theta(\mathbf{H}, t_M) + 1, v_i - \Theta(\mathbf{H}, t_M - 1)] \tag{22}$$

corregge tutti i vettori di errore con peso minore o uguale a t *in una iterazione.*

Invece di considerare l'intera matrice di parità si può infatti considerare il sottografo indotto da questi FAS $\mathcal{G}(\mathcal{T})$ e calcolarne la matrice delle adiacenze $\mathbf{\Gamma}_{\mathcal{T}}$; successivamente, si calcola $\Theta(\mathbf{H}_{\mathcal{T}}, t_i)$ per ogni t_i necessario a verificare la (21); il t_i massimo che la verifica viene indicato con t_M. Se le soglie del decoder BF soddisfano la (22), come si verifica nel caso in esame, t_M sarà un limite inferiore alla dimensione dell'inducing set minore, e quindi del numero critico μ. Tale analisi risulta lecita, in quanto il supporto degli inducing set $S(\mathbf{e})$ è sempre un sottoinsieme del supporto del trapping set $\mathcal{T}$; pertanto, è sufficiente studiare la sottomatrice indotta da $\mathcal{T}$. Dunque, si può calcolare un limite inferiore al numero critico, e utilizzarlo per effettuare una stima dell'error floor utilizzando la (17).

In ultimo, vale la pena di evidenziare la particolare struttura di questi trapping set; si può notare infatti come ogni nodo variabile sia connesso, mediante i nodi check, ad ogni altro nodo variabile del grafo. Questo aspetto può essere messo in evidenza mediante il grafo delle adiacenze del fully absorbing set. La matrice delle adiacenze relativa alla sottomatrice indotta[6] dai FAS di cui sopra genera un grafo completamente connesso o, in altre parole, una *clique*:

Definizione 19. *Una clique, o cricca, è un insieme $\mathcal{V}$ di nodi (o vertici) in un grafo non orientato tale che esiste un ramo che collega ogni coppia di nodi. In modo equivalente si può dire che essa costituisce un grafo completo.*

I trapping set (d_v, d_v) dovuti alla caratteristica circolante dei blocchi della matrice di parità costituiscono quindi delle clique; questo apre ad uno scenario potenzialmente interessante: data la densità della matrice di parità nei codici MDPC, è praticamente certo che il grafo di Tanner contenga molti oggetti di questo tipo. Le clique inoltre costituiscono, a parità di numero di vertici, la struttura più densa in assoluto, ovvero con il più alto numero di rami, caratteristica cruciale secondo la "logica" degli Absorbing Set o, più in generale, dei δ-connected set. Avere una struttura densa significa infatti ridurre il più possibile la presenza di nodi check non soddisfatti di grado 1, in modo da ottenere con maggiore probabilità una riduzione dei contatori. È certamente vero che, in tal modo, c'è comunque la possibilità di ottenere nodi check non soddisfatti di grado 3, 5 o più, che è però da tenere in conto, dato che da quanto è emerso in questa tesi, è estremamente probabile che non siano gli

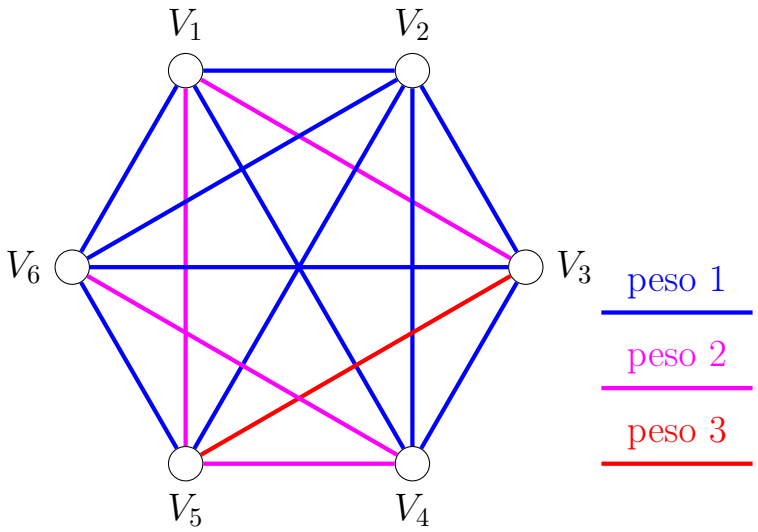
[6]Per sottomatrice indotta si intende la matrice costituita dalle colonne della matrice di parità "attivate" dal trapping set. È evidente che il grafo di Tanner della sottomatrice indotta da un trapping set sarà proprio il sottografo indotto.

Elementary Trapping Set le strutture più pericolose (questo sarebbe dato dal fatto che il codice non rispetta il vincolo RC).

La caratteristica di essere delle clique di questi trapping set ha ispirato la seconda parte del lavoro, che concerne la ricerca di ulteriori (e, possibilmente, più piccoli) trapping set.

Esempio 5. *Si faccia riferimento sempre alla matrice di parità dell'Esempio 2. Il vettore d'errore* $\mathbf{e}$ *tale che* $S(\mathbf{e}) = \{2, 5, 7, 8, 9, 10\}$ *induce un grafo completo.*

$$\mathbf{H}^{(\mathbf{e})} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad \mathbf{\Gamma}^{(\mathbf{e})} = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & 2 & 3 & 1 \\ 1 & 1 & 2 & 0 & 2 & 2 \\ 2 & 1 & 3 & 2 & 0 & 1 \\ 1 & 1 & 1 & 2 & 1 & 0 \end{pmatrix}$$



5.3 Ricerca di clique

La ricerca di clique è stata impiegata per la ricerca di trapping set. In particolare si è scelto di applicare un algoritmo che risolvesse il *Maximum Edge-Weight Clique Problem* (MEWCP), che è strettamente legato al *Maximum Clique Problem* (MPC), che è a sua volta notoriamente noto come problema NP-complete. Si noti da subito che tale approccio permette di ottenere come output la clique a peso massimo in un grafo di adiacenze; pertanto, non si è intrapresa la strada dell'esaustività (ovvero della ricerca di tutti i trapping set), in quanto si è voluto in primis osservare se tale approccio potesse portare a dei risultati. In accordo con quanto detto precedentemente, una clique ad alto peso equivale ad un sottografo di Tanner denso, e che quindi ha un'elevata probabilità di costituire un AS. Per quanto i progressi nell'ambito della risoluzione dell'MEWCP abbiano portato ad avere algoritmi sempre più efficienti, la complessità rimane tale da non poter applicare un algoritmo di ricerca alla matrice di parità di un codice MDPC. Pertanto, la ricerca di clique è stata applicata in combinazione con la tecnica *Subgraph-Expansion*, presentata in [153]. L'obiettivo è quello di limitare il raggio d'azione dell'algoritmo di ricerca su una sottomatrice più piccola e densa della matrice di parità, ottenuta espandendo una coppia di nodi. Limitando la ricerca ad un sottoinsieme di nodi variabile si riduce drasticamente la complessità; inoltre, se esiste una clique densa all'interno del grafo di Tanner, questa comprenderà con maggiore probabilità i nodi più densamente connessi tra loro; si tratta ovviamente di un discorso probabilistico, che intrinsecamente esclude l'ipotesi di effettuare una ricerca esaustiva. Chiaramente, l'algoritmo di espansione di [153] è stato leggermente modificato, in modo da poterlo applicare in maniera più consona ai codici MDPC, ed è illustrato nell'Algoritmo 2, discusso e spiegato di seguito.

$\mathcal{L}_{INI}$ rappresenta la lista delle strutture di partenza. Originalmente, Raveendran et al. [153] hanno proposto di partire da cicli semplici di lunghezza g; tuttavia, si è ritenuto più opportuno, in questa sede, partire dalle coppie di nodi variabile con la massima intersezione. Data la natura quasi ciclica della matrice di parità, è sufficiente partire dalla prima colonna e trovare con quali altre colonne essa si interseca maggiormente, per esempio calcolando la prima riga della matrice delle adiacenze. Tutte le altre coppie non saranno altro che permutazioni cicliche delle coppie così trovate. Tuttavia, in assenza di una struttura quasi ciclica, sarebbe sufficiente modificare leggermente un algoritmo di ricerca di cicli da 4 (in modo da trovare coppie di nodi variabile che condividono il massimo numero di nodi check), per ottenere lo stesso

risultato (la complessità è comunque irrisoria). Con $\mathcal{V}^{(0)}$ si intende l'insieme $\{v_1, v_2\}$ dei nodi variabile delle strutture di partenza in $\mathcal{L}_{INI}$. Dato l'insieme iniziale di nodi variabile, $\mathcal{G}(\mathcal{V}^{(0)})$ ne denota il sottografo indotto; i nodi check del sottografo indotto, indicati con $\Gamma^{(0)}$, si dividono per il loro grado, in modo che $\Gamma^{(0)} = \Gamma_1^{(0)} \cup \Gamma_2^{(0)} \cup ... \cup \Gamma_{d_c}^{(0)}$, dove $\Gamma_x^{(0)}$ rappresenta l'insieme dei nodi check di grado x, mentre l'apice denota la profondità di espansione del grafo. Per ciascun nodo check di grado uno appartenente a $\Gamma_1^{(0)}$ si ricava la lista dei suoi nodi variabile vicini, che non appartengono già alla struttura di partenza; per ognuno di questi si prende, a loro volta, la lista dei nodi check vicini (e si esclude il nodo check dal quale si è partiti); se il numero di nodi check in condivisione tra la struttura di partenza e il nodo variabile w è superiore ad una certa soglia D, allora w viene aggiunto alla struttura di partenza. La struttura di partenza, assieme a tutti i nodi variabile che soddisfano la condizione di cui sopra, diventa a sua volta la struttura di partenza per l'espansione alla profondità successiva. Se la struttura espansa e quella di partenza sono della stessa dimensione, non ci sono più nodi variabile da aggiungere, e la procedura termina; inoltre, questa può anche essere interrotta mediante il parametro κ, che pone un vincolo arbitrario sulla dimensione dei grafi espansi. Vale la pena soffermarsi sul parametro D: questo viene posto in funzione di δ, e rappresenta appunto la soglia minima di intersezioni che un nuovo nodo variabile deve avere per poter essere aggiunto alla struttura; è evidente che, all'aumentare delle dimensioni del grafo di partenza, e quindi del numero di nodi check in esso, i nuovi nodi variabile tenderanno ad avere sempre più nodi check in comune con il grafo di partenza; la scelta di $f(\delta)$, pur arbitraria, mira a far sì che i nuovi nodi variabile siano soltanto quelli con un numero minimo di intersezioni con la struttura: assumere una funzione lineare porterà ad avere grafi ampi e meno densi; una funzione esponenziale, viceversa, permetterà di ottenere grafi espansi più piccoli e più densi (è ovvio dire che la soglia deve aumentare all'aumentare di δ).

Una volta ottenuto il grafo espanso, si verifica che induca il decoder in errore: se il decoder riesce a correggerlo, è evidente che il grafo espanso non contiene alcun trapping set all'interno; se il decoder fallisce, si applica l'algoritmo di ricerca di clique a massimo peso. In questo lavoro si è scelto di applicare l'algoritmo descritto in [178] da Shimizu et al. che permette un'implementazione piuttosto semplice. Tale algoritmo utilizza un approccio branch-and-bound, che divide ricorsivamente il problema iniziale $P(C = \emptyset, S)$ (dove C è una clique soluzione del sottoproblema superiore, ed S è l'insieme dei vertici su cui si effettua la ricerca) in sottoproblemi più piccoli $P(C', S')$

Algorithm 2 Subgraph expansion

Input: $\mathbf{H}$, κ, $\mathcal{L}_{INI}$
Output: $\mathcal{L}_{EXP}$

$\mathcal{L}_{EXP} \leftarrow \emptyset$
for tutti gli elementi $\in \mathcal{L}_{INI}$ **do**
    Insieme iniziale di nodi variabile $= \mathcal{V}^{(0)}$
    Profondità d'espansione $\delta \leftarrow 0$
    Soglia di intersezione iniziale $D = f(0)$
    do
        $W \leftarrow \emptyset$
        Sottografo $\mathcal{G}(\mathcal{V}^{(\delta)})$ e suoi nodi check $\Gamma^{(\delta)}$
        for tutti i $c \in \Gamma_1^{(\delta)}$ **do**
            for tutti i $w \in \mathcal{N}(c), w \notin \mathcal{V}^{(\delta)}$ **do**
                if $|\mathcal{N}(w) \setminus c \in \Gamma^{(\delta)}| > D$ **then**
                $W \leftarrow W \cup w$
                end if
            end for
        end for
        $\delta \leftarrow \delta + 1$
        $\mathcal{V}^{(\delta)} \leftarrow W \cup \mathcal{V}^{(\delta-1)}$
        Aggiornamento soglia $D = f(\delta)$
    while $|\mathcal{V}^{(\delta)}| > |\mathcal{V}^{(\delta-1)}|$ e $|\mathcal{V}^{(\delta)}| < \kappa$
    Insieme espanso di nodi variabile $= \mathcal{V}^{(\delta)}$
    $\mathcal{L}_{EXP} \leftarrow \mathcal{V}^{(\delta)}$
end for

alla ricerca di soluzioni ottime. Per ogni sottoproblema viene calcolato un limite superiore al peso di ogni possibile soluzione, permettendo così di scartare quelle che non hanno possibilità di produrre una clique di peso maggiore di quella relativa al problema corrente. Un algoritmo di ricerca di clique si divide in: *branching strategy*, *search strategy* e *pruning rule*. Partendo dall'ultima, le pruning rules hanno il ruolo fondamentale di ridurre la complessità dell'algoritmo scartando le opzioni con meno probabilità di successo; in questo algoritmo la pruning rule sfrutta un upper bound al peso delle clique. Gli Algoritmi 3,4 e 5 illustrano la ricerca di clique; l'applicazione della pruning rule si può osservare nell'Algoritmo 5, dove $w(v)$ e $w(u, v)$ denotano il peso del vertice v e del ramo (u, v), rispettivamente; in tale procedura si effettua

la *colorazione dei vertici*, ovvero ad ogni vertice viene assegnato un colore in modo che tutti i vertici adiacenti tra loro abbiano colori diversi. Il numero minimo di colori necessari alla colorazione è chiamato *numero cromatico* $\mathcal{X}(\mathcal{G})$. Gli insiemi di vertici dello stesso colore prendono il nome di *set indipendenti*. Dato che in una clique ogni vertice è connesso a tutti gli altri, ogni vertice dovrà appartenere ad un diverso set indipendente; è quindi chiaro che il numero cromatico $\mathcal{X}(\mathcal{G})$ costituisce un limite superiore al numero di vertici di una clique. Per il *Maximum Weight Clique Problem* (MWCP), la somma dei massimi pesi di vertice per ogni set indipendente costituisce il limite superiore al peso della clique; in questo caso, dove il peso è associato ai rami e non ai vertici, ci si riconduce al MWCP assegnando ad ogni vertice un peso sulla base del peso dei rami incidenti. Si ottiene perciò $\sigma[v]$, che è il peso totale assegnato al vertice v mediante la seguente formula:

$$\sigma[v] = w(v) + \sum_{i<\tau(v)} \max\{w(u,v)|u \in I_i \cap \mathrm{N}(v)\} \tag{23}$$

dove $\tau(v)$ è l'indice del set indipendente I che contiene v. Essendo il grafo pesato sui rami e non sui vertici, $w(v)$ è nullo per ogni v all'interno del problema iniziale $P(\emptyset, S)$, mentre nei sottoproblemi successivi $w(v)$ equivale al $\sigma[v]$ del sottoproblema superiore. L'Algoritmo 3 mostra la parte principale della procedura. Gli ingressi sono il grafo $\mathcal{G}$ (ottenuto dalla matrice di incidenza delle strutture in $\mathcal{L}_{MAX}$, l'output dell'Algoritmo 2), e una soluzione iniziale $\mathcal{C}_{\text{initial}}$, che può essere l'insieme vuoto o, semplicemente, una coppia di nodi qualsiasi (in quest'ultimo caso si riesce ad effettuare un pruning iniziale delle soluzioni). La ricerca della soluzione dell'MEWCP avviene mediante chiamate ricorsive alla funzione EXPAND. L'Algoritmo 4 mostra la procedura ricorsiva che aggiorna la soluzione al problema. Se S è vuoto, la soluzione al sottoproblema è $C_{\max}$; altrimenti la funzione CALC_SEQ_AND_UB restituisce una sequenza $\Pi = [p_1, p_2, ..., p_{|S|}]$ di vertici in S con i relativi upper bound, e per ognuno di questi, se l'upper bound consente di aumentare il peso di $C_{\max}$, viene ripetuta ricorsivamente la procedura. Branching strategy, pruning rules e search strategy dell'algoritmo sono descritte di seguito:

- **Branching Strategy**: per ogni p_i vengono generati una serie di sottoproblemi $P(C \cup \{p_i\}, (S \setminus \{p_i|j < i\}) \cap \mathcal{N}(p_i))$; poiché i vertici in Π sono ordinati in modo da avere un upper bound non crescente, il sottoproblema include solo i vertici vicini di p_i e che hanno un upper bound più elevato di quello di p_i.

- **Pruning Rules**: per ogni $P(C \cup \{p_i\}, (S \setminus \{p_i | j < i\}) \cap \mathcal{N}(p_i))$ viene calcolato l'upper bound sulla base della (23), e associato ad ogni p_i mediante l'array *upper*.

- **Search Strategy**: Per ogni p_i l'algoritmo cerca $P(C \cup \{p_i\}, (S \setminus \{p_i | j < i\}) \cap \mathcal{N}(p_i))$; poiché, come detto prima, Π è in ordine non crescente in base all'upper bound, le soluzioni trovate per prime sono quelle con upper bound maggiore, e quindi con maggiore probabilità di essere la soluzione al problema.

Algorithm 3 MECQ

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $C_{initial}$
Output: C_{max}

1: $C_{max} \leftarrow C_{initial}$
2: $C_{max} \leftarrow \text{EXPAND}(C_{max}, \emptyset, \mathcal{V})$
3: **return** C_{max}

Algorithm 4 EXPAND

Input: un sottoproblema $P(C_{max}, C, S)$
Output: Aggiorna C_{max} ad una clique più densa, se esiste

1: **if** $S = \emptyset$ **then**
2: **if** $\text{wt}(C)/|C| > \text{wt}(C_{max})/|C_{max}|$ **then**
3: $C_{max} \leftarrow C$
4: **end if**
5: **else**
6: $\Pi, upper \leftarrow \text{CALC\_SEQ\_AND\_UB}(C, S)$
7: **for** ogni $p_i \in \Pi$ **do**
8: **if** $\text{wt}(C) + upper(p_i) > \text{wt}(C_{max})$ **then**
9: $C_{max} \leftarrow \text{EXPAND}(C_{max}, C \cup \{p_i\}, (S \setminus \{p_i | j < i\}) \cap \mathcal{N}(p_i))$
10: **end if**
11: **end for**
12: **end if**
13: **return** C_{max}

Algorithm 5 CALC_SEQ_AND_UB

Input: una clique C e un insieme di vertici S
Output: una sequenza di vertici Π e un array *upper*

1: **for** $v \in S$ **do**
2: $\quad \sigma[v] \leftarrow w_p(v)$
3: **end for**
4: $S' \leftarrow S$
5: $k \leftarrow 0$
6: **while** $S' \neq \emptyset$ **do**
7: $\quad k \leftarrow k + 1$
8: $\quad I_k \leftarrow \emptyset$
9: $\quad X \leftarrow S'$
10: $\quad$ **while** $X \neq \emptyset$ **do**
11: $\quad\quad v \leftarrow$ un vertice con σ minimo in X
12: $\quad\quad upper[v] \leftarrow \sigma[v] + \sum_{i<k} \max\{\sigma[u] | u \in I_i\}$
13: $\quad\quad I_k \leftarrow I_k \cup v$
14: $\quad\quad \Pi \leftarrow \{v, \Pi\}$
15: $\quad\quad X \leftarrow X \setminus \mathcal{N}(v)$
16: $\quad\quad S' \leftarrow S' \setminus \{v\}$
17: $\quad$ **end while**
18: $\quad$ **for** $v \in S'$ **do**
19: $\quad\quad \sigma[v] \leftarrow \sigma[v] + \max\{w(u, v) | u \in \mathcal{N}(v) \cap I_k\}$
20: $\quad$ **end for**
21: **end while**
22: **return** $\Pi, upper$

L'Algoritmo 5 mostra la funzione CALC_SEQ_AND_UB. Questa riceve come input un sottoproblema $P(C, S)$ e restituisce una sequenza $\Pi = [p_1, p_2, ..., p_{|S|}]$ di vertici in S e un vettore *upper* di upper bound legati ai vertici in Π. Viene innanzitutto inizializzato $\sigma[v]$ con i valori di $w_p(C, v) = \sum_{u \in C} w(u, v)$. Dalla linea 6 alla 21 vengono creati una serie di set indipendenti I_k; i vertici appartenenti a I_k vengono aggiunti **in testa** a Π nell'ordine in cui vengono assegnati ai set indipendenti, e ne viene aggiornato il rispettivo peso σ. Dalla linea 10 alla 17 vengono costruiti i set indipendenti massimi, dove X rappresenta l'insieme dei candidati ad essere aggiunti a I_k. Alla linea 11 viene scelto un vertice $v \in X$ tale che abbia σ minimo: in tal modo, il vettore *upper* risulterà non crescente; in questo modo l'algoritmo riesce a trovare subito clique pesanti. Infine, alla linea 19 viene aggiornato il σ dei vertici che non sono stati aggiunti a nessun set indipendente, ma che sono adiacenti ai vertici aggiunti a I_k. Si noti che, rispetto alla versione originale, è stata apportata una piccola modifica alla linea 2 dell'Algoritmo 4: la clique viene aggiornata se il rapporto tra peso e numero di vertici è maggiore, in modo da ottenere clique più dense, piuttosto che con più nodi. È evidente che l'algoritmo può essere modificato in maniera decisamente più profonda per portarlo alla ricerca di clique dense, mentre tale modifica rappresenta soltanto un primo piccolo passo verso tale obiettivo.

In ogni caso, l'intera procedura di formazione di un grafo espanso e ricerca di clique può essere riassunta come segue:

1. trovare le coppie di colonne a massima intersezione all'interno della matrice di parità;

2. espandere ogni coppia tramite l'Algoritmo 2;

3. calcolare la matrice delle adiacenze $\mathbf{\Gamma}$ delle strutture espanse ed applicare l'algoritmo di ricerca di clique.

Sono state effettuate numerose prove, variando la dimensione delle strutture espanse. Seppur non siano stati trovati nuovi absorbing set, si è visto che si riescono comunque a trovare set con una discreta riduzione dei contatori, il che conferisce una discreta validità al metodo di ricerca il quale, se affinato, presenta delle probabilità di trovare con successo degli Absorbing Set. Certo è che tutto dipende da come viene effettuata l'espansione: più grande è infatti il grafo espanso, e più le clique ad alto peso sono semplicemente quelle con un maggior numero di vertici. Perciò l'algoritmo è stato leggermente modificato: sebbene la procedura di ricerca sia la stessa, l'output di ogni sotto-problema

è la clique più densa, ossia con il maggiore rapporto tra peso complessivo dei rami e numero di vertici. Questo ha permesso di migliorare l'efficacia dell'algoritmo, che tuttavia non è riuscito ad individuare (pur variando i parametri) clique che facessero fallire il decoder. Questo fa comunque riflettere sull'importanza dei trapping set descritti nella sezione precedente che, allo stato attuale, costituiscono le strutture pericolose più piccole note. Inoltre, ulteriori analisi, per esempio ricercando clique su grafi espansi partendo da colonne ad intersezione minore, o costruiti in altro modo, vanno sicuramente compiute.

6 Conclusioni e sviluppi futuri

In questa tesi si è cercato di studiare la regione di error floor di codici MDPC mediante l'analisi dei trapping set contenuti nel loro grafo di Tanner. Innanzitutto è stata chiarita l'analogia tra δ-connected set e Absorbing Set, evidenziando come i primi possano rappresentare una generalizzazione degli ultimi.

Si è visto poi, mediante una dimostrazione pur semplificata ma significativa, che la caratteristica di quasi ciclicità porta alla presenza di p Absorbing Set o Fully Absorbing Set nel singolo blocco circolante; per poter determinare rigorosamente il loro impatto sulla regione di error floor, applicando quindi la (17), si devono conoscere il numero critico μ e la forza ν. Data la dimensione di questi set, non è possibile intraprendere una ricerca esaustiva di inducing set: questa rappresenta un'altra importante differenza rispetto allo studio dei trapping set in codici LDPC. Almeno secondo le conoscenze attuali, questa classe di trapping set sembra essere quella con effetto dominante sulla probabilità di errore, e in generale la sensazione è che all'aumentare del peso di colonna della matrice di parità la dimensione media di questi trapping set aumenti, assieme alla loro molteplicità. Per questo c'è la necessità di sviluppare analisi teoriche che riescano a stimare μ e ν di un trapping set in funzione dell'algoritmo di decodifica. In ogni caso, in questa tesi viene proposto un limite inferiore alla dimensione dell'inducing set critico.

La caratteristica saliente di questi set è quella di essere delle clique, ed ha ispirato la ricerca di questi oggetti mediante l'Algoritmo 3: se esistono degli Absorbing Set più piccoli di quelli noti, è lecito supporre che questi siano topologicamente più densi, ma poiché non esiste struttura più densa della clique (che ha il numero massimo di rami possibile, a parità di nodi), la ricerca di ulteriori clique sembra la strada più logica. La ricerca di clique permette inoltre di sfruttare algoritmi che, allo stato dell'arte, sono di gran lunga più efficienti di qualsiasi algoritmo di ricerca di Absorbing Set esistenti in letteratura; nonostante questo, essi non sono applicabili all'intera matrice di parità di un codice MDPC, per questo si è limitata la ricerca a sottografi ottenuti mediante un'espansione, illustrata dall'Algoritmo 2. Sebbene questa strada non abbia portato alla scoperta di nuovi Absorbing Set, ha permesso comunque di trovare strutture con una discreta riduzione dei contatori, ossia δ-connected set. È chiaro che l'algoritmo risolve un problema, quello di trovare la clique a peso massimo all'interno di un grafo, e non restituisce quindi una lista delle strutture trovate; nondimeno, spesso la clique a peso massimo

coincide con quella con il maggior numero di vertici, e non con la più densa, che sarebbe invece di maggior interesse. Pertanto, potrebbe essere utile modificare l'algoritmo (più profondamente di quanto fatto in questo lavoro), in modo da rendere più efficace la ricerca.

In conclusione, le problematiche relative alla densità della matrice di parità non si riducono soltanto ad un aumento della complessità degli algoritmi di ricerca di trapping set, ma sono relative anche (e soprattutto) alle nuove geometrie che tale densità comporta nel grafo di Tanner, causate principalmente dalla rimozione del vincolo RC; in questo contesto, lo studio del grafo delle adiacenze, rispetto al grafo di Tanner, sembra essere più efficace, poiché rappresenta meglio le connessioni tra i nodi variabile. Attualmente, lo studio di sottografi espansi sembra l'unica soluzione possibile per riuscire ad applicare algoritmi di ricerca, precludendo però in partenza l'obiettivo dell'esaustività. Inoltre, non è detto che la ricerca di clique sia l'approccio migliore possibile, data l'ipotesi intrinseca sulla topologia delle strutture che questo comporta, a discapito di un approccio più generale. Insomma, questo lavoro rappresenta soltanto un primo passo verso la conoscenza e lo studio dell'argomento, che dovrà proseguire continuando a considerare il possibile ruolo dei δ-connected set, producendo risultati teorici (gli unici in grado di superare realmente il problema della complessità) e, infine, impiegando nuove tecniche (di cui la ricerca di clique è solo un esempio).

Riferimenti bibliografici

[1] Plutarch. *Parallel Lives*.

[2] Suetonius. *De Vita Caesarum*.

[3] Wenbo Mao. *Modern cryptography: theory and practice*. Pearson Education India, 2003.

[4] Friedrich Ludwig Bauer. *Decrypted secrets: methods and maxims of cryptology*. Springer Science & Business Media, 2002.

[5] R. Davis. The Data Encryption Standard in perspective. *IEEE Communications Society Magazine*, 16(6):5–9, 1978.

[6] M.E. Smid and D.K. Branstad. Data Encryption Standard: past and future. *Proceedings of the IEEE*, 76(5):550–559, 1988.

[7] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. 1999.

[8] Joan Daemen and Vincent Rijmen. *The design of Rijndael*, volume 2. Springer, 2002.

[9] James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, and Edward Roback. Report on the development of the Advanced Encryption Standard (AES). *Journal of Research of the National Institute of Standards and Technology*, 106(3):511, 2001.

[10] W Stanley Jevons. *The principles of science*. 1877.

[11] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[12] James H Ellis. The possibility of secure non-secret digital encryption. *UK Communications Electronics Security Group*, 8, 1970.

[13] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[14] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[15] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International journal of information security*, 1(1):36–63, 2001.

[16] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography with Coding Theory (2nd Edition)*. Prentice-Hall, Inc., USA, 2005.

[17] Uma Somani, Kanika Lakhani, and Manish Mundra. Implementing digital signature with RSA encryption algorithm to enhance the data security of cloud in cloud computing. In *2010 First International Conference On Parallel, Distributed and Grid Computing (PDGC 2010)*, pages 211–216, 2010.

[18] Donald W Davies. Applying the RSA digital signature to electronic mail. *Computer*, 16(02):55–62, 1983.

[19] Lein Harn. New digital signature scheme based on discrete logarithm. *Electronics Letters*, 30(5):396–398, 1994.

[20] Lein Harn and Y Xu. Design of generalised ElGamal type digital signature schemes based on discrete logarithm. *Electronics letters*, 30(24):2025–2026, 1994.

[21] Sourav Sen Gupta. Blockchain. *IBM Online (http://www. IBM. COM)*, 2017.

[22] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck. Blockchain. *Business & Information Systems Engineering*, 59(3):183–187, 2017.

[23] Xiangyi Hu, Guifen Zhao, and Guanning Xu. Security scheme for online banking based on secret key encryption. In *2009 Second International Workshop on Knowledge Discovery and Data Mining*, pages 636–639, 2009.

[24] Young Sil Lee, Nack Hyun Kim, Hyotaek Lim, HeungKuk Jo, and Hoon Jae Lee. Online banking authentication system using mobile-OTP

with QR-code. In *5th International Conference on Computer Sciences and Convergence Information Technology*, pages 644–648, 2010.

[25] Gene Tsudik. Message authentication with one-way hash functions. *ACM SIGCOMM Computer Communication Review*, 22(5):29–38, 1992.

[26] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Annual international cryptology conference*, pages 1–15. Springer, 1996.

[27] Hugo Krawczyk. New hash functions for message authentication. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 301–310. Springer, 1995.

[28] James M Turner. The keyed-Hash Message Authentication Code (HMAC). *Federal Information Processing Standards Publication*, 198(1), 2008.

[29] Bernard Aboba, Larry Blunk, John Vollbrecht, James Carlson, Henrik Levkowetz, et al. Extensible authentication protocol (EAP). Technical report, RFC 3748, June, 2004.

[30] Larry Blunk and John Vollbrecht. PPP Extensible Authentication Arotocol (EAP). Technical report, RFC 2284, March, 1998.

[31] Jari Arkko, Henry Haverinen, et al. Extensible Authentication Protocol method for 3rd generation Authentication and Key Agreement (EAP-AKA). Technical report, RFC 4187, January, 2006.

[32] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably secure authenticated group Diffie-Hellman key exchange. *ACM Transactions on Information and System Security (TISSEC)*, 10(3):10–es, 2007.

[33] Wen-Shenq Juang. Efficient multi-server password authenticated key agreement using smart cards. *IEEE Transactions on Consumer Electronics*, 50(1):251–255, 2004.

[34] Russell Housley, Warwick Ford, William Polk, David Solo, et al. Internet X. 509 public key infrastructure certificate and CRL profile. Technical report, RFC 2459, January, 1999.

[35] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[36] George Robert Blakley. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*, pages 313–313. IEEE Computer Society, 1979.

[37] Carlisle Adams, Pat Cain, Denis Pinkas, and Robert Zuccherato. Internet X. 509 public key infrastructure time-stamp protocol (TSP). Technical report, RFC 3161, August, 2001.

[38] Mahmoud E Farfoura, Shi-Jinn Horng, Jui-Lin Lai, Ray-Shine Run, Rong-Jian Chen, and Muhammad Khurram Khan. A blind reversible method for watermarking relational databases based on a time-stamping protocol. *Expert Systems with Applications*, 39(3):3185–3196, 2012.

[39] Donald P Moynihan. Building secure elections: e-voting, security, and systems theory. *Public administration review*, 64(5):515–528, 2004.

[40] Josh D Cohen and Michael J Fischer. *A robust and verifiable cryptographically secure election scheme*. Yale University. Department of Computer Science, 1985.

[41] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, 22(5):563–591, 1980.

[42] Richard P Feynman. Simulating physics with computers. In *Feynman and computation*, pages 133–153. CRC Press, 2018.

[43] David Deutsch. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[44] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[45] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[46] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*, pages 212–219, 1996.

[47] Lov K Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review letters*, 79(2):325, 1997.

[48] Online. `https://www.rigetti.com`.

[49] Online. `https://www.xanadu.ai`.

[50] Online. `https://psiquantum.com`.

[51] Online. `https://www.honeywell.com/us/en/company/quantum`.

[52] Online. `https://ionq.com/`.

[53] Online. `https://www.dwavesys.com/`.

[54] Online. `https://qutech.nl/`.

[55] Online. `https://quantumbrilliance.com`.

[56] Farzan Jazaeri, Arnout Beckers, Armin Tajalli, and Jean-Michel Sallese. A review on quantum computing: From Qubits to front-end electronics and cryogenic MOSFET physics. In *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems"*, pages 15–25, 2019.

[57] K.K. Berggren. Quantum computing with superconductors. *Proceedings of the IEEE*, 92(10):1630–1638, 2004.

[58] Jeremy L O'brien, Akira Furusawa, and Jelena Vučković. Photonic quantum technologies. *Nature Photonics*, 3(12):687–695, 2009.

[59] Hartmut Häffner, Christian F Roos, and Rainer Blatt. Quantum computing with trapped ions. *Physics Reports*, 469(4):155–203, 2008.

[60] Zhenghan Wang. *Topological quantum computation.* Number 112. American Mathematical Soc., 2010.

[61] LCL Hollenberg, AS Dzurak, C Wellard, AR Hamilton, DJ Reilly, GJ Milburn, and RG Clark. Charge-based quantum computing using single donors in semiconductors. *Physical Review B*, 69(11):113301, 2004.

[62] Online. https://www.scottaaronson.com/blog/?p=1400.

[63] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[64] Michele Mosca. Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security Privacy*, 16(5):38–41, 2018.

[65] Tsuyoshi Takagi. *Post-Quantum Cryptography: 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, volume 9606. Springer, 2016.

[66] Online. https://pqcrypto2016.jp/.

[67] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology, 2016.

[68] Online. https://www.ledacrypt.org.

[69] Gorjan Alagic, Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. *Status report on the first round of the NIST post-quantum cryptography standardization process*. US Department of Commerce, National Institute of Standards and Technology, 2019.

[70] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the second round of the NIST post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2020.

[71] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pages 169–178, 2009.

[72] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[73] Dan Boneh, Yuval Ishai, Amit Sahai, and David J Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 247–277. Springer, 2017.

[74] Xavier Boyen. Attribute-based functional encryption on lattices. In *Theory of Cryptography Conference*, pages 122–142. Springer, 2013.

[75] Jiang Zhang, Zhenfeng Zhang, and Aijun Ge. Ciphertext policy attribute-based encryption from lattices. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 16–17, 2012.

[76] Miklós Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of computing*, pages 10–19, 1998.

[77] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM (JACM)*, 52(5):789–808, 2005.

[78] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of computing*, pages 333–342, 2009.

[79] Online. https://pq-crystals.org/kyber/.

[80] Online. https://www.esat.kuleuven.be/cosic/pqcrypto/saber/.

[81] Online. https://ntru.org/.

[82] Online. https://ntruprime.cr.yp.to/.

[83] Online. https://frodokem.org/.

[84] Online. https://pq-crystals.org/dilithium/.

[85] Online. https://falcon-sign.info/.

[86] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 419–453. Springer, 1988.

[87] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 33–48. Springer, 1996.

[88] Online. https://www.pqcrainbow.org/.

[89] Online. https://www-polsys.lip6.fr/Links/NIST/GeMSS.html.

[90] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 123–139. Springer, 1999.

[91] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 333–350. Springer, 2009.

[92] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-hashing for message authentication. Technical report, RFC 2104, February, 1997.

[93] Perry Metzger and William Simpson. IP authentication using keyed MD5. Technical report, RFC 1828, August, 1995.

[94] Cheryl Madson and Rob Glenn. The use of HMAC-SHA-1-96 within ESP and AH. Technical report, RFC 2404, November, 1998.

[95] Online. https://microsoft.github.io/Picnic/.

[96] Online. https://sphincs.org/.

[97] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.

[98] Online. https://sike.org/.

[99] Online. https://bikesuite.org/.

[100] Online. https://pqc-hqc.org/.

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

[102] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[103] A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.

[104] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[105] Yann Hamdaoui and Nicolas Sendrier. A non asymptotic analysis of information set decoding. *IACR Cryptol. ePrint Arch.*, 2013:162, 2013.

[106] Pil Joong Lee and Ernest F Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 275–280. Springer, 1988.

[107] Thammavarapu RN Rao and Kil-Hyun Nam. Private-key algebraic-coded cryptosystems. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 35–48. Springer, 1986.

[108] J.S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.

[109] Jacques Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.

[110] Anne Canteaut and Nicolas Sendrier. Cryptanalysis of the original McEliece cryptosystem. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 187–199. Springer, 1998.

[111] Florent Chabaud. On the security of some cryptosystems based on error-correcting codes. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 131–139. Springer, 1994.

[112] Thomas A Berson. Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. In *Annual International Cryptology Conference*, pages 213–220. Springer, 1997.

[113] Kazukuni Kobara and Hideki Imai. Semantically secure McEliece public-key cryptosystems-conversions for McEliece PKC. In *International Workshop on Public Key Cryptography*, pages 19–35. Springer, 2001.

[114] Edoardo Persichetti. On the CCA2 security of McEliece in the standard model. In *International Conference on Provable Security*, pages 165–181. Springer, 2018.

[115] K Preetha Mathew, Sachin Vasant, Sridhar Venkatesan, and C Pandu Rangan. An efficient IND-CCA2 secure variant of the Niederreiter encryption scheme in the standard model. In *Australasian Conference on Information Security and Privacy*, pages 166–179. Springer, 2012.

[116] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece: New McEliece variants from Moderate Density Parity-Check codes. In *2013 IEEE International Symposium on Information Theory*, pages 2069–2073, 2013.

[117] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. In *International conference on the theory and application of cryptology and information security*, pages 789–815. Springer, 2016.

[118] Jean-Pierre Tillich. The decoding failure probability of MDPC codes. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 941–945, 2018.

[119] Nicolas Sendrier and Valentin Vasseur. On the decoding failure rate of QC-MDPC bit-flipping decoders. In *International Conference on Post-Quantum Cryptography*, pages 404–416. Springer, 2019.

[120] Nir Drucker, Shay Gueron, and Dusan Kostic. On constant-time QC-MDPC decoding with negligible failure rate. *IACR Cryptol. ePrint Arch.*, 2019:1289, 2019.

[121] Nicolas Sendrier and Valentin Vasseur. On the existence of weak keys for QC-MDPC decoding. 2020.

[122] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[123] R. Townsend and E. Weldon. Self-orthogonal quasi-cyclic codes. *IEEE Transactions on Information Theory*, 13(2):183–195, 1967.

[124] Anatolii Karatsuba. Multiplication of multidigit numbers on automata. In *Soviet physics doklady*, volume 7, pages 595–596, 1963.

[125] Stephen A Cook and Stål O Aanderaa. On the minimum computation time of functions. *Transactions of the American Mathematical Society*, 142:291–314, 1969.

[126] Andrei L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics Doklady*, volume 3, pages 714–716, 1963.

[127] Shmuel Winograd. *Arithmetic complexity of computations*, volume 33. Siam, 1980.

[128] B. Vasic and O. Milenkovic. Combinatorial constructions of Low-Density arity-Check codes for iterative decoding. *IEEE Transactions on Information Theory*, 50(6):1156–1176, 2004.

[129] S.J. Johnson and S.R. Weller. A family of irregular LDPC codes with low encoding complexity. *IEEE Communications Letters*, 7(2):79–81, 2003.

[130] M Baldi and F Chiaraluce. New quasi cyclic Low Density Parity Check codes based on difference families. In *Proc. 8th Int. Symp. Commun. Theory and Appl., ISCTA*, volume 5, pages 244–249, 2005.

[131] Tao Xia and Bo Xia. Quasi-cyclic codes from extended difference families. In *IEEE Wireless Communications and Networking Conference, 2005*, volume 2, pages 1036–1040 Vol. 2, 2005.

[132] R. Gallager. Low-Density Parity-Check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962.

[133] Paolo Santini, Massimo Battaglioni, Marco Baldi, and Franco Chiaraluce. Hard-decision iterative decoding of LDPC codes with bounded error rate. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6, 2019.

[134] Paolo Santini, Massimo Battaglioni, Marco Baldi, and Franco Chiaraluce. Analysis of the error correction capability of LDPC and MDPC codes under parallel bit-flipping decoding and application to cryptography. *IEEE Transactions on Communications*, 68(8):4648–4660, 2020.

[135] P. Zarrinkhat and A.H. Banihashemi. Threshold values and convergence properties of majority-based algorithms for decoding regular Low-Density Parity-Check codes. *IEEE Transactions on Communications*, 52(12):2087–2097, 2004.

[136] Victor Vasilievich Zyablov and Mark Semenovich Pinsker. Estimation of the error-correction complexity for Gallager Low-Censity codes. *Problemy Peredachi Informatsii*, 11(1):23–36, 1975.

[137] David Burshtein. On the error correction of regular LDPC codes using the flipping algorithm. *IEEE Transactions on Information Theory*, 54(2):517–530, 2008.

[138] Shashi Kiran Chilappagari, Dung Viet Nguyen, Bane Vasic, and Michael W. Marcellin. On the guaranteed error correction capability of LDPC codes. In *2008 IEEE International Symposium on Information Theory*, pages 434–438, 2008.

[139] Shashi Kiran Chilappagari, Bane Vasic, and Michael W. Marcellin. Guaranteed error correction capability of codes on graphs. In *2009 Information Theory and Applications Workshop*, pages 50–55, 2009.

[140] Wen-Yao Chen and Chung-Chin Lu. On error correction capability of bit-flipping algorithm for LDPC codes. In *2011 IEEE International Symposium on Information Theory Proceedings*, pages 1283–1286, 2011.

[141] T.J. Richardson and R.L. Urbanke. The capacity of Low-Density Parity-Check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, 2001.

[142] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke. Design of capacity-approaching irregular Low-Density Parity-Check. *IEEE Transactions on Information Theory*, 47(2):619–637, 2001.

[143] L. Bazzi, T.J. Richardson, and R.L. Urbanke. Exact thresholds and optimal codes for the binary-symmetric channel and Gallager's decoding algorithm A. *IEEE Transactions on Information Theory*, 50(9):2010–2021, 2004.

[144] Aiden Price and Joanne Hall. A survey on trapping sets and stopping sets. *arXiv preprint arXiv:1705.05996*, 2017.

[145] Changyan Di, D. Proietti, I.E. Telatar, T.J. Richardson, and R.L. Urbanke. Finite-length analysis of Low-Density Parity-Check codes on the binary erasure channel. *IEEE Transactions on Information Theory*, 48(6):1570–1579, 2002.

[146] Junan Zhang and A. Orlitsky. Finite-length analysis of LDPC codes with large left degrees. In *Proceedings IEEE International Symposium on Information Theory*, 2002.

[147] A. Orlitsky, K. Viswanathan, and Junan Zhang. Stopping set distribution of LDPC code ensembles. *IEEE Transactions on Information Theory*, 51(3):929–953, 2005.

[148] M. Schwartz and A. Vardy. On the stopping distance and the stopping redundancy of codes. *IEEE Transactions on Information Theory*, 52(3):922–932, 2006.

[149] Juan Camilo Salazar Ripoll and Néstor R Barraza. A new algorithm to construct LDPC codes with large stopping sets. *Simulation*, 505:1, 2013.

[150] Morteza Esmaeili and Mohammad Javad Amoshahy. On the stopping distance of array code parity-check matrices. *IEEE Transactions on Information Theory*, 55(8):3488–3493, 2009.

[151] Matthew L. Grimes and David G. M. Mitchell. Design of nested protograph-based LDPC codes with low error-floors. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 575–579, 2018.

[152] Shashi Kiran Chilappagari, Michael Chertkov, Mikhail G. Stepanov, and Bane Vasic. Instanton-based techniques for analysis and reduction of error floors of LDPC codes. *IEEE Journal on Selected Areas in Communications*, 27(6):855–865, 2009.

[153] Nithin Raveendran, David Declercq, and Bane Vasić. A sub-graph expansion-contraction method for error floor computation. *IEEE Transactions on Communications*, 68(7):3984–3995, 2020.

[154] Peter Jephson Cameron, Jacobus Hendricus van Lint, et al. *Graphs, codes and designs*, volume 43. Cambridge University Press, 1980.

[155] Bane Vasić, Shashi Kiran Chilappagari, Dung Viet Nguyen, and Shiva Kumar Planjery. Trapping set ontology. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–7, 2009.

[156] Yoones Hashemi and Amir H. Banihashemi. New characterization and efficient exhaustive search algorithm for leafless elementary trapping sets of variable-regular LDPC codes. *IEEE Transactions on Information Theory*, 62(12):6713–6736, 2016.

[157] Yoones Hashemi and Amir H. Banihashemi. On characterization and efficient exhaustive search of elementary trapping sets of variable-regular LDPC codes. *IEEE Communications Letters*, 19(3):323–326, 2015.

[158] Qiuju Diao, Ying Yu Tai, Shu Lin, and Khaled Abdel-Ghaffar. Trapping set structure of finite geometry LDPC codes. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 3088–3092, 2012.

[159] Mehdi Karimi and Amir H. Banihashemi. On characterization of elementary trapping sets of variable-regular LDPC codes. *IEEE Transactions on Information Theory*, 60(9):5188–5203, 2014.

[160] Stefan Laendner, Olgica Milenkovic, and Johannes B. Huber. Characterization of small trapping sets in LDPC codes from Steiner triple systems. In *2010 6th International Symposium on Turbo Codes Iterative Information Processing*, pages 93–97, 2010.

[161] Christian Schlegel and Shuai Zhang. On the dynamics of the error floor behavior in (regular) LDPC codes. *IEEE Transactions on Information Theory*, 56(7):3248–3264, 2010.

[162] Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of symbolic computation*, 60:94–112, 2014.

[163] Zhengya Zhang, Lara Dolecek, Borivoje Nikolic, Venkat Anantharam, and Martin Wainwright. GEN03-6: Investigation of error floors of structured Low-Density Parity-Check Codes by hardware emulation. In *IEEE Globecom 2006*, pages 1–6, 2006.

[164] Lara Dolecek, Zhengya Zhang, Venkat Anantharam, Martin J. Wainwright, and Borivoje Nikolic. Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes. *IEEE Transactions on Information Theory*, 56(1):181–201, 2010.

[165] Lara Dolecek, Zhengya Zhang, Martin Wainwright, Venkat Anantharam, and Borivoje Nikolic. Evaluation of the low frame error rate performance of LDPC codes using importance sampling. In *2007 IEEE Information Theory Workshop*, pages 202–207, 2007.

[166] L. Dolecek, P. Lee, Zhengya Zhang, V. Anantharam, B. Nikolic, and M. Wainwright. Predicting error floors of structured LDPC codes: deterministic bounds and estimates. *IEEE Journal on Selected Areas in Communications*, 27(6):908–917, 2009.

[167] David G.M. Mitchell, Lara Dolecek, and Daniel J. Costello. Absorbing set characterization of array-based spatially coupled LDPC codes. In *2014 IEEE International Symposium on Information Theory*, pages 886–890, 2014.

[168] Massimo Battaglioni, Franco Chiaraluce, Marco Baldi, and David Mitchell. Efficient search and elimination of harmful objects in optimized QC SC-LDPC codes. *Globecom 2019*, 2019.

[169] Andrew McGregor and Olgica Milenkovic. On the hardness of approximating stopping and trapping sets. *IEEE Transactions on Information Theory*, 56(4):1640–1650, 2010.

[170] Gyu Bum Kyung and Chih-Chun Wang. Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder. In *2010 IEEE International Symposium on Information Theory*, pages 739–743, 2010.

[171] Gyu Bum Kyung and Chih-Chun Wang. Finding the exhaustive list of small fully absorbing sets and designing the corresponding low error-floor decoder. *IEEE Transactions on Communications*, 60(6):1487–1498, 2012.

[172] Emily McMillon, Allison Beemer, and Christine A. Kelley. Analysis of absorbing sets using cosets and syndromes. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 367–372, 2020.

[173] Roxana Smarandache and Pascal O. Vontobel. Pseudo-codeword analysis of Tanner graphs from projective and Euclidean planes. *IEEE Transactions on Information Theory*, 53(7):2376–2393, 2007.

[174] Michael Chertkov and Mikhail Stepanov. Pseudo-codeword landscape. In *2007 IEEE International Symposium on Information Theory*, pages 1546–1550, 2007.

[175] David JC MacKay and Michael S Postol. Weaknesses of Margulis and Ramanujan-Margulis Low-Density Parity-Check codes. *Electronic Notes in Theoretical Computer Science*, 74:97–104, 2003.

[176] Myungin Kim and Jeongseok Ha. Density evolution of the Bit-Flipping decoding algorithm of regular Low-Density Parity-Check codes. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 143–146, 2019.

[177] Myungin Kim and Jeongseok Ha. Analysis of Bit-Flipping algorithm of irregularLow-Density Parity-Check. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1456–1461, 2020.

[178] Satoshi Shimizu, Kazuaki Yamaguchi, and Sumio Masuda. A maximum edge-weight clique extraction algorithm based on branch-and-bound. *Discrete Optimization*, 37:100583, 2020.