Introduzione alla decodifica algebrica dei codici ciclici

L. Giuzzi

Brescia 29, 31 Maggio 2006

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Indice

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Capitolo 1

Burst di errore

1.1 Premessa

In alcuni sistemi di comunicazione, come ad esempio il canale binario simmetrico, la probabilità di errore è la medesima per ogni bit di informazione. In particolare, la presenza di un errore in una posizione i di un vettore ricevuto $\mathbf{r}$ non influenza minimamente la probabilità che si verifichi (o meno) un errore nella posizione $i+1$. In altri casi, gli errori tendono ad essere raggruppati in blocchi. Un esempio è quello dei graffi su di un disco digitale (CD e DVD, si vedano [25, 27, 26]): la presenza di un graffio è casuale, ma i bit che esso rende illeggibili risultano adiacenti fra loro. Un altro esempio è quello di un telefono cellulare che si trova ad uscire di copertura per un breve lasso di tempo (ad esempio poiché la persona che stava parlando si trova a passare innanzi un edificio che blocca parte del segnale): in questo caso è estremamente improbabile che solo un singolo bit sia cancellato ma (chiaramente) i dati danneggiati sono adiacenti fra loro.

Lo studio di tecniche per affrontare la correzione di burst di errore riveste pertanto una profonda valenza pratica. Ci sono due possibili approcci al problema:

a. l'utilizzo di codici ciclici opportuni;

b. la costruzione di codici intrecciati.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

1.2 Descrizione dei burst di errore

La nozione burst di errore è strettamente legata a quella di catena ciclica. Premettiamo due definizioni formali.

Definizione 1.1. Sia $\mathbf{v} = (v_0 v_1 \cdots v_{n-1})$ un vettore; due componenti v_i, v_j di $\mathbf{v}$ sono dette *ciclicamente consecutive* se $j = i + 1$ oppure $i = n - 1$ e $j = 0$. Una successione di $k \leq n$ componenti $v_{i_0}, v_{i_1}, \ldots v_{i_k}$ ciclicamente consecutive di un vettore $\mathbf{v}$ è detta *catena ciclica* di lunghezza k.

Definizione 1.2. Un errore di tipo *burst* (detto anche *burst di errore*) di lunghezza b è un vettore $\mathbf{l}$ le cui componenti non nulle sono contenute in una catena ciclica di lunghezza b. Una tale catena inizia e finisce sempre con una componente non nulla.

Due osservazioni sono opportune:

a. Una catena ciclica può andare oltre la fine della sequenza in cui compare, per continuare dall'inizio. Se ciò non accade, diremo un siffatto burst *non–ciclico*.

b. La prima e l'ultima posizione del vettore $\mathbf{l}$ sono sicuramente errate.

I burst di errore possono sempre essere descritti in forma "compressa" come segue. Supponiamo che $\mathbf{e}$ sia un vettore di errore che contiene un burst di errore.

Definizione 1.3. Il *formato* del burst in $\mathbf{e}$ è il sottovettore $\mathbf{p}$ di $\mathbf{e}$ formato dalle componenti non nulle di $\mathbf{e}$. La *lunghezza* di $\mathbf{p}$ sarà denotata con $|\mathbf{p}|$. La *posizione* l del burst è l'indice della prima componente non nulla di $\mathbf{e}$. Indicheremo un errore di formato $\mathbf{p}$ e posizione i col simbolo $(\mathbf{p}, i)$. Tale coppia ordinata è detta *descrizione di errore*

Notiamo che dalla conoscenza di $\mathbf{p}$ ed i è sempre possibile ricostruire il vettore $\mathbf{e}$.

Esempio 1.1. Osserviamo che, in generale, un vettore di errore può essere descritto in modi diversi. Sia ad esempio $\mathbf{e} = (01000\,00110)$. Sono possibili per $\mathbf{e}$ le tre descrizioni in Tabella 1.1. È immediato verificare che ogni vettore $\mathbf{e}$ di peso w ammette esattamente w descrizioni differenti.

Il Teorema 1.1 consente di ovviare all'inconveniente della mancanza di unicità della scrittura che si è visto nell'Esempio 1.1. Prima di procedere alla dimostrazione è bene premettere una definizione e un lemma.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Formato	Posizione	Lunghezza
100 00011	1	8
11001	6	5
10010 00001	7	10

Tabella 1.1: Descrizioni possibili per un burst di errore di peso 3

Definizione 1.4. Sia $\mathbf{e}$ un errore e sia $(\mathbf{p}, i)$ una descrizione di errore. Le componenti di $\mathbf{e}$ non in $\mathbf{p}$ formano una catena ciclica di 0, che inizia (ciclicamente) nella posizione $i + |\mathbf{p}| + 1$ e finisce nella posizione $i - 1$. L'insieme degli indici corrispondenti a tale catena è detto *catena zero* di $(\mathbf{p}, i)$.

La descrizione degli errori è essenzialmente unica, come enunciato nel seguente teorema.

Teorema 1.1. *Sia* $\mathbf{e}$ *un vettore di errore di lunghezza* n *con due descrizioni di errore del tipo* $(\mathbf{p_1}, i_1)$ *e* $(\mathbf{p_2}, i_2)$. *Allora, se* $|\mathbf{p_1}| + |\mathbf{p_2}| \leq n + 1$ *abbiamo* $(\mathbf{p_1}, i_1) = (\mathbf{p_2}, i_2)$.

Conseguenza immediata del Teorema 1.1 è il seguente corollario.

Corollario 1.2. *Ogni vettore di errore* $\mathbf{e}$ *ammette al più una descrizione come burst di errore di lunghezza minore o uguale a* $(n + 1)/2$.

Si osservi che è comunque possibile avere burst di errore che non ammettono tale descrizione di errore. Questo è il caso, ad esempio, di tutti quegli errori che hanno peso strettamente maggiore di $(n + 1)/2$.

1.3 Limitazioni

In generale, la capacità correttiva di un codice lineare è legata al numero di sindrome diverse disponibili. In particolare, un $[n, k, d]$–codice su $\mathbb{F}_q$ può, a priori, correggere al più q^{n-k} diverse tipologie di errore.

Rammentiamo che, posto $t = \lfloor (d - 1)/2 \rfloor$ si può dire che ogni codice di distanza minima d è sicuramente in grado di correggere t errori casuali. In questo paragrafo forniremo delle limitazioni sul numero b di burst di errore che un tale codice è in grado di correggere.

Un burst di errore di b bit è descritto mediante un formato di errore che è una catena ciclica di lunghezza b. Al fine di determinare quanto lungo un

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

burst correggibile possa essere è dunque necessario calcolare il numero di tutte le catene cicliche di lunghezza n. Per semplicità ci limitiamo al caso binario.

Teorema 1.3. *Fissato n, sia b un intero tale che $1 \leq b \leq (n+1)/2$. Allora, esistono esattamente $n2^{b-1}+1$ vettori di lunghezza n su $\mathbb{F}_2$ che contengono catene cicliche di lunghezza al più b.*

Il seguente teorema è una riformulazione della limitazione di Hamming per codici binari che correggano burst di errore alla luce del risultato precedente. Osserviamo che fra le ipotesi non si richiede che il codice sia lineare.

Teorema 1.4 (Limitazione di Hamming per burst di errore). *Sia $1 \leq b \leq (n+1)/2$. Un codice binario $\mathcal{C}$ di lunghezza n che corregge tutti gli burst di errore di lunghezza al massimo b contiene al più $2^n/(n2^{b-1}+1)$ parole.*

Dimostrazione. Per il Teorema 1.3 vi sono esattamente $n2^{b-1}+1$ possibili vettori di errore burst di peso non superiore a b. Siano M il numero delle parole di codice di $\mathcal{C}$. Il numero di parole che differiscono da un elemento di $\mathcal{C}$ in un vettore ciclico di peso al più b è dunque $M(n2^{b-1}+1)$. Poiché tutti tali vettori si trovano in $V^n(\mathbb{F}_2)$, ne segue che $M(n2^{b-1}+1) \leq 2^n$. □

L'analogo della limitazione di Singleton per codici b–correttori è contenuta nel seguente teorema.

Teorema 1.5 (Limitazione di Reiger). *Sia $0 \leq b \leq n/2$. Un $[n,k]$–codice binario $\mathcal{C}$ in grado di correggere burst di errore di lunghezza al più b deve soddisfare*

$$r \geq 2b,$$

ove $r = n-k$ è la ridondanza di $\mathcal{C}$.

1.4 Campi di ordine non primo

I campi finiti più facili da implementare sono quelli $\mathbb{F}_p \simeq \mathbb{Z}_p$, contenenti un numero primo di elementi. Infatti, ogni elemento di tali campi è rappresentato da esattamente un intero z con $0 \leq z \leq p-1$ e le operazioni di somma e prodotto fra due elementi $a, b \in \mathbb{F}_p$ possono essere implementate semplicemente moltiplicando (o sommando) fra loro gli interi che rappresentano gli elementi a, b e riducendo il risultato modulo p.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

L'implementazione di $\mathbb{F}_q$, ove $q = p^h$ è una potenza non banale di un primo p presenta maggiori difficoltà, sia da un punto di vista teorico (è necessario determinare un opportuno polinomio irriducibile di grado h su $\mathbb{F}_p$) che pratico. In particolare, difficilmente un microprocessore non dedicato possiede una versione ottimizzata delle operazioni di prodotto che servono.

I codici correttori su campi di ordine non primo, d'altro canto, godono di alcune importanti proprietà ne che giustificano lo studio. In effetti, molti dei codici implementati concretamente in applicazioni di tipo commerciale (CD, DVD, Televisione Digitale, etc.) sono definiti su campi di questo tipo. Questo è il caso, ad esempio, dei codici di Reed–Solomon, che saranno studiati nel Capitolo 3.

In questo paragrafo mostreremo uno dei vantaggi più importanti che si hanno lavorando con un codice $\mathcal{C}$ definito su di un campo di ordine composto: il fatto che $\mathcal{C}$ risulta in modo naturale in grado di correggere un numero di errori ciclici molto più elevato rispetto quanto ci si potrebbe attendere a partire dalla distanza minima.

Siano dunque q una potenza di primo e $m \geq 1$ un intero.

Teorema 1.6. *Dato un* $[n, k]$*–codice* $\mathcal{C}$ *su* $\mathbb{F}_{q^m}$ *in grado di correggere errori burst di lunghezza al più* b*, esiste sempre un* $[mn, mk]$ *codice* $\mathcal{C}'$ *su* $\mathbb{F}_q$ *in grado di correggere almeno* $(m-1)b+1$ *burst di errore.*

Dimostrazione. Fissiamo una base $\mathfrak{B}$ di $\mathbb{F}_{q^m}$, visto come spazio vettoriale su $\mathbb{F}_q$ di dimensione m.

Consideriamo il $[mn, mk]$–codice $\mathcal{C}'$ su $\mathbb{F}_q$ ottenuto a partire da $\mathcal{C}$ sostituendo ad ogni elemento del codice originario il vettore su $\mathbb{F}_q$ che lo rappresenta rispetto $\mathfrak{B}$, e sia $\pi : \mathcal{C} \mapsto \mathcal{C}'$ l'applicazione che realizza questa trasformazione. Il teorema ora segue osservando che un vettore $\mathbf{r}' \in \mathcal{C}'$, contenente un burst di errore con lunghezza al più $(m-1)b+1$, viene trasformato da π^{-1} in un vettore $\mathbf{r} \in \mathcal{C}$, contenente un burst di errore di lunghezza al più b. La situazione è illustrata nella Figura 1.1. Ne segue che il vettore $\mathbf{r}$ può essere corretto in un vettore $\mathbf{c}$ e che $\mathbf{c}' = \pi(\mathbf{c})$ è la parola corretta in $\mathcal{C}'$. □

È importante osservare che l'ipotesi che gli errori siano burst è essenziale per la dimostrazione del Teorema 1.6. Infatti, la distanza minima di $\mathcal{C}$ e quella di $\mathcal{C}'$ coincidono, per cui il numero t di errori generici che entrambi i codici possono correggere è il medesimo. Come suggerisce la Figura 1.1, quando m è abbastanza grande non è necessario che il codice di partenza su $\mathbb{F}_{q^m}$ possegga proprietà particolari per poter correggere numerosi burst di errore.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

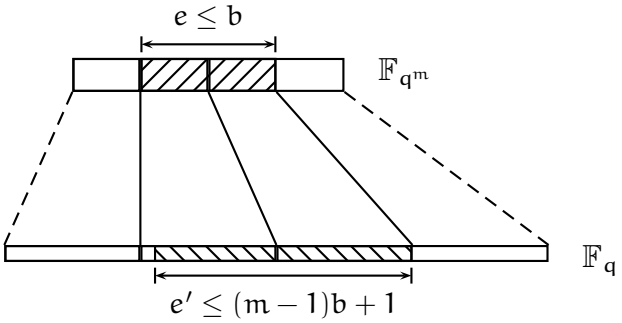

Figura 1.1: Codici q^m–ari e codici q–ari

Per concludere, osserviamo che, in generale, un codice t–correttore lineare può essere utilizzato per correggere t errori oppure per identificarne $d-1=2t$, ove d è la distanza minima. Le due modalità non possono solitamente essere impiegate contemporaneamente, in quanto in presenza di un numero $h > t$ di errori l'algoritmo di decodifica che cerca la parola di codice più vicina al vettore ricevuto fornirebbe un risultato errato. Per gli errori ciclici è talvolta possibile fare di meglio, utlizzando dei codici ciclici opportuni, come si vedrà nel seguente capitolo.

1.5 Codici prodotto

Una generalizzazione della nozione di codice in serie è quella di codice prodotto. Tale nozione richiede di applicare più operazioni di codifica in serie su opportuni insiemi di parole. È importante osservare che in tale costruzione entrambi i codici C_1 e C_2 costituenti possono avere parametri arbitrari.

Siano dunque $[n_1, k_1]$, $[n_2, k_2]$ rispettivamente i parametri di C_1 e C_2 Le costruzione prodotto corrisponde a codificare k_2 parole di C_1 in parallelo e poi ad applicare a tali parole n_1 codifiche in parallelo mediante C_2, come illustrato in Figura 1.2. La struttura delle parole del codice così ottenuto è mostrata in Figura 1.3.

Definizione 1.5. Siano A, B due matrici rispettivamente $m_1 \times n_1$ e $m_2 \times n_2$. Il *prodotto secondo Kronecker* o *prodotto diretto* $A \otimes B$ di A con B è la matrice $n_1 n_2 \times t_1 t_2$ ottenuta sostituendo ogni entrata a_{ij} di A con la matrice $a_{ij}B$.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Esempio 1.2. Siano $A = \left[\begin{array}{c|c} 1 & 2 \end{array}\right]$ e $B = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$. Allora,

$$A \otimes B = \left[\begin{array}{cc|cc} 1 & 0 & 2 & 0 \\ 1 & 2 & 2 & 4 \end{array}\right].$$

Ricordiamo che il prodotto tensoriale di due spazi vettoriali V, W è lo spazio vettoriale $V \otimes W$ generato da tutti gli elementi della forma $v \otimes w$ con $v \in V_1$, $w \in V_2$ che soddisfano le regole

$$(v_1 + v_2) \otimes w = v_1 \otimes w + v_2 \otimes w,$$

$$v \otimes (w_1 + w_2) = v \otimes w_1 + v \otimes w_2,$$

$$\alpha(v \otimes w) = (\alpha v) \otimes w = v \otimes (\alpha w).$$

Da queste tre affermazioni segue

$$v \otimes w = 0 \text{ se, e soltanto se, } v = 0 \text{ oppure } w = 0.$$

Osserviamo in particolare che se C_1 è un codice con matrice generatrice G_1 e C_2 è un codice con matrice generatrice G_2, allora lo spazio vettoriale $C_1 \otimes C_2$ ha matrice generatrice esattamente $G_1 \otimes G_2$.

Forniamo ora una definizione formale di codice prodotto.

Definizione 1.6. Siano C_1, C_2 due codici con matrici generatrici rispettivamente G_1, G_2. Il *codice prodotto*[1] $C_1 \otimes C_2$ è il codice la cui matrice generatrice è $G_1 \otimes G_2$.

Teorema 1.7. *Siano C_1 e C_2 rispettivamente un $[n_1, k_1, d_1]$ e un $[n_2, k_2, d_2]$ codice lineare. Allora il codice prodotto $C_1 \otimes C_2$ ha parametri $[n_1 n_2, k_1 k_2, d_1 d_2]$.*

Dimostrazione. È immediato vedere che la lunghezza di $C = C_1 \otimes C_2$ è $n_1 n_2$. Inoltre ogni parola di $C_1 \otimes C_2$ è della forma $v_1 \otimes v_2$ con $v_1 \in C_1$ e $v_2 \in C_2$. In particolare una parola ha peso minimo in $v_1 \otimes v_2 \in C_1 \otimes C_2$ ha peso minimo se, e soltanto se, v_1 ha peso minimo in C_1 e v_2 ha peso minimo in C_2. A questo punto è immediato vedere che

$$w(v_1 \otimes v_2) = w(v_1) w(v_2) = d_1 d_2,$$

per cui la distanza minima di $C_1 \otimes C_2$ è $d_1 d_2$. Infine, come sopra osservato, tutte le righe della matrice $G_1 \otimes G_2$ sono linearmente indipendenti, per cui la dimensione del codice prodotto è esattamente $k_1 k_2$. □

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

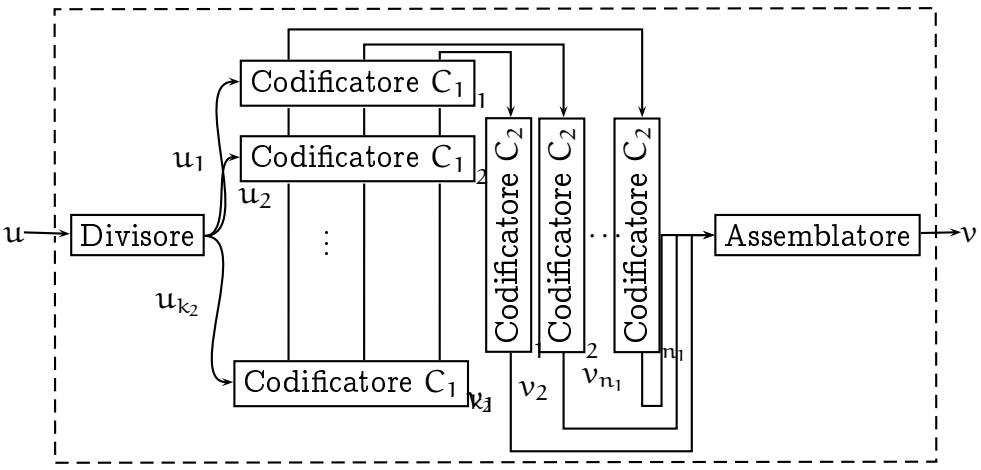

Figura 1.2: Codificatore prodotto

Teorema 1.8. *Siano* C_1, C_2 *come sopra e si ponga* $t_i = \lfloor (d_i - 1)/2 \rfloor$. *Il codice prodotto* $C = C_1 \otimes C_2$ *è in grado di correggere tutte le* sequenze di errori burst *di lunghezza al più*

$$b = \max\{n_1 t_2, n_2 t_1\}.$$

Dimostrazione. Osserviamo che, per ogni $0 \leq j \leq n_2 - 1$, al variare di k le componenti $(j + kn_1)$–esime di una parola di codice di C sono esattamente n_2 e formano una parola di codice per C_2. Il codice C_2 può essere usato per correggere t_2 errori in ognuna di queste n_1 parole. Ne segue che, qualora gli errori siano di tipo burst, è possibile correggerne $t_2 n_1$. Un ragionamento analogo mostra come usando C_2 sia possibile correggere $t_1 n_2$ burst di errore, da cui si determina il valore di b. □

Esempio 1.3. Siano C_1 e C_2 due copie del codice di Hamming di parametri $[7, 4, 3]$. Allora $C_1 \otimes C_2$ è un $[49, 16, 9]$–codice lineare capace di correggere 4 errori in qualsivoglia posizione e al più 7 burst di errore.

1.6 Interleaving

Un caso particolare della costruzione prodotto vista nel Paragrafo 1.5 è quello dell'*interleaving* di codici.

Il *codice banale* di lunghezza n su $\mathbb{F}_q$ è l'unico codice di parametri $[n, n, 1]$. Denoteremo tale codice col simbolo $\mathcal{I}_{(n)}$.

[1] direct product

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

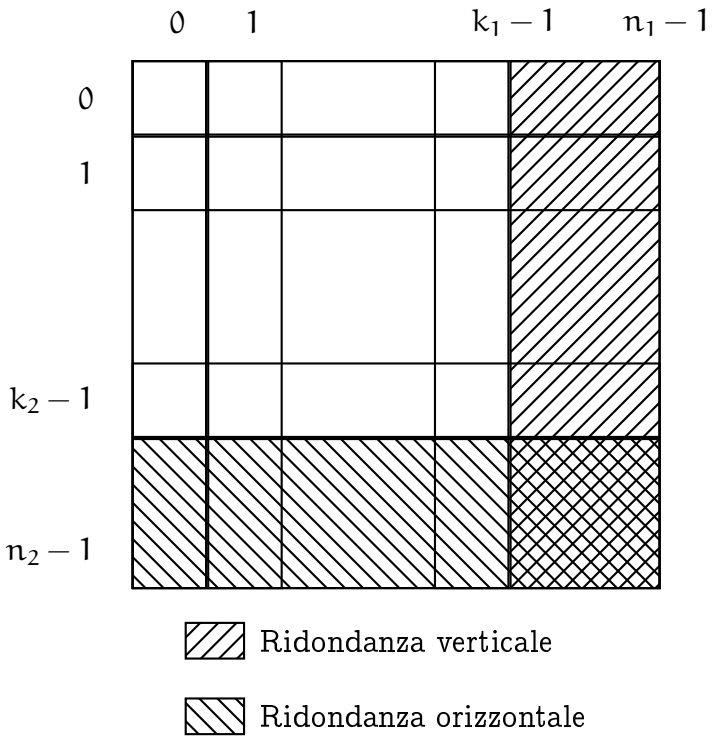

Figura 1.3: Struttura di una parola nel codice prodotto

Definizione 1.7. Si dice *block interleaved code* di *profondità* o *grado* n ottenuto a partire da C il codice $C^{(n)}$ dato da

$$C^{(n)} = C \otimes \mathcal{I}_{(n)}.$$

Osserviamo che nel caso di un codice n_2–interleaved ottenuto a partire da un codice $\mathcal{C}$ di parametri $[n_1, k_1]$, le parole vengono ad essere disposte in una matrice $n_2 \times n_1$, come illustrato in Figura 1.4, ove

$$\mathbf{v_i} = (v_{i,1}, v_{i,2}, \ldots, v_{i,n_1})$$

è una parola di C per $1 \leq i \leq n_2$ L'importanza dei codici con interleaving risiede nel fatto che è possibile utilizzare per $\mathcal{C}^{(n_2)}$ il medesimo algoritmo di decodifica che per $\mathcal{C}$. In particolare, sempre facendo riferimento alla Figura 1.4 le componenti di $\mathcal{C}^{(n_2)}$ vengono trasmesse per colonna, cioè nell'ordine

$$\mathbf{v} = (v_{1,1}, v_{2,1}, \cdots, v_{n_2,0}, v_{1,2}, \cdots, v_{1,n_1}, \cdots, v_{n_2,n_1}).$$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

$$\begin{array}{cccc} v_{1,1} & v_{1,2} & \cdots & v_{1,n_1} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,n_1} \\ \vdots & & & \vdots \\ v_{n_2,0} & v_{n_2,1} & \cdots & v_{n_2,n_1} \end{array}$$

Figura 1.4: Struttura di un codice con interleaving

In fase di decodifica si deve dunque innanzi tutto ricostruire la tabella colonna per colonna e, successivamente, procedere a decodificare e correggere riga per riga.

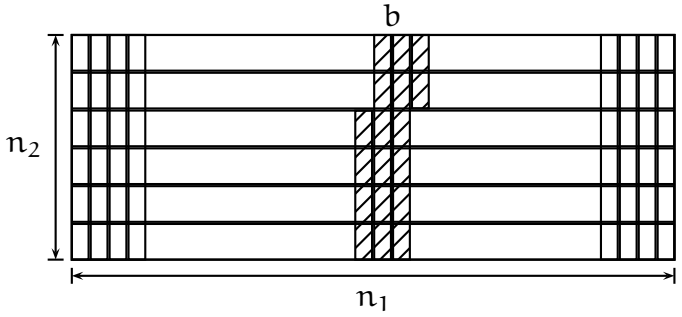

Figura 1.5: Burst di errore e codici con interleaving

Un codice con interleaving $\mathcal{C}^{(n)}$ possiede esattamente la stessa distanza minima del codice di partenza $\mathcal{C}$, per cui esso è in grado di correggere esattamente lo stesso numero di errori generici. D'altro canto tali codici risultano estremamente pratici per correggere burst di errore. Supponiamo in particolare che si verifichino b errori in sequenza in un messaggio ricevuto $\mathbf{r}$. La situazione è quella descritta in Figura 1.5 per un codice $\mathcal{C}^{(n_2)}$, ove $\mathcal{C}$ ha lunghezza n_1 e raggio di impacchettamento e. Osserviamo che in ogni riga solamente b/n_2 componenti risultano alterate. Ne risulta che se $b < en_2$, allora ogni riga, che è un vettore di lunghezza n_1 su cui applichiamo il codice $\mathcal{C}$, contiene un numero di errori inferiore ad e e dunque, mediante $\mathcal{C}$ è possibile ricostruirne i valori originari.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Capitolo 2

Codici ciclici

Un codice $\mathcal{C}$ è univocamente identificato dall'elenco delle parole che esso contiene; d'altro canto questo non è di per sè sufficiente per implementare praticamente lo stesso.

Infatti, l'implementazione concreta di un codice correttore consiste in almeno tre distinti algoritmi:

1. Un algoritmo di codifica;

2. Un algoritmo per individuare e eventualmente correggere gli errori;

3. Un algoritmo di decodifica.

Osserviamo che, a priori, non è necessario essere in grado scrivere la lista di tutte le parole in $\mathcal{C}$, anche se questo può aiutare molto nel determinarne le proprietà.

Un generico codice a blocchi, privo di ulteriore struttura può essere fornito solamente elencando tutti i suoi elementi. Per caratterizzare tutte le parole di un $[n, k]$–codice lineare $\mathcal{C}$, al contrario, basta fornire k vettori distinti che formino una base dello stesso.

In questo capitolo considereremo una classe particolare di codici lineari: i codici ciclici. Questa famiglia di $[n, k]$–codici lineari gode di notevoli proprietà, fra cui:

a. Sono costruiti a partire da un singolo vettore;

b. Ammettono degli algoritmi di decodifica efficienti;

c. Risultano particolarmente comodi per correggere alcune tipologie di errore;

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

d. Possono essere descritti ed enumerati in modo esaustivo mediante la loro relazione con struttrure algebriche.

2.1 Sottospazi ciclici

Iniziamo questo paragrafo considerando un tipo particolare di sottospazi di uno spazio vettoriale V.

Definizione 2.1. Un sottospazio S di $V_n(\mathbb{F})$ è detto *sottospazio ciclico* se comunque dato

$$(a_0 a_1 \dots a_{n-2} a_{n-1}) \in S$$

si ha sempre

$$(a_{n-1} a_0 a_1 \dots a_{n-2}) \in S$$

Osserviamo che tutto lo spazio $V_n(\mathbb{F})$, così come il sottospazio $\{\mathbf{0}\}$ formato dal solo vettore nullo sono sottospazi ciclici. Tali sottospazi ciclici sono detti *banali*.

Sia $\mathbf{v} = (v_0 v_1 \dots v_{n-1}) \in V$ un vettore e sia $\sigma = (1\,2 \dots n-1)$. Un vettore $\mathbf{w} \in V$ è una *permutazione ciclica* di $\mathbf{v}$ se esiste un intero $i \in \{0, 1, \dots n-1\}$ tale che

$$\mathbf{w} = (v_{\sigma^i(0)} v_{\sigma^i(1)} \dots v_{\sigma^i(n-1)}) = \sigma^i(\mathbf{v}).$$

In particolare, in questo caso il vettore $\mathbf{w}$ può scriversi come

$$\mathbf{w} = (v_i v_{i+1} \dots v_{i+n-1}),$$

ove gli indici sono da intendersi ridotti modulo n.

Teorema 2.1. *Sia S un sottospazio ciclico di uno spazio vettoriale V. Allora, S contiene tutte le possibili permutazioni cicliche di un suo qualsiasi elemento.*

Dimostrazione. Sia $\mathbf{s} = (s_0 s_1 \dots s_{n-1}) \in S$ e fissiamo $i \in \{0, 1, \dots n-1\}$. Chiaramente $\sigma^0(\mathbf{s}) = \mathbf{s} \in S$; per definizione di sottospazio ciclico si ha che anche $\sigma^1(\mathbf{s}) \in S$. D'altro canto se $\sigma^i(\mathbf{s}) \in S$, allora $\sigma^{i+1}(\mathbf{s}) = \sigma(\sigma^i(\mathbf{s})) \in S$. Ne segue, per induzione, che tutte le possibili permutazioni cicliche di $\mathbf{s}$ appartengono ad S. □

I codici ciclici sono dei casi particolari dei codici lineari.

Definizione 2.2. Un *codice ciclico* sopra $\mathbb{F}$ è un codice lineare su $\mathbb{F}$ che è al contempo un sottospazio ciclico di $V_n(\mathbb{F})$.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

2.2 Rappresentazione degli spazi ciclici

Il metodo più conveniente per rappresentare un sottospazio ciclico C è quello di considerarlo come un opportuno insieme di polinomi.

In particolare, per ogni vettore

$$\mathbf{c} = (c_0\, c_1 \cdots c_{n-1}) \in C,$$

è sempre possibile scrivere un polinomio

$$c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$$

che lo rappresenta. Lo spazio vettoriale $V_n(\mathbb{F}_q)$, di dimensione n è dunque rappresentato dall'insieme di tutti i polinomi in x su $\mathbb{F}_q$ aventi grado al più $n-1$.

È possibile introdurre in $V_n(\mathbb{F}_q)$ una operazione di moltiplicazione: il prodotto di due vettori è determinato calcolando il prodotto usuale fra i polinomi associati e, successivamente calcolandone il resto quando li si divide per (x^n-1). La struttura algebrica $\mathcal{C}$ così costruita è l'*anello quoziente* di $\mathbb{F}[x]$ rispetto l'ideale generato dal polinomio (x^n-1).

Per un fondamentale teorema di natura algebrica questo anello è 1–generato; in altre parole esiste un elemento $g(x) \in \mathcal{C}$ tale che le permutazioni cicliche di $g(x)$ costituiscono un insieme di generatori per $\mathcal{C}$.

In termini più formali, sia $\mathcal{C}$ un codice ciclico diverso da $\{\mathbf{0}\}$. Allora

1. esiste un unico polinomio monico $g(x)$ avente grado minimo r in $\mathcal{C}$, $\dim\mathcal{C} = n-r$ e inoltre
$$\mathcal{C} = \{g(x)q(x) : q(x) \in \mathbb{F}[x]; \deg q(x) < n-r\}.$$
2. Il polinomio $g(x)$ divide x^n-1 in $\mathbb{F}[x]$.

Il seguente teorema caratterizza tutti i codici ciclici di lunghezza n.

Teorema 2.2. *Vi è una corrispondenza biunivoca fra i sottospazi ciclici di* $V_n(\mathbb{F})$ *e i polinomi monici* $g(x) \in \mathbb{F}[x]$ *che sono divisori di* $f(x) = x^n-1$.

Esempio 2.1. Consideriamo tutti i possibili codici ciclici binari di lunghezza 7. Tali codici sono dunque sottospazi di $V_7(\mathbb{Z}_2)$. Poniamo $f(x) = x^7-1$. In $\mathbb{Z}_2$ tale polinomio $f(x)$ si fattorizza in irriducibili come

$$x^7-1 = (x+1)(x^3+x^2+1)(x^3+x+1),$$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

pertanto i divisori monici di $f(x)$ sono tutti e soli i seguenti

$$\begin{aligned}
g_1(x) &= 1\\
g_2(x) &= x+1\\
g_3(x) &= x^3+x^2+1\\
g_4(x) &= x^3+x+1\\
g_5(x) &= (x+1)(x^3+x^2+1)\\
g_6(x) &= (x+1)(x^3+x+1)\\
g_7(x) &= (x^3+x^2+1)(x^3+x+1)\\
g_8(x) &= f(x).
\end{aligned}$$

Ne segue che $V_7(\mathbb{Z}_2)$ contiene esattamente 8 sottospazi ciclici.

A titolo di esempio, osserviamo che il polinomio $g_6(x)$ genera il sottospazio ciclico

$$\begin{aligned}
S = \{&(0000000), (1011100), (0101110), (0010111)\\
&(1001011), (1100101), (1110010), (0111001)\}.
\end{aligned}$$

Similmente, il polinomio $g_7(x)$ genera il sottospazio ciclico

$$S = \{(0000000), (1111111)\}.$$

2.3 Matrice generatrice di un codice ciclico

Osserviamo che la forma più naturale per la matrice generatrice di un codice ciclico è quella *a scala*[1]. In particolare, la matrice quadrata ottenuta considerando le prime $k = n - r$ colonne è triangolare superiore e ha determinante non nullo. Pertanto, le prime k posizioni bastano a ricostruire, in assenza di errori, la parola originariamente trasmessa e sono dunque di informazione. D'altro canto questa forma ciclica non è quasi mai standard, per cui la codifica non risulta sistematica.

Un'altra forma "interessante," che è sempre possibile dare alla matrice generatrice, è quella "*antisistematica*", ottenuta mediante la costruzione descritta qui di seguito. Tale forma si rivela particolarmente comoda in fase di implementazione di algoritmi di decodifica, in quanto una copia della parola originale si ritrova, in questo caso, nelle ultime k posizioni trasmesse. Sia $\mathcal{C}$ un $[n, k]$–codice ciclico

[1] *row–echelon*

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

sopra $\mathbb{F}$ con polinomio generatore $g(x)$. Il nostro obiettivo è dunque quello di determinare per $\mathcal{C}$ una matrice generatrice della forma $G = (R\, I_k)$.

Procediamo ora alla costruzione della matrice G.

1. Per ogni $i = 0, 1, \ldots, k-1$, si divida x^{n-k+i} per $g(x)$, cosicché
$$x^{n-k+i} = q_i(x)g(x) + r_i(x),$$
ove $\deg r_i(x) < \deg g(x) = n - k$ oppure $r_i(x) = 0$;

2. Si ponga
$$p_i(x) = x^{n-k+i} - r_i(x) = q_i(x)g(x) \in C;$$

3. Osserviamo che
$$\deg p_i(x) - \deg^0 p_i(x) \geq i;$$

4. Ne segue che considerando i coefficienti di $p(x) = x^{n-k+i} - r_i(x)$ come righe di una matrice, otteniamo una struttura del tipo
$$\left(\begin{array}{c|cccc} \overset{0}{} \quad \overset{r-1}{} & \overset{r}{} & & & \overset{n-1}{} \\ \boxed{-r_0(x)} & 1 & 0 & \ldots & 0 \\ \boxed{-r_1(x)} & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \boxed{-r_{k-1}(x)} & 0 & \ldots & 0 & 1 \end{array}\right)$$
ovvero una matrice $G = (R\, I_k)$ in cui le righe di R corrispondono a $-r_i(x)$ con $0 \leq i \leq k-1$;

5. Ogni riga di G è una parola di $\mathcal{C}$; inoltre le righe sono tutte fra loro linearmente indipendenti, per cui G è una matrice generatrice per il codice $\mathcal{C}$ avente la forma desiderata.

Esempio 2.2. Si consideri nuovamente il $[7, 4]$–codice ciclico generato dal polinomio $1 + x + x^3$. Per mezzo dell'algoritmo euclideo, determiniamo

$$\begin{aligned} x^3 &= (1)(x^3 + x + 1) + (1 + x) \\ x^4 &= (x)(x^3 + x + 1) + (x + x^2) \\ x^5 &= (x^2 + 1)(x^3 + x + 1) + (1 + x + x^2) \\ x^6 &= (x^3 + x + 1)(x^3 + x + 1) + (1 + x^2), \end{aligned}$$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

In tal modo resta definita la seguente matrice generatrice:

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} = (R\, I_4), \quad \text{con } R = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

Come già osservato, le righe di R corrispondono ai vettori dei coefficienti dei polinomi $1 + x$, $x + x^2$, $1 + x + x^2$ ed $1 + x^2$. Una sequenza di informazione $\mathbf{m} = (1011)$ viene codificata mediante G come $\mathbf{c} = \mathbf{m}G = (100\,1011)$.

2.4 Polinomio di controllo di parità

La matrice generatrice di un codice ciclico può essere costruita a partire da un singolo polinomio $g(x)$. Similmente è possibile determinare una matrice di controllo di parità, utilizzando un altro polinomio.

Definizione 2.3. Sia $h(x) = \sum_{i=0}^{k} a_i x^i$, con $\deg h(x) = k$. Si dice *polinomio reciproco* $h_R(x)$ di $h(x)$ il polinomio dato da

$$h_R(x) = \sum_{i=0}^{k} a_{k-i} x^i.$$

È immediato vedere che, poiché $\deg h(x) = k$, il polinomio $h_R(x)$ può sempre scriversi formalmente come $x^k h(1/x)$.

Definizione 2.4. Dato un codice ciclico $\mathcal{C}$, il *codice invertito* $\mathcal{C}^R$ di $\mathcal{C}$ è il codice ottenuto da $\mathcal{C}$ invertendo ogni parola di codice.

In particolare, $(c_0\, c_1 \ldots c_{n-1}) \in \mathcal{C}$ se, e soltanto se, $(c_{n-1}\, c_{n-2} \ldots c_1\, c_0) \in \mathcal{C}^R$. Notiamo che se $g(x)$ è un generatore per $\mathcal{C}$, allora $g_0^{-1} g_R(x)$ è un generatore monico per $\mathcal{C}^R$.

Definizione 2.5. Sia $\mathcal{C}$ un codice ciclico con polinomio generatore $g(x)$. Il *polinomio di controllo di parità* di $\mathcal{C}$ è il polinomio $h(x)$ tale che

$$g(x)h(x) = x^n - 1.$$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Osserviamo che in $\mathbb{F}[x]_n$ si ha

$$(g(x) + (x^n - 1))\,(h(x) + (x^n - 1)) = 0 + (x^n - 1).$$

Il seguente teorema fornisce la motivazione per il nome dato al polinomio $h(x)$ nella Definizione 2.5.

Teorema 2.3. *Sia* $g(x) \in \mathbb{F}[x]$ *un divisore monico di grado* $n-k$ *di* $f(x) = x^n - 1$ *e sia* $\mathcal{C}$ *il* $[n,k]$*–codice ciclico che esso genera. Denotiamo con* $h(x) = (x^n-1)/g(x)$ *il polinomio di controllo di parità di* $\mathcal{C}$*. Allora,*

$$\mathcal{C} = \{c(x) \in \mathbb{F}[x] : \deg c(x) \leq n, c(x)h(x) = 0 \pmod{x^n - 1}\}.$$

In altre parole, le parole $c(x)$ del codice C sono univocamente determinate dal fatto che il prodotto $c(x)h(x)$ è divisibile per $x^n - 1$.

È ora possibile mostrare come i concetti di polinomio generatore, polinomio reciproco e polinomio di controllo di parità sono strettamente associati a quello di codice ortogonale. In tale modo si potrà fornire un metodo diretto per determinare $h(x)$.

Teorema 2.4. *Sia* $g(x)$ *il polinomio generatore di un* $[n, n-r]$*–codice ciclico* $\mathcal{C}$ *sopra* $\mathbb{F}$*. Il codice ortogonale* $\mathcal{C}^\perp$ *è il codice ciclico generato da* $h_R(x)$ *ove* $h(x)$ *è il polinomio di controllo di parità di* $\mathcal{C}$*.*

Osserviamo che, in generale, $h_R(x)$ non è un polinomio monico.

2.5 Codifica dei codici ciclici

Nei paragrafi precedenti si è visto come sia possibile costruire la matrice generatrice G di un $[n,k]$–codice ciclico $\mathcal{C}$ a partire dal suo polinomio generatore $g(x)$. In particolare, si può assumere che tale matrice abbia la forma

$$G = (R\, I_k)$$

come fatto nella costruzione di Pagina 14. In effetti, vedremo ora come sia possibile codificare una sequenza di informazione $\mathbf{a}$ senza dover necessariamente scrivere espicitamente tutta la matrice.

Per ogni $i = 0, 1, \ldots, k-1$ chiamiamo $r_i(x)$ il polinomio in x tale che

$$x^{n-k+i} = q_i(x)g(x) + r_i(x).$$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Rammentiamo che data una sequenza di informazione $\mathbf{a} = (a_0, a_1, \ldots, a_k)$, possiamo sempre costruire un polinomio

$$a(x) = \sum_{i=0}^{k-1} a_i x^i.$$

Poiché la forma della matrice G è antisistematica, ogni parola di codice conterrà i k simboli rappresentanti l'informazione nelle ultime componenti, mentre i simboli di controllo di parità saranno i primi $n-k$. Pertanto, per determinare la rappresentazione di $a(x)$ basta trovare l'unico vettore di $\mathcal{C}$ contenente i termini

$$\begin{matrix} a_0 x^{n-k} \\ a_1 x^{n-k+1} \\ \vdots \\ a_k x^n \end{matrix} .$$

In altre parole, si deve scrivere un polinomio $q(x)$ tale che

$$x^{n-k} a(x) = q(x)g(x) + t(x), \tag{2.1}$$

ossia

$$q(x)g(x) = -t(x) + x^{n-k} a(x),$$

con $t(x) = 0$ oppure $\deg t(x) < \deg g(x) = n-k$.

Osserviamo che la forma di una parola di codice risulta

$$\begin{array}{llllll} [\boxed{-a_0 r_0(x)} & a_0 & 0 & \ldots & 0 &]+ \\ [\boxed{-a_1 r_1(x)} & 0 & a_1 & \ddots & 0 &]+ \\ [\quad \vdots \quad & \vdots & \ddots & \ddots & \vdots &]+ \ ; \\ [\boxed{-a_{k-1} r_{k-1}(x)} & 0 & \ldots & 0 & a_{k-1} &]= \\ \\ [\boxed{-t(x)} & a_0 & a_1 & \ldots & a_{k-1} &] \end{array}$$

In particolare, i simboli di controllo di parità, associati all'informazione rappresentata da $a(x)$, sono proprio i coefficienti di un polinomio $-t(x)$ dove $t(x) = \sum_{i=0}^{k-1} a_i r_i(x)$. Da questo fatto, per la relazione (2.1), discende che il polinomio $t(x)$ è esattamente il resto della divisione di $x^{n-k} a(x)$ per il polinomio generatore del codice $g(x)$.

Descriviamo ora formalmente un primo conciso algoritmo di codifica, per un $[n, k]$–codice ciclico binario, basato su quanto appena osservato.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Algoritmo 2.1 (Codifica codici ciclici). DATI:

D1 un $[n, k]$–codice ciclico C con polinomio generatore $g(x)$;

D2 un vettore di informazione $\mathbf{a} = (a_0, a_1, \ldots, a_{k-1})$.

DETERMINARE:

G1 un vettore $\mathbf{s} = (s_0, s_1, \ldots, s_{n-k-1})$ di simboli di controllo parità tale che $(\mathbf{a}, \mathbf{s}) \in C$

SI PROCEDA COME SEGUE:
Si ponga $\tilde{\mathbf{g}} = (g_0 g_1 \ldots g_{n-k-1})$.

S1 Porre $s_j = 0$ per $0 \leq j \leq n-k-1$.

S2 Porre $i = 1$.

S3 Distinguiamo due casi:

(a) Se $a_{k-i} = s_{n-k-1}$, porre $s_j = s_{j-1}$ per j che va da $n-k-1$ ad 1 ed $s_0 = 0$.

(b) Se $a_{k-i} \neq s_{n-k-1}$, porre $s_j = s_{j-1} + g_j$ per j che va da $n-k-1$ ad 1 ed $s_0 = g_0$.

S4 Se $i > k$ fermarsi, altrimenti tornare al passo 3.

Il grande vantaggio di questo algoritmo è che non si richiede di memorizzare tutta la matrice generatrice $G = (R\, I_k)$ e nemmeno di calcolare a priori i polinomi $r_i(x)$.

2.6 Decodifica dei codici ciclici

Anche la decodifica di un codice ciclico può essere implementata in modo particolarmente conciso in termini di operazioni fra polinomi.

Al solito, fissiamo un $[n, k]$–codice ciclico $\mathcal{C}$ sopra $\mathbb{F}$, con polinomio generatore $g(x)$. Da quanto visto nel precedente Paragrafo 2.5, è sempre possibile rappresentare la operazione di codifica mediante una matrice generatrice G della forma $G = (R|I_k)$. Conseguentemente, la matrice di controllo di parità di $\mathcal{C}$ deve avere la forma

$$H = \left(I_{n-k}| - R^T\right). \tag{2.2}$$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

In termini dei polinomi $r_i(x)$ si vede immediatamente che

$$H = \left(\begin{array}{cccc|c|c|c|c} 1 & 0 & \dots & 0 & & & & \\ 0 & 1 & \ddots & \vdots & r_0(x) & r_1(x) & \cdots & r_{k-1}(x) \\ \vdots & \ddots & \ddots & 0 & & & & \\ 0 & \dots & 0 & 1 & & & & \end{array}\right).$$

Sia adesso $h(x)$ il polinomio di controllo di parità di $\mathcal{C}$ e consideriamo ora una sequenza ricevuta $\mathbf{n}$. Poniamo $\mathbf{s} = \mathbf{n}H^T$; pertanto, il vettore $\mathbf{s}$ è la sindrome associata a $\mathbf{n}$. Rammentiamo che, per costruzione, la matrice di controllo di parità H è la matrice generatrice di un codice ciclico con polinomio generatore $h_R(x)$. In particolare, si vede che il calcolo della sindrome corrisponde, in termini polinomiali, ad un prodotto del polinomio $s(x)$ associato alla sequenza ricevuta s per il polinomio $h(x)$, il tutto calcolato nell'anello $\mathbb{F}[x]_{n-k}$. C'è una particolare convenienza nello scegliere per $\mathcal{C}$ una matrice di controllo di parità H del tipo della (2.2), come dimostra il seguente teorema.

Teorema 2.5. *Consideriamo un* $[n, k]$*–codice ciclico* $\mathcal{C}$ *sopra* $\mathbb{F}$ *generato da* $g(x)$. *Siano* $\mathbf{v} \in \mathbb{F}^n$ *e* $\mathbf{s}$ *rispettivamente un vettore e la sua sindrome e indichiamo con* $v(x)$ *e* $s(x)$ *le corrispondenti rappresentazioni polinomiali. Allora,* $s(x)$ *è il resto della divisione di* $v(x)$ *per* $g(x)$.

In altre parole, il Teorema 2.5 mostra che la sindrome di una sequenza può essere determinata semplicemente eseguendo una divisione fra polinomi mediante l'algoritmo euclideo. Questo, in generale, risulta molto più semplice da realizzare, che non il calcolare la sindrome mediante il prodotto matrice/vettore.

2.7 Codici ciclici per burst di errore

Prima di concludere osserviamo che esistono dei codici ciclici che sono contemporaneamente in grado di decodificare burst errors e errori generici. Questi sono i cosiddetti codici fortemente b–burst error correcting.

In particolare, utilizzando dei codici che soddisfino la Definizione 2.6 è possibile disegnare un decodificatore che contemporaneamente corregga alcuni errori ciclici e ne identifichi altri. Questo è il contenuto del Teorema 2.6.

Definizione 2.6. Un $[n, k]$–codice ciclico con polinomio generatore $g(x)$ è detto un codice *fortemente* b*–burst error correcting* se, dati due vettori di errore $\mathbf{e}_1$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

e $\mathbf{e_2}$ con la medesima sindrome e descrizioni rispettivamente $(\mathbf{p_1}, i_1)$ e $(\mathbf{p_2}, i_2)$ con $|\mathbf{p_1}| + |\mathbf{p_2}| \leq 2b$, si ha $\mathbf{e_1} = \mathbf{e_2}$.

Teorema 2.6. *Sia $\mathcal{C}$ un codice fortemente* b*–correttore di burst error. Allora, per ogni b_1, b_2 tali che $b_1 \leq b_2$ e $b_1 + b_2 = 2b$, è possibile implementare un decodificatore che corregga tutti i burst di errore di lunghezza al più b_1 e,* contemporaneamente, *identifichi tutti i burst di errore di lunghezza al più b_2.*

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Capitolo 3

Codici di Reed–Solomon

LA FAMIGLIA di codici piú utilizzata nelle applicazioni pratiche è sicuramente quella di Reed–Solomon, [73].

3.1 Costruzione di Reed–Solomon

I codici di Reed–Solomon sono un tipo particolare di codice ciclico; un metodo per introdurli è quello di utilizzare la cosiddetta costruzione BCH (da introdotta nel 1960 da R. C. Bose e D. Ray–Chaudhuri e, indipendentemente, nel 1959 da A. Hocquenghem), scegliendo opportunamente i parametri. Tale costruzione è descritta nei due seguenti paragrafi.

Un metodo più diretto per ottenere tale codici è quello presentato nella seguente definizione.

Definizione 3.1. Sia $n = p^t - 1$, ove p è un primo e t un intero maggiore di 0. Per ogni $k \leq n$, Si dice *codice di Reed–Solomon* $\mathrm{RS}(n, k)$ lo spazio vettoriale tutte le funzioni polinomiali $f : \mathbb{F}_{n+1}^{\star} \mapsto \mathbb{F}_{n+1}$ aventi grado al più k.

In particolare, un modo per scrivere tutte le parole del codice di Reed–Solomon $\mathrm{RS}(n, k)$ è il seguente:

$$\mathcal{C} = \{(c_0, c_1, \ldots, c_{n-1}) : c_i = a(\omega^i); a(x) \in \mathbb{F}_q[x], \deg a(x) < k\},$$

ove ω è un elemento primitivo[1] di $\mathbb{F}_{n+1}$.

[1] Questo significa che per ogni elemento $\alpha \in \mathbb{F}_{n+1}$ diverso da 0 esiste un indice i tale che $\alpha = \omega^i$. Elementi di questo tipo possono sempre essere trovati in $\mathbb{F}_{n+1}$. Osserviamo infine che per ogni elemento $\alpha \in \mathbb{F}_{n+1}$ diverso da 0 vale sempre la relazione $\alpha^n = 1$.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Calcoliamo ora la distanza minima del codice $\mathrm{RS}(n,k)$. Rammentiamo che un codice è detto MDS (*Maximum distance separable*) se i suoi parametri soddisfano con l'uguaglianza la limitazione di Singleton

$$d-1 \leq n-k.$$

Teorema 3.1. *La distanza minima del codice* $\mathrm{RS}(n,k)$ *è esattamente* $n-k+1$*; pertanto questo codice è MDS.*

Dimostrazione. Siano $p(x)$, $q(x)$ due polinomi distinti di grado al più $k-1$. Poniamo $r(x) = p(x) - q(x)$. Chiaramente,

$$p(x) = q(x) \Leftrightarrow r(x) = 0,$$

$\deg r(x) < k$ e $r(x)$ non è il polinomio nullo. Sia I l'insieme degli indici tali che $p(\omega^i) = q(\omega^i)$. Per il teorema di Ruffini, possiamo scrivere

$$r(x) = \prod_{i \in I} (x - \omega^i).$$

In particolare, il grado di $r(x)$ è uguale alla cardinalità di I. Ne segue $|I| \leq k$ e dunque la distanza di Hamming fra le parole $\mathbf{p}$ e $\mathbf{q}$ associate a $p(x)$ e $q(x)$ deve soddisfare

$$d(\mathbf{p},\mathbf{q}) \geq n - (k-1) = n-k+1.$$

Dq questo discende la tesi. □

3.2 Codifica dei codici di Reed–Solomon

Un metodo particolarmente efficiente per codificare una messaggio

$$\mathbf{m} = (m_0\, m_1 \cdots m_k)$$

mediante un $[n,k]$–codice di Reed–Solomon $\mathcal{C} = \mathrm{RS}_d(n)$ è il seguente. Sia ω un elemento primitivo di $\mathbb{F}_{n+1}$. Si può rappresentare $\mathbf{m}$ mediante un polinomio

$$m(x) = \sum_{i=0}^{n-d-1} m_i x^i.$$

A questo punto associamo a $m(x)$ il vettore

$$\mathbf{c} = (m(\omega^0)\, m(\omega^1) \cdots m(\omega^{n-1})).$$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Per quanto visto prima, $\mathbf{c} \in \mathcal{C}$, per cui l'operazione descritta $\mathbb{F}_{n+1}^k \mapsto \mathbb{F}_{n+1}^n$ è una codifica. Affinché questa procedura di codifica sia efficiente è necessario prestare particolare attenzione all'implementazione dell'operazione di prodotto del campo finito $\mathbb{F}_{n+1}$. Osserviamo che la codifica qui presentata non è di tipo sistematico e, in generale, i valori m_i non compaiono in alcuna posizione.

3.3 Decodifica e DFT

Innanzi tutto mostriamo come sia possibile ricostruire a partire da un vettore

$$\mathbf{c} = (m(\omega^0)\, m(\omega^1) \cdots m(\omega^{n-1}))$$

i coefficienti del polinomio $m(x)$.

A tal fine introduciamo la nozione di trasformata discreta di Fourier.

Definizione 3.2. Sia ω un elemento primitivo fissato del campo finito $\mathbb{F}_{n+1}$; indichiamo con V lo spazio vettoriale $\mathbb{F}_{n+1}^n$ di dimensione n su $\mathbb{F}_{n+1}$. Consideriamo un vettore generico

$$\mathbf{f} = (f_0\, f_1 \cdots f_{n-1}) \in V.$$

La *trasformata discreta di Fourier* (DFT) di $\mathbf{c}$ è il vettore

$$\hat{\mathbf{f}} = (\hat{c}_0\, \hat{c}_1 \cdots \hat{c}_{n-1}) \in V$$

la cui entrata t–esima è

$$\hat{f}_t = \sum_{i=0}^{n-1} c_i \omega^{it}.$$

Osserviamo che se scriviamo

$$f(x) = \sum_{i=0}^{n-1} f_i x^i,$$

allora la componente t–esima della DFT di $\mathbf{f}$ è esattamente il valore $f(\omega^t)$. Pertanto si può dire che la operazione di codifica di un vettore mediante un codice di Reed–Solomon corrisponde al calcolarne la trasformata.

Mostriamo ora che la DFT è invertibile. A tal fine è necessario premettere un lemma.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Lemma 3.2. *Sia $\alpha \in \mathbb{F}_{n+1}$ un elemento di un campo finito. Allora,*

$$\sum_{i=0}^{n-1} \alpha^i = \begin{cases} n & \text{se } \alpha = 1 \\ 0 & \text{altrimenti} \end{cases}.$$

Dimostrazione. Consideriamo l'espressione

$$\alpha \sum_{i=0}^{n-1} \alpha^i = \sum_{i=0}^{n-1} \alpha^{i+1} = \sum_{i=1}^{n} \alpha^i.$$

D'altro canto, $\alpha^n = 1 = \alpha^0$ per ogni $\alpha \in \mathbb{F}_{n+1}$, per cui

$$(\alpha - 1)\left(\sum_{i=0}^{n-1} \alpha^i\right) = 0.$$

Ne segue che $\alpha = 1$ oppure la sommatoria è nulla. □

Scriviamo la componente t–esima della trasformata

$$\hat{\hat{f}}_t = \sum_{i=0}^{n-1} \hat{f}_i \omega^{it} = \sum_{i=0}^{n-1} \left(\sum_{j=0}^{n-1} f_j \omega^{ij}\right) \omega^{it} = \sum_{j=0}^{n-1} f_j \left(\sum_{i=0}^{n-1} \omega^{i(j+t)}\right).$$

Per il Lemma 3.2, la somma fra parentesi è nulla tranne quando $j + t = n$, nel quale caso, essa vale n. Pertanto,

$$\hat{\hat{f}}_t = n f_{-t},$$

ove l'indice t è da intendersi ridotto modulo n. ed è dunque sempre possibile ricavare i valori di $\mathbf{f}$ a partire da quelli di $\hat{f}$. In termini di polinomi questa relazione si può scrivere come

$$f_t = \frac{1}{n} \hat{\hat{f}}(\omega^{-t}).$$

3.4 Il problema dalla correzione

Il problema chiave per l'algoritmo di correzione è il seguente.

Problema 3.1 (Interpolazione di polinomi)**.** Siano

a. q una potenza di primo;

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

b. $t \leq n/2$ un intero.

Poniamo $n = q - 1$ e $k = n - 2t$.
Dati

D1. n elementi distinti $x_0, x_1, x_2, \ldots, x_{n-1}$ di $\mathbb{F}_q$ e

D2. n elementi (non necessariamente distinti) $y_0, y_1, \ldots, y_{n-1} \in \mathbb{F}_q$

determinare un polinomio $P(x) \in \mathbb{F}_q[x]$ con:

a. $\deg P(x) \leq k - 1$;

b. $P(x_i) \neq y_i$ per al più t valori di i.

Nella formulazione generale questo problema è molto difficile da affrontare dal punto di vista computazionale[2].

Fortunatamente, la decodifica di un codice di Reed–Solomon corrisponde ad una istanza molto particolare del problema sopra presentato ed è fattibile in tempo polinomiale. Il seguente schema mostra come si possa procedere in questo caso.

1. Innanzi tutto verifichiamo che risolvere il Problema 3.1 è equivalente a decodificare una parola ricevuta con Reed–Solomon. Sia ora $\mathcal{C}$ un il codice $\mathrm{RS}_d(n)$. Poniamo $x_i = \alpha^i$ e supponiamo che sia ricevuta una parola

$$\mathbf{r} = (y_0\, y_1 \ldots y_{n-1}),$$

associata ad una parola di codice originariamente trasmessa $\mathbf{P}$. Notiamo che $\mathbf{P}$ è la valutazione su tutti gli elementi non nulli di $\mathbb{F}_q$ di un qualche polinomio $P(x) \in \mathbb{F}_q[x]$ con $\deg P(x) \leq k - 1$. La presenza di errori correggibili, ovvero con peso non superiore a $t = \lfloor (d-1)/2 \rfloor$, implica che $\mathbf{r}$ differisce da $\mathbf{P}$ in al più t posizioni. Le condizioni del Problema 3.1, e, in particolare la (b), sono dunque soddisfatte e quindi una soluzione di quel problema consente di determinare il polinomio $P(x)$ e, conseguentemente, la parola $\mathbf{P}$.

[2] Esso sicuramente appartiene alla classe NP (si veda [36] per una trattazione approfondita della teoria della complessità), in quanto è possibile verificare in tempo polinomiale se una soluzione $P(x)$ è corretta, ma non sono correntemente noti algoritmi in tempo polinomiale per determinare tale $P(x)$. Inoltre, non è nemmeno noto se sia possibile dimostrare che una soluzione non possa essere calcolata in tempo polinomiale (questa ultima condizione corrisponderebbe a mostrare che il Problema 3.1 appartiene alla classe co–NP)

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

2. Proviamo ora che, se esiste una soluzione a tale problema, allora essa è, sotto le condizioni correnti, unica e coincide esattamente con $P(x)$. La tecnica dimostrativa è come segue. Supponiamo dunque che esistano due polinomi $Q(x)$, $R(x)$ che rispondano al Problema 3.1 e soddisfino le condizioni di cui sopra. Allora il polinomio

$$S(x) = Q(x) - R(x)$$

è diverso da 0 per al più $2t$ valori di x. D'altro canto, l'ordine del campo $\mathbb{F}_q$ è $n+1$, per cui $S(x)$ ha necessariamente almeno $n+1-2t > k$ radici. Ne segue $S(x) = 0$ identicamente oppure $\deg S(x) > k$, una contraddizione, in quanto i gradi di $Q(x)$ e $R(x)$ sono al più $k-1$.

3. Dalle due considerazioni precedenti discende che, se il Problema 3.1 ammette soluzione, allora tale soluzione fornisce l'unica decodifica della parola $\mathbf{r}$ secondo il codice $\mathcal{C}$.

3.5 Algoritmo di Welch–Berlekamp

Presentiamo ora un algoritmo concreto di complessità polinomiale $O(n^3)$ per decodificare i codici di Reed–Solomon. Tale algoritmo è dovuto a Welch e Berlekamp ; esso è stato successivamente migliorato e attualmente sono note versioni della procedura di decodifica di complessità $O(n\text{poly}\log n)$.

Supponiamo che $p(x)$ sia il polinomio *corretto* associato ad una parola ricevuta $\mathbf{y} = (y_0\, y_1 \cdots y_{n-1})$. Questo polinomio è quello che l'algoritmo di correzione deve ricostruire.

Consideriamo ora un polinomio, $E(x)$, che goda della proprietà che $E(\omega^i) = 0$ se $p(\omega^i) \neq y_i$.

In generale, gli elementi ω^i corrispondenti alle posizioni di errore sono tutti radici del polinomio $E(x)$, ma ma non è detto che tutte le radici di $E(x)$ siano posizioni di errore.

Introduciamo ora un altro polinomio, $N(x)$ che dovrà rappresentare

$$N(x) = p(x)E(x).$$

In particolare vogliamo che

$$N(\omega^i) = p(\omega^i)E(\omega^i) = y_i E(\omega^i).$$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

A priori $N(x)$ è difficile da determinare, così come $E(x)$. Il seguente algoritmo mostra che la cosa è fattibile. In tale algoritmo t è il numero massimo di errori che vogliamo poter correggere; $p(x)$ il polinomio corrispondente alla parola corretta e ω^i potenze distinte di un elemento primitivo.

Algoritmo 3.1 (Welch–Berlekamp). DATI:

D1 n, k interi;

D2 $t \leq \frac{n-k}{2}$ intero;

D3 $\omega^0, \omega^1, \ldots, \omega^{n-1} \in \mathbb{F}_{n+1}$ elementi tutti fra loro distinti;

D4 $y_0, y_1, \ldots, y_{n-1} \in \mathbb{F}$.

DETERMINARE:

G1 Un polinomio $p(x)$ tale che

a. $\deg p(x) < k$;

b. $p(x_i) = y_i$ per ogni i compreso fra 0 e $n-1$ tranne che in al più t casi.

SI PROCEDA COME SEGUE:

S1 Cerchiamo due polinomi $N(x) = \sum_{j=0}^{k+t-1} N_j x^j$ ed $E(x) = \sum_{j=0}^{t} E_j x^j$ tali che

(a) $\deg E(x) = t$;

(b) $E(x)$ è monico, cioè

$$E_t = 1 \tag{3.1}$$

(c) $\deg N(x) \leq k + t - 1$;

(d) per ogni $i = 0, \ldots, n-1$,

$$N(\omega^i) = y_i E(\omega^i). \tag{3.2}$$

S2 Le relazioni (3.1) e (3.2) corrispondono ad un sistema di $n+1$ equazioni nelle $k+t-1+t+1 = k+2t \leq n$ incognite $N_0, N_1, \ldots, N_{k+t-1}$, $E_0, E_1, \ldots E_t$. Tale sistema può essere risolto in $O(n^3)$ con tecniche standard e fornisce i due polinomi $N(x)$ e $E(x)$.

S3 A questo punto è possibile ricavare $p(x)$ come $p(x) = N(x)/E(x)$.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

a. Il fatto che il grado di $E(x)$ sia t è *esplicitamente* richiesto perché l'algoritmo possa funzionare.

b. In generale la coppia $(N(x), E(x))$ calcolata non è unica. D'altro canto, se $(N(x), E(x))$ e $(N'(x), E'(x))$ sono due polinomi che soddisfano le condizioni di cui sopra, allora

$$y_i E(\omega^i) = N(\omega^i) \text{ ed } N'(\omega^i) = y_i E'(\omega^i),$$

da cui, moltiplicando le relazioni fra loro,

$$y_i E(\omega^i) N'(\omega^i) = y_i E'(\omega^i) N(\omega^i). \tag{3.3}$$

Dalla relazione (3.3) segue che

$$E(\omega^i) N'(\omega^i) = E'(\omega^i) N(\omega^i) \tag{3.4}$$

per ogni i. Infatti, per $y_i \neq 0$ l'affermazione è banale, dividendo per y_i. Se invece $y_i = 0$, allora $N(\omega^i) = N'(\omega^i) = 0$ e dunque abbiamo nuovamente che la relazione (3.4) è soddisfatta. Ora, poiché

$$\deg E'(x)N(x) = \deg E(x)N'(x) \leq k + 2t - 1 < n,$$

abbiamo che i due polinomi $E(x)N'(x)$ e $E'(x)N(x)$ coincidono in un numero di punti superiore al loro grado e sono dunque identici. Ne segue che $p(x) = N(x)/E(x)$ è univocamente determinato.

c. È immediato verificare che $\deg p(x) < k$; d'altro canto, se ω^i non è una radice di $E(x)$, allora

$$p(\omega^i) = N(\omega^i)/E(\omega^i) = y_i E(\omega^i)/E(\omega^i) = y_i,$$

per cui il polinomio $p(x)$ soddisfa le proprietà richieste ed è soluzione al problema della decodifica del codice di Reed–Solomon.

Per concludere, osserviamo che l'algoritmo presentato può essere implementato in tempo polinomiale ($O(n^3)$). Infatti, la risoluzione di un sistema lineare in $(k + 2t) \leq n$ incognite è sempre possibile in un tempo che è $O(n^3)$.

I fatti essenziali che sono stati sfruttati per la decodifica dei codici di Reed–Solomon sono stati i seguenti:

a. gli indici delle parole sono rappresentati da elementi $x_i = \alpha^i$ di $\mathbb{F}_q$;

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

b. le valutazioni di due polinomi che corrispondono a parole di codice distinte non possono coincidere in molti punti;

c. il prodotto di due polinomi di grado basso ha a sua volta grado basso;

d. il quoziente di due polinomi risulta (in casi opportuni), a sua volta un polinomio.

3.6 Applicazioni dei codici di Reed–Solomon

In questo paragrafo presenteremo due esempi di applicazione dei codici di Reed–Solomon, il primo è il sistema comunicazione delle sonde Voyager, il secondo il formato di archiviazione dei dati su di un DVD.

Esempio 3.1. Le sonde Voyager sono state lanciate nel 1977. L'obiettivo originario di fornire informazioni e immagini ad alta risoluzione di Giove e Saturno e, eventualmente dei pianeti più esterni. Il progetto del sistema prevede due canali di comunicazione, su bande di frequenza distinte, denominate con le lettere `S` (2115 MHz) e `X` (8415 MHz). In particolare, il canale sulla banda `X` è dedicato alla trasmissione dei dati dalla sonde verso terra, mentre il canale sulla banda `S` è per la trasmissione da terra verso le sonde e, eventualmente, come canale secondario di trasmissione dati dalla sonda. La capacità in bit al secondo del canale `S` è compresa fra 10 e 2560, mentre per sul canale `X` è possibile trasmettere sino a 115.2 kilobits al secondo; si veda [56] per i dettegli.

La distanza della sonda Voyager 2 dalla terra al momento dell'incontro con Giove era di circa 5.2 AU; al momento dell'incontro con Nettuno di 30 AU, ove un unità astronomica (AU) corrisponde a circa 1.496×10^8 chilometri. È naturale prevedere che su queste distanze si possano verificare errori nella propagazione del segnale, in particolare qualora si vogliano trasmettere dati ad alta velocià sulla banda `X`. Per poter correggere questi errori, i progettisti delle missioni hanno provveduto ad implementare dei codici correttori di errore. Tali codici sono costruiti combinando un codice lineare con un codice convoluzionale opportuno.

Il codice lineare utilizzato per trasmettere le immagini da Giove e Saturno era il $[24, 12, 8]$–codice binario di Golay esteso[3] $\mathcal{G}_{24}$. Questo codice è in grado di correggere al più 3 errori per ogni 24 bits di dati trasmessi (esattamente 1 errore per byte), ma ha una ridondanza di 12 bits, pari al 100% della lunghezza del messaggio originale.

[3] Il codice lineare binario di tali parametri è unico

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Per trasmettere le informazioni dai pianeti più esterni (Urano e Nettuno) era necessario utilizzare una codifica più efficiente. A tal fine si è sostituito il codice $\mathcal{G}_{24}$ con il codice di Reed–Solomon $\mathrm{RS}(255, 223)$ di distanza minima 33. Tale codice:

a. ha dimensione 223 su $\mathbb{F}_{2^8}$;

b. garantisce di correggere almeno 16 errori per ogni blocco di 255 bits, pari a 1 errore per ogni blocco di 16 bits;

c. ha una ridondanza di soli 32 bits pari al 14% della lunghezza del messaggio originario.

È chiaro che il codice di Reed–Solomon, da solo, può correggere meno errori che non quello di Golay, ma il risparmio in termini di informazione trasmessa rende l'operazione comunque conveniente. In realtà, la modifica dell'apparecchiatura di codifica nonchè l'impiego di algoritmi più sofisticati di decodifica ha consentito di abbassare la frazione di errori non corretti da 5×10^{-3} sino a 10^{-6}.

Esempio 3.2. I DVD–RO (*Read–Only digital versatile disk*) sono un supporto di memorizzazione dati ad alta capacità (possono contenere sino a 8.5 Gigabyte di informazioni. Dal punto di vista fisico si tratta di dischi del diametro di 120 millimetri con un foro centrale di 15 millimetri. Ogni disco è formato da due strati di supporto incollati fra loro. Questi strati isolano una o due lamine su cui i dati sono immagazzinati come variazioni dell'indice di diffrazione delle lamine stesse.

Un supporto di memorizzazione quali i DVD può essere soggetto a danni dovuti sia all'invecchiamento del supporto che all'usura. In particolare, vi può essere un alterazione della lamina su cui sono immagazzinati i dati a causa di condizioni avverse (ad esempio, esposizione al calore o a luce solare intensa) oppure gli strati di supporto possono essere danneggiati da graffi e/o abrasioni. Al fine di incrementare la durata utile dei dischi, i dati devono essere immagazzinati utilizzando un codice a correzione di errore. Il procedimento è come segue. Ogni blocco di 192×172 bytes viene rappresentato mediante una matrice rettangolare B. Dapprima si applica ad ogni riga della matrice il codice $\mathrm{RS}(182, 172)$ di parametri $[182, 172, 11]$. In tal modo si ottiene una nuova matrice B' di dimensioni 192×182. A questo punto, ad ogni colonna di B' si applica il codice $\mathrm{RS}(208, 192)$ di parametri $[208, 192, 17]$, di modo da ottenere una matrice finale B'' di dimensioni 208×182. Questo è il blocco di informazione che viene concretamente scritto su disco.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

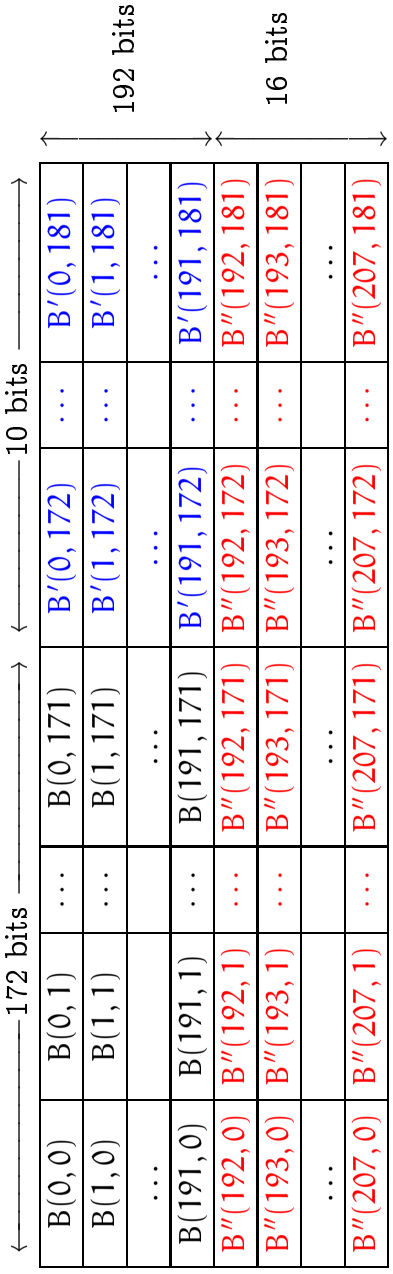

Figura 3.1: Struttura del codice usato per i DVD

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Questa costruzione è un esempio di codifica prodotto. In particolare la distanza minima del codice che associa a B il blocco B′ nel modo sopra descritto è il prodotto $11 \times 17 = 187$. Pertanto, si ottiene un $[37856, 33024, 187]$–codice lineare $\mathcal{C}$. Questo codice ha una ridondanza di 4832 bits, pari al 6% della lunghezza del messaggio da codificare ed è sicuramente in grado di correggere almeno 93 errori, pari ad un errore ogni 407 bytes.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Capitolo 4

Erasures

4.1 Generalità

Il metodo più intuitivo per fornire una descrizione di un generico vettore $\mathbf{e} = (e_0\, e_1 \cdots e_{n-1})$ su $\mathbb{F}_q$ è quello di fornire l'elenco dei valori di tutte le sue componenti. Questo è equivalente a fornire un insieme E contenente n coppie ordinate (i, e_i), ove i è un intero compreso fra 0 e $n-1$, mentre $e_i \in \mathbb{F}_q$. Il metodo sopra presentato si presta ad una possibilità ottimizzazione: invece che considerare l'insieme E formato da tutte le coppie ordinate, possiamo scrivere l'insieme

$$\widehat{E} = \{(i, e_i) : e_i \neq 0\}.$$

È chiaro che è possibile ricostruire il vettore originario $\mathbf{e}$ anche a partire da questa ultima rappresentazione. Gli elementi i tali che esista un e_i con $(i, e_i) \in \widehat{E}$ sono dette *posizioni*, mentre i corrispondenti elementi di $\mathbb{F}_q$ prendono il nome di *ampiezze*.

Nel caso di una trasmissione di un messaggio su di un canale $\mathbb{F}_q$–ario è necessario, per poter effettuare una correzione di errore, riuscire a determinare integralmente l'eventuale vettore di errore $\mathbf{e}$, ovvero sia le posizioni che le ampiezze d'errore sono incognite e devono venire calcolate.

In questo capitolo ci porremo in un contesto diverso: in particolare investigheremo cosa accade quando si vuole determinare un vettore di errore $\mathbf{e}$ la posizione delle cui entrate non nulle è, almeno parzialmente, nota a priori.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

4.2 Il canale q–ario con erasure

Il il *canale q–ario con erasure* è una naturale generalizzazione del canale q–ario. In questo caso si suppone che l'apparato ricevente, invece che fornire semplicemente al decodificatore caratteri dell'alfabeto $\mathbb{F}_q$ su cui il codice è definito, possa anche generare un simbolo aggiuntivo, diciamo ?, corrispondente a valori ricevuti ritenuti a priori non intellegibili.

Premettiamo una generalizzazione della nozione di distanza di Hamming da utilizzare in questo caso.

Definizione 4.1. Sia $\mathbb{F}$ un alfabeto. Denotiamo con $\overline{\mathbb{F}}$ l'insieme $\mathbb{F} \cup \{?\}$. La *distanza di Hamming estesa* $\overline{d}(x, y)$ in $\overline{\mathbb{F}}$ è la seguente funzione $\overline{\mathbb{F}} \times \overline{\mathbb{F}} \mapsto \mathbb{Q}$

$$\overline{d}(x, y) = \begin{cases} 0 & \text{se } x = y \\ 1 & \text{se } x \neq y \text{ e inoltre } x, y \neq ? \\ \frac{1}{2} & \text{se } x \neq y \text{ e } x = ? \text{ oppure } y = ?. \end{cases}$$

La distanza di Hamming estesa si può applicare ad ogni coppia di vettori di lunghezza n ad entrate in $\overline{\mathbb{F}}$, ponendo

$$\overline{d}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \overline{d}(x_i, y_i).$$

Quando si suppone che il simbolo ? (che denota una erasure) non appaia, la distanza $\overline{d}$ coincide semplicemente la distanza di Hamming ordinaria. Osserviamo che in generale la parola ricevuta $\mathbf{r}$ è definita sull'alfabeto $\overline{\mathbb{F}}$, ma sia l'errore che si vuole determinare $\mathbf{e}$ che la eventuale parola decodificata $\mathbf{c}$ sono vettori su $\mathbb{F}$.

Intuitivamente, una *erasure*[1] è un errore di cui è nota la posizione ma non la grandezza.

La seguente è una descrizione formale della situazione.

Definizione 4.2. Sia $\mathbf{r} = (r_0 r_1 \cdots r_{n-1})$, una parola definita sull'alfabeto $\overline{\mathbb{F}}$. Si dice *erasure* in $\mathbf{r}$ ogni indice $0 \leq i \leq n-1$ tale che $r_i = ?$.

Per comodità di notazione, definiamo per ogni $x \in \overline{\mathbb{F}}$,

$$x + ? = x.$$

[1] *erasure*

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

4.3 Decodifica di erasures

Supponiamo ora di avere un $[n, k]$–codice $\mathcal{C}$ definito su $\mathbb{F}$. L'algoritmo generico di decodifica a distanza minima si applica direttamente al caso di canale con erasure. L'unica accortezza è che ad ogni iterazione si deve incrementare la distanza di $\frac{1}{2}$ invece che di 1.

Il seguente teorema consente di formalizzare quale è l'effettivo potere di correzione di un codice lineare su di un canale q–ario con erasures.

Teorema 4.1. *Sia $\mathcal{C}$ un $[n, M]$-codice sopra un alfabeto $\mathbb{F}$ con distanza minima d. Allora $\mathcal{C}$ è in grado di correggere ogni forma di errore composta da e_0 erasures e e_1 errori ove*

$$e_0 + 2e_1 \leq d - 1.$$

Per un $[n, k]$–codice lineare sopra $\mathbb{F}_q$, le erasures possono essere determinate risolvendo un sistema di equazioni lineari, come illustrato nel seguente esempio.

Esempio 4.1. Consideriamo il codice $\mathcal{C}$ su $\mathbb{F}_{16}$ costruito come segue. Sia $\alpha \in \mathbb{F}_{16}$ un elemento primitivo con $1 + \alpha + \alpha^4 = 0$ e poniamo $\beta = \alpha^3$. In particolare, β è una radice quinta primitiva dell'unità. Allora, il polinomio

$$g(x) = (x - \beta)(x - \beta^2)(x - \beta^3)$$

genera un codice BCH $_{16}(5, 4)$ di dimensione 2 e distanza designata 4. Si verifica direttamente che la distanza minima di tale codice è effettivamente 4. Osserviamo che la matrice generatrice G e la matrice di controllo di parità H di $\mathcal{C}$ sono

$$G = \begin{pmatrix} \alpha^3 & \alpha^2 & \alpha^{11} & 1 & 0 \\ 0 & \alpha^3 & \alpha^2 & \alpha^{11} & 1 \end{pmatrix}, \qquad H = \begin{pmatrix} 1 & \alpha^{11} & \alpha^{12} & 0 & 0 \\ 0 & 1 & \alpha^{11} & \alpha^{12} & 0 \\ 0 & 0 & 1 & \alpha^{11} & \alpha^{12} \end{pmatrix}.$$

Sia

$$\mathbf{r} = (?\, \alpha^6\, ?\, ?\, 1),$$

in cui "?" indica componenti per cui si può asserire che si è verificata una erasure, un vettore ricevuto e supponiamo che si possa garantire che non ci sono stati altri errori. A norma del Teorema 4.1, è possibile trovare una unica parola in $\mathbf{c} = \mathcal{C}$ a distanza minima da $\mathbf{r}$.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Osserviamo che, poiché non si sono verificati errori che non siano erasures possiamo sicuramente scrivere

$$\mathbf{c} = (c_0\, c_1\, c_2\, c_3\, c_4) = \mathbf{r} + (e_0\, 0\, e_2\, e_3\, 0). \tag{4.1}$$

Dato che si deve avere $\mathbf{c}H^T = 0$, otteniamo $\mathbf{c}H^T = (e_0 r_1 e_2 e_3 r_4)H^T$, ossia, un sistema di tre equazioni lineari nelle incognite e_0, e_2, e_3

$$\begin{cases} \alpha^2 = (\alpha^3 + \alpha^2 + \alpha + 1)e_2 + e_0 \\ \alpha^3 + \alpha^2 = (\alpha^3 + \alpha^2 + \alpha + 1)e_3 + (\alpha^3 + \alpha^2 + \alpha)e_2 \\ \alpha^3 + \alpha^2 + \alpha + 1 = (\alpha^3 + \alpha^2 + \alpha)e_3 + e_2 = 0, \end{cases}$$

altrimenti scritto come

$$\begin{cases} \alpha^2 &= \alpha^{12}e_2 + e_0 \\ \alpha^6 &= \alpha^{12}e_3 + \alpha^{11}e_2 \\ \alpha^{12} &= \alpha^{11}e_3 + e_2 \end{cases}.$$

Tale sistema ammette, come unica soluzione, $e_0 = \alpha^3$, $e_2 = \alpha^9$, $e_3 = \alpha^{12}$. Ne consegue

$$\mathbf{c} = (\alpha^3\, \alpha^6\, \alpha^9\, \alpha^{12}\, 1).$$

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

Bibliografia

[1] O. Amrani and Y. Be'ery. Efficient bounded–distance decoding of the hexacode and associated decoders for the leech lattice and the golay code. *IEEE Trans. Comm.*, 44(5):612–629, 1996.

[2] I. Anderson and I. Honkala. A short course in combinatorial designs. University of Turku (Finland), 1997. (http://www.utu.fi/~honkala/designs.ps).

[3] E. Artin. *Geometric algebra*. Wiley Classics Library. John Wiley & Sons Inc., New York, 1988. Reprint of the 1957 original, A Wiley-Interscience Publication.

[4] Emil Artin. *Galois theory*. Dover Publications Inc., Mineola, NY, second edition, 1998. Edited and with a supplemental chapter by Arthur N. Milgram.

[5] Robert B. Ash. *Information theory*. Dover Publications Inc., New York, 1990. Corrected reprint of the 1965 original.

[6] E. F. Assmus, Jr. On the Reed-Muller codes. *Discrete Math.*, 106/107:25–33, 1992. A collection of contributions in honour of Jack van Lint.

[7] E. F. Assmus, Jr. The category of linear codes. *IEEE Trans. Inform. Theory*, 44(2):612–629, 1998.

[8] E. F. Assmus, Jr. and J. D. Key. *Designs and their codes*, volume 103 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 1992.

[9] E. F. Assmus, Jr. and J. D. Key. Hadamard matrices and their designs: a coding-theoretic approach. *Trans. Amer. Math. Soc.*, 330(1):269–293, 1992.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

[10] E. F. Assmus, Jr. and J. D. Key. Polynomial codes and finite geometries, 1996.

[11] Alexander Barg. Extremal problems of coding theory. In *Coding theory and cryptology (Singapore, 2001)*, volume 1 of *Lect. Notes Ser. Inst. Math. Sci. Natl. Univ. Singap.*, pages 1–48. World Sci. Publishing, River Edge, NJ, 2002.

[12] Lynn Margaret Batten. *Combinatorics of finite geometries*. Cambridge University Press, Cambridge, second edition, 1997.

[13] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Information Theory*, IT-24(3):384–386, 1978.

[14] Thomas Beth, Dieter Jungnickel, and Hanfried Lenz. *Design theory. Vol. I*, volume 69 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 1999.

[15] Albrecht Beutelspacher and Ute Rosenbaum. *Projective geometry: from foundations to applications*. Cambridge University Press, Cambridge, 1998.

[16] J. Bierbrauer. Introduction to codes and their use, February 1999. (http://www.math.mtu.edu/~jbierbra/HOMEZEUGS/Codecourse.ps).

[17] A. R. Calderbank. The art of signaling: fifty years of coding theory. *IEEE Trans. Inform. Theory*, 44(6):2561–2595, 1998. Information theory: 1948–1998.

[18] P.J. Cameron. Polynomial aspects of codes, matroids and permutation groups. University of London, March 2002. (http://www.maths.qmw.ac.uk/~pjc/csgnotes/cmpgpoly.pdf).

[19] R. Chapman. Constructions of the goolay codes: A survey. University of Exeter (UK), 1997.

[20] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

[21] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley and sons, New York, 1991.

[22] Reinhard Diestel. *Graph Theory*. Springer-Verlag, New York, second edition, 2000. Graduate Texts in Mathematics, Vol. 173.

[23] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, January 2003. Preliminary version in FOCS 1999.

[24] I.M. Duursma. *Decoding codes from curves and cyclic codes*. PhD thesis, 1993. (http://www.math.uiuc.edu/~duursma/pub/).

[25] ECMA. *Data interchange on read–only* 120 *mm optical data disks (CDROM)*, Giugno 1996. ECMA–130.

[26] ECMA. 120 *mm DVD Rewritable Disk (DVD–RAM)*, Giugno 1999. ECMA–272.

[27] ECMA. 120 *mm DVD — Read–only disk*, Aprile 2001. ECMA–267.

[28] P. Elias. Error–free coding. Technical Report 285, Massachusetts Institute of Technology (Boston), 1954. (http://hdl.handle.net/1721.1/4795).

[29] P. Elias. List decoding for noisy channels. Technical Report 335, Massachusetts Institute of Technology (Boston), 1957. (http://hdl.handle.net/1721.1/4484).

[30] M. A. Epstein. Algebraic decoding for a binary erasure channel. Technical Report 340, Massachusetts Institute of Technology (Boston), 1958. (http://hdl.handle.net/1721.1/4480).

[31] G. D. Forney. *Concatenated codes*. PhD thesis, M.I.T. Dept. of Electrical Engineering, 1965. (http://hdl.handle.net/1721.1/13449).

[32] W. Fulton. *Algebraic curves. An introduction to algebraic geometry*. W. A. Benjamin, Inc., New York-Amsterdam, 1969.

[33] Robert G. Gallager. Low–density parity–check codes. MIT Press, 1963.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

[34] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4*, 2004. (http://www.gap-system.org).

[35] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for computer algebra*. Kluwer Academic Publishers, Boston, MA, 1992.

[36] Oded Goldreich. Introduction to complexity theory – lecture notes. The Weizmann Institute of Science, Israel, 1999. (http://www.wisdom.weizmann.ac.il/~oded/cc99.html).

[37] V. D. Goppa. Codes on algebraic curves. *Dokl. Akad. Nauk SSSR*, 259(6):1289–1290, 1981.

[38] V. D. Goppa. *Geometry and codes*, volume 24 of *Mathematics and its Applications (Soviet Series)*. Kluwer Academic Publishers Group, Dordrecht, 1988. Translated from the Russian by N. G. Shartse.

[39] R.L. Graham, D.E. Knuth, and Patashnik O. *Concrete Mathematics*. Addison Wesley, 1988.

[40] R. D. Gray and L. D. Davisson. *An Introduction to Statistical Signal Processing*. Cambridge University Press, Cambridge, 2004. (http://www-ee.stanford.edu/~gray/sp.html).

[41] Robert M. Gray. *Entropy and information theory*. Springer-Verlag, New York, 1990. (http://www-ee.stanford.edu/~gray/it.html).

[42] G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.0. A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern, 2005. http://www.singular.uni-kl.de.

[43] V. Guruswami and M. Sudan. Extensions to the johnson bound. Manuscript, February 2001.

[44] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *IEEE Symposium on Foundations of Computer Science*, pages 28–39, 1998.

[45] R. W. Hamming. Error detecting and error correcting codes. *Bell System Tech. J.*, 26:147–160, 1950.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

[46] J. W. P. Hirschfeld. *Projective geometries over finite fields*. Oxford Mathematical Monographs. The Clarendon Press Oxford University Press, New York, second edition, 1998.

[47] T. Høholdt and R. Pellikaan. On the decoding of algebraic–geometric codes. *IEEE Trans. Inform. Theory*, IT–41:1589–1614, 1995.

[48] Daniel R. Hughes and Fred C. Piper. *Projective planes*. Springer-Verlag, New York, 1973. Graduate Texts in Mathematics, Vol. 6.

[49] Dieter Jungnickel and Bernhard Schmidt. Difference sets: An update.

[50] D.E. Knuth. *The art of computer programming*, volume 2 – Seminumerical Algorithms. Addison Wesley Longman, 3 edition, 1998.

[51] R. Koekoek and R. F. Swattouw. The askey–scheme of hypergeometric orthogonal polynomials and its q–analogue. Technical report, Delft University of Technology, 1994, no. 94–05. (http://aw.twi.tudelft.nl/~koekoek/askey.html).

[52] William Judson LeVeque. *Topics in number theory. Vol. I, II*. Dover Publications Inc., Mineola, NY, 2002. Reprint of the 1956 original [Addison-Wesley Publishing Co., Inc., Reading, Mass.], with separate errata list for this edition by the author.

[53] Rudolf Lidl and Harald Niederreiter. *Finite fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 1997. With a foreword by P. M. Cohn.

[54] Rudolf Lidl and Günter Pilz. *Applied abstract algebra*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1998.

[55] D. Lorenzini. *An invitation to arithmetic geometry*, volume 9 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1996.

[56] R. Ludwig and J. Taylor. *Voyager Telecommunications*. NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, 2002.

[57] David J. C. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inform. Theory*, 45(2):399–431, 1999.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

[58] David J. C. MacKay. Errata for: "Good error-correcting codes based on very sparse matrices" [IEEE Trans. Inform. Theory **45** (1999), no. 2, 399–431; MR1677007 (99j:94077)]. *IEEE Trans. Inform. Theory*, 47(5):2101, 2001.

[59] David J.C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Cambridge (UK), 2004. (http://www.inference.phy.cam.ac.uk/mackay/itila/).

[60] J. L. Massey. Threshold decoding. Technical Report 410, Massachusetts Institute of Technology (Boston), 1963. (http://hdl.handle.net/1721.1/4415).

[61] James L. Massey. Applied digital information theory. ETH Zurich, 1998. (http://www.isi.ee.ethz.ch/education/public/free_docs.en.html).

[62] Francesco Mazzocca. Appunti di geometria superiore. Caserta, 2004. (http://www.dimat.unina2.it/mazzocca/Geom_Sup.htm).

[63] R. J. McEliece. The algebraic theory of convolutional codes. California Institute of Technology, May 1996.

[64] R. J. McEliece. *The theory of information and coding*, volume 86 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 2002.

[65] R. J. McEliece. The guruswami–sudan decoding algorithm for reed–solomon codes. Technical report, JPL Interplanetary Network Progress Report 42–153, May 2003. (http://ipnpr.jpl.nasa.gov/progress_report/42-153/title.htm).

[66] Robert J. McEliece, Eugene R. Rodemich, Howard Rumsey, Jr., and Lloyd R. Welch. New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE Trans. Information Theory*, IT-23(2):157–166, 1977.

[67] Bhubaneswar Mishra. *Algorithmic algebra*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1993.

[68] R. H. Morelos-Zaragoza. *The Art of Error Correcting Coding*. Wiley and sons, New York, 2002.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

[69] C. Moreno. *Algebraic Curves Over Finite Fields*. Cambridge University Press, Cambridge, 1991.

[70] D.J Mudgway. *Uplink–Downlink — A History of the Deep Space Network*. NASA Office of External Relations, Wasington (DC), 2001. (http://history.nasa.gov/SP-4227/Uplink-Downlink.pdf).

[71] K. H. Powers. *A unified theory of information*. PhD thesis, M.I.T. Dept. of Electrical Engineering, 1956. (http://hdl.handle.net/1721.1/4771).

[72] Oliver Pretzel. *Codes and algebraic curves*, volume 8 of *Oxford Lecture Series in Mathematics and its Applications*. The Clarendon Press Oxford University Press, New York, 1998.

[73] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, June 1960.

[74] B. Reiffen. Sequential encoding and decoding for the discrete memoryless channel. Technical Report 374, Massachusetts Institute of Technology (Boston), 1960. (http://hdl.handle.net/1721.1/4448).

[75] Thomas J. Richardson and Rüdiger L. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Trans. Inform. Theory*, 47(2):638–656, 2001.

[76] Tom Richardson and Ruediger Urbanke. Modern coding theory. EPFL Lausanne, 2004. (http://lthcwww.epfl.ch/papers/ics.ps).

[77] D. Salomon. *Data Compression — The complete reference*. Springer–Verlag, 3 edition, 2004.

[78] Alex Samorodnitsky. On the optimum of Delsarte's linear program. *J. Combin. Theory Ser. A*, 96(2):261–287, 2001.

[79] C. B. Schleger and L. C. Pérez. *Trellis and Turbo Coding*. Wiley and sons, New York, 2004.

[80] A. Seidenberg. *Elements of the theory of algebraic curves*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1968.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

[81] J. G. Semple and L. Roth. *Introduction to algebraic geometry*. Oxford Science Publications. The Clarendon Press Oxford University Press, New York, 1985.

[82] J.-P. Serre. *A course in arithmetic*. Springer-Verlag, New York, 1973. Translated from the French, Graduate Texts in Mathematics, No. 7.

[83] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948.

[84] James Singer. A theorem in finite projective geometry and some applications to number theory. *Trans. Amer. Math. Soc.*, 43(3):377–385, 1938.

[85] Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42(6, part 1):1710–1722, 1996. Codes and complexity.

[86] A. N. Skorobogatov and S. G. Vlăduţ. On the decoding of algebraic–geometric codes. *IEEE Trans. Inf. theor.*, 36:1051–1060, 1990.

[87] S. A. Stepanov. *Codes on algebraic curves*. Kluwer Academic/Plenum Publishers, New York, 1999.

[88] W. Richard Stevens. *TCP/IP Illustrated — The implementation*, volume 2. Addison–Wesley, New York, 1993.

[89] W. Richard Stevens. *TCP/IP Illustrated — The protocols*, volume 1. Addison–Wesley, New York, 1993.

[90] Henning Stichtenoth. *Algebraic function fields and codes*. Universitext. Springer-Verlag, Berlin, 1993.

[91] M. Sudan. Algorithmic introduction to coding theory. Massachusetts Institute of Technology (Boston), 2002. (http://theory.lcs.mit.edu/~madhu/FT01/).

[92] Y. Suhov. Lecture notes on algebraic coding theory. University of Cambridge, January 2003. (http://www.statslab.com.ac.uk/~yms/).

[93] R. Michael Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, 27(5):533–547, 1981.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

[94] M. A. Tsfasman and S. G. Vlăduţ. *Algebraic-geometric codes*, volume 58 of *Mathematics and its Applications (Soviet Series)*. Kluwer Academic Publishers Group, Dordrecht, 1991. Translated from the Russian by the authors.

[95] M. A. Tsfasman, S. G. Vlăduţ, and T. Zink. Modular curves, Shimura curves and Goppa codes better than the Varshamov-Gilbert bound. *Math. Nachr.*, 109:21–28, 1982.

[96] J. H. van Lint. *Introduction to coding theory*, volume 86 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, second edition, 1992.

[97] Alexander Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *STOC '97 (El Paso, TX)*, pages 92–109 (electronic). ACM, New York, 1999.

[98] A. J. Viterbi. Convolutional codes and their performace in communication systems. *IEEE Trans. Comm.*, COM-19(5):751–772, 1971.

[99] Pless V.S., W.C. Huffman, and Brualdi R.A., editors. *Handbook of Coding Theory*. Elsevier, 1998.

[100] J. L. Walker. *Codes and curves*, volume 7 of *Student Mathematical Library*. American Mathematical Society, Providence, RI, 2000. IAS/Park City Mathematical Subseries (http://www.math.unl.edu/~jwalker/).

[101] R. J. Walker. *Algebraic curves*. Springer-Verlag, New York, 1978.

[102] Harold N. Ward and Jay A. Wood. Characters and the equivalence of codes. *J. Combin. Theory Ser. A*, 73(2):348–352, 1996.

[103] Dominic Welsh. *Codes and cryptography*. Oxford Science Publications. The Clarendon Press Oxford University Press, New York, 1988.

[104] S. G. Wilson. *Digital Modulation and Coding*. Prentice Hall, 1995.

[105] P. Wocjan. The brill–noether algorithm: Construction of geometric goppa codes and absolute factorization. Master's thesis, University of Kalsruhe, 1999. http://www.cs.caltech.edu/~wocjan/.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi

[106] Jay A. Wood. Extension theorems for linear codes over finite rings. In *Applied algebra, algebraic algorithms and error-correcting codes (Toulouse, 1997)*, volume 1255 of *Lecture Notes in Comput. Sci.*, pages 329–340. Springer, Berlin, 1997.

[107] J. R. Wozencraft. Sequential decoding for reliable communication. Technical Report 325, Massachusetts Institute of Technology (Boston), 1957. (http://hdl.handle.net/1721.1/4758).

[108] K. Sh. Zigangirov. *Theory of Code Division Multiple Access Communication*. Wiley and sons, New York, 2004.

Bozza 3 giugno 2006 -- Bozza 3 giugno 2006 -- Bozza 3 gi