Algoritmi Randomizzati per parole di peso minimo

Camilla Ferretti

Dip. di Scienze Economiche e Sociali, UCSC Piacenza

Trento, 10 Marzo 2011

Problema della distanza minima

Cerchiamo un algoritmo efficiente per calcolare il parametro d del codice $C[n, k]$.

Distanza minima

Il calcolo della distanza minima di un codice lineare binario è un problema **NP-hard** [Vardy(1997)]

Distribuzione dei pesi

Dato un codice lineare $C[n, k]$ con matrice generatrice G, e il parametro $w \leq n$, vogliamo trovare una parola di peso (inferiore o) uguale a w. (la versione decisionale è un problema **NP-completo**, [Berlekamp *et al.*(1978)])

Un esempio di applicazione

La conoscenza del peso minimo di un codice serve per attaccare critto-sistemi di McEliece/Niederreiter:

McEliece:

- Chiave privata: codice C con matrice generatrice G e capacità di correzione t, S matrice $k \times k$ random invertibile, P matrice $n \times n$ random di permutazione.
- Chiave pubblica: $G' = SGP$ e t.
- Codifica: $x = mG' + e$, con e errore random di peso t.
- Decodifica: $xP^{-1} = mSG + eP^{-1}$, con un algoritmo di decodifica otteniamo mS ed infine $m = (mS)S^{-1}$.

Niederreiter utilizza la matrice di parità H al posto di G.

Attacco a McEliece e Niederreiter

C viene scelto in modo che C' non sia distinguibile da un codice lineare random, cioè G' non dà informazioni sulla struttura di C (es: codici di Goppa).
Decodificare C' equivale a trovare e, l'unica parola di peso minimo nel codice $C''[n, k+1]$:

$$C'' = C' \oplus x = \{c + x | c \in C'\}$$

che ha matrice generatrice $\left[\frac{G'}{x}\right]$

Ipotesi di base

Nell'affrontare il problema abbiamo supposto che:

Hp1 Il codice C è stato estratto casualmente dalla famiglia di tutti i codici lineari $n \times k$.

Hp2 Il codice C contiene almeno una parola di peso (inferiore o) uguale a w.

Algoritmi deterministici

Un algoritmo è *deterministico* se la successione di operazioni da svolgere è fissata a priori.

Algoritmo di Brouwer

L'Algoritmo di Brouwer controlla tutte le righe di G, tutte le somme di due righe, tutte le somme di tre righe...
[Bosma *et al.*(1997)]

Algoritmi Randomizzati

Un algoritmo è *randomizzato* se una o più istruzioni si basano sulla realizzazione di una specifica variabile aleatoria.
Il numero di iterazioni (o tempo impiegato) N è anch'esso una v.a. e l'efficienza dell'algoritmo è misurata dal valore medio $\bar{N}$.

Algoritmo CC

1. Dati $G = [I|R]$, w e p;
2. controlla tutte le somme di $2p$ righe di G;
   - 2.1 se una ha peso w, STOP
   - 2.2 altrimenti scambia una colonna random di I con una colonna random di R e applica Gauss per trovare $G' = [I|R']$.

[Chabaud and Canteaut(1998)]

Equivalenza fra codici

Invece di enumerare le parole di C, AlgCC si muove fra codici equivalenti, finchè la parola richiesta non è somma di poche righe di G. Infatti se $n \to \infty$ (con $r = \frac{k}{n}$ costante):

- se $w \leq w_c(r)$ vale che

$$\frac{\text{nr. medio di parole di peso } w}{2^k} \to 0$$

- invece per i codici equivalenti vale che

$$\frac{\text{nr. } G \text{ equivalenti con una riga di peso } w}{n!} \to \text{cost.}$$

Proposta di algoritmo randomizzato

Vogliamo modificare AlgCC per migliorarne l'efficienza.

Osservazione

Dati C e $p > 1$ non è sempre vero che esiste una base sistematica di C tale che la parola cercata è somma di esattamente p vettori di tale base.

Esempio

Sia $C = \{(1,0,1,1,1,0,0),(0,1,0,0,0,1,1),(1,1,1,1,1,1,1),(0,0,0,0,0,0,0)\}$, *in questo caso non è mai possibile scrivere la parola di peso minimo* $(0,1,0,0,0,1,1)$ *come somma di due parole dello stesso codice che costituiscano una base in forma sistematica.*

L'algoritmo SF(m, T)

Introduciamo due nuovi parametri: il numero di colonne da scambiare m e il numero di controlli da fare T:

Algoritmo SF(m, T)

1. Dati $G = [I|R]$, w, m, T, $Nmax$
2. per $i = 1, \ldots, Nmax$
   - 2.1 scegli m colonne da I e m colonne da R ($\rightarrow$ random!)
   - 2.2 scambia le colonne scelte: $G \rightarrow G'$
   - 2.3 applica la riduzione di Gauss: $G' \rightarrow G'' = [I|R'']$
   - 2.4 per $j = 1, \ldots, T$
     - 2.4.1 controlla le somme di j righe di G''
     - 2.4.2 se una ha peso w STOP

Analisi di SF(m, T)

Per valutare SF(m, T) dobbiamo stabilire se:

- il criterio di arresto è ben definito;
- l'algoritmo converge alla soluzione;
- il tempo medio di esecuzione è accettabile.

Il criterio di arresto

Osservazione

Per ogni parola in C che soddisfa il Bound di Singleton ($w \leq n - k + 1$) esiste, a meno di equivalenze, una base in forma sistematica che contiene tale parola.

Questa osservazione supporta la scelta di fermare l'algoritmo quando la parola richiesta compare come riga della matrice generatrice.

Per incrementare la velocità controlliamo ad ogni passo tutte le somme di $1, 2, \ldots, T$ righe della matrice.

Convergenza

Per dimostrare la convergenza consideriamo:

- la parola c di peso w e la sua versione equivalente $c^{(t)}$ al t-esimo passo,
- l'insieme $S = \{1, \ldots, k\}$,
- la v.a $X_t = w(c^{(t)})|_S$ che indica il peso di $c^{(t)}$ sulle sue prime k coordinate,
- l'insieme finito $\{1, \ldots, \min\{k, w\}\}$ di valori assumibili da X.

X_t evolve secondo una catena di Markov su un insieme finito di stati, gli stati da 1 a T sono *assorbenti*. Un risultato classico delle catene di Markov assicura che con probabilità 1 la catena tocca uno degli stati assorbenti dopo un tempo finito.

Il tempo medio di esecuzione (I)

Il tempo medio di esecuzione è conseguenza di due fattori:

1. La distribuzione di probabilità del punto iniziale: $\pi_i = \mathbb{P}(X_0 = i)$.
2. Il numero di iterazioni N dato il punto iniziale: $\tau_i = \mathbb{P}(N|X_0 = i)$.

Il tempo atteso globale di esecuzione è dato da

$$\bar{N} = \langle \pi, \tau \rangle = \sum \pi_i \tau_i$$

Il tempo medio di esecuzione (II)

La distribuzione iniziale è data da:

$$\pi_i = \binom{k}{i}\binom{n-k}{w-i}/Z$$

Il vettore τ può essere calcolato esattamente per $m = 1$, sfruttando la teoria delle Catene di Markov. Riunificando i risultati otteniamo:

$$\bar{N} = \frac{k(n-k)}{Z} \cdot \sum_{l=T+1}^{\min\{k,w\}} \left\{ \frac{1}{l(n-k-w+l)} \cdot \right.$$

$$\left. \cdot \frac{1}{\binom{k}{l}\binom{n-k}{w-l}} \right\} \cdot \left[\sum_{j=l}^{\min\{k,w\}} \binom{k}{j}\binom{n-k}{w-j} \right]^2$$

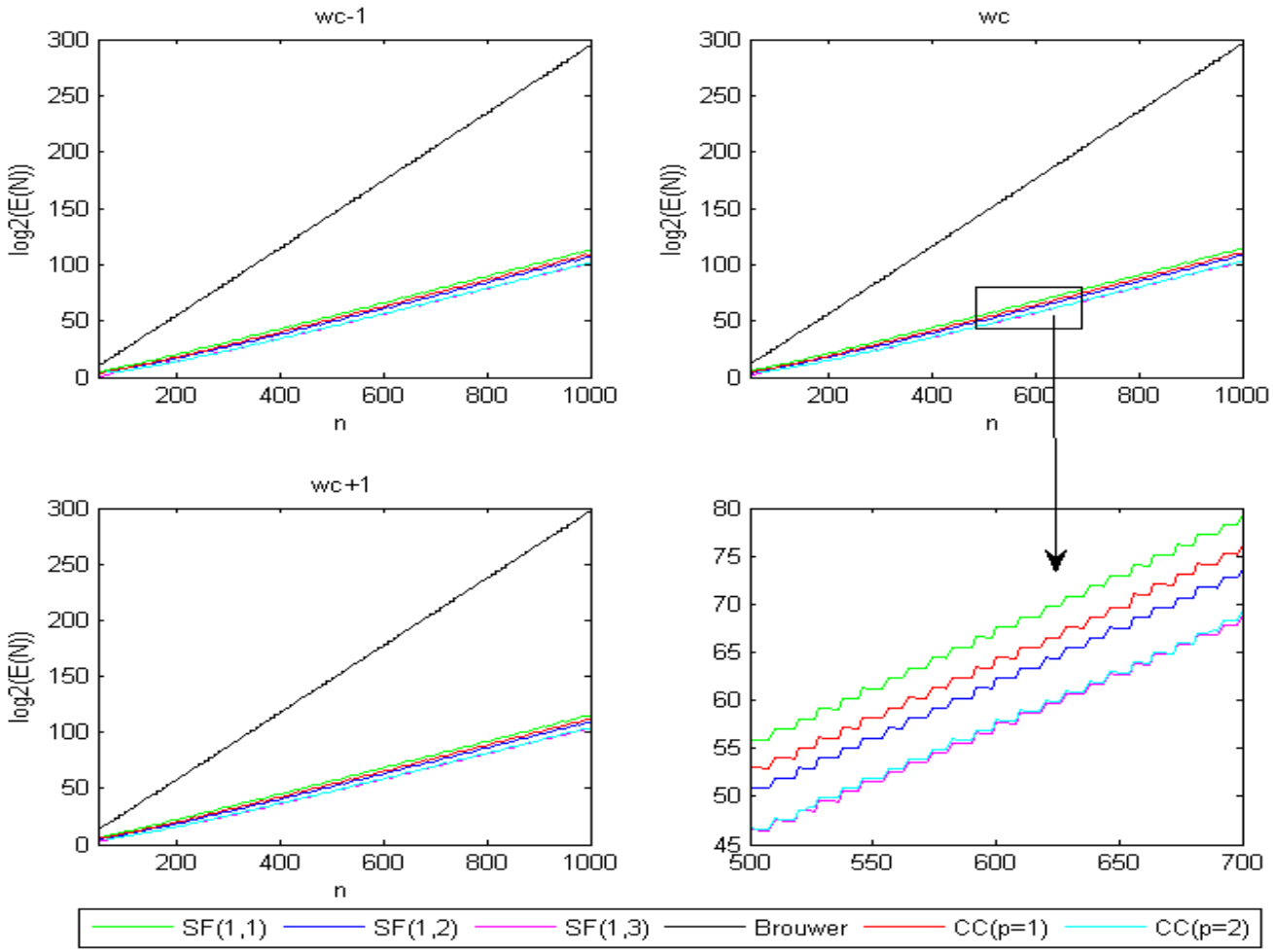
Confronto fra algoritmi (I)

Adattando la formula precedente ad AlgCC abbiamo confrontato il tempo medio per alcuni codici ($n = 2k$):

	Tempo medio						
k, w, p	32;7;1	64;15;1	128;29;1	256;57;1	384;85:2	512;113;2	768;170;2
CC	5.5e+1	6.9e+3	7.5e+7	4.1e+16	2.3e+23	1.4e+32	1.9e+50
SF$(1,1)$	1.4e+2	2.3e+4	4.3e+8	3.1e+17	3.1e+26	3.5e+35	1.2e+54
SF$(1,2)$	3.6e+1	3.3e+3	2.6e+7	9.2e+15	5.9e+24	5e+33	1.1e+52
SF$(1,3)$	9.9e+0	7.9e+2	2.6e+6	4.1e+14	1.8e+23	1.2e+32	1.6e+50
	Costo per iterazione						
k, w, p	32;7;1	64;15;1	128;29;1	256;57;1	384;85:2	512;113;2	768;170;2
CC	2.6e+3	7.4e+3	2.3e+4	7.1e+4	1e+7	2.1e+7	5.8e+7
SF$(1,1)$	1.5e+3	6.1e+3	2.5e+4	9.8e+4	2.2e+5	3.9e+5	8.8e+5
SF$(1,2)$	1.7e+4	1.4e+5	1.1e+6	8.5e+6	2.8e+7	6.7e+7	2.3e+8
SF$(1,3)$	1.8e+5	2.8e+6	4.5e+7	7.2e+8	3.6e+9	1.1e+10	5.8e+10

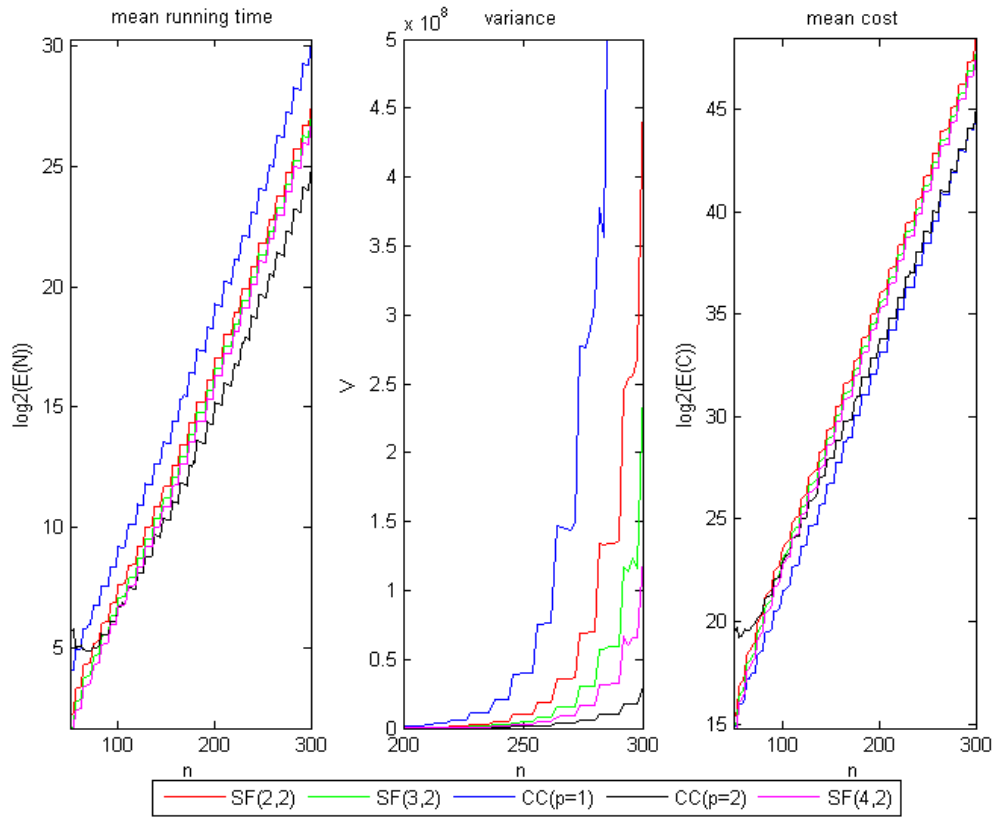
Confronto fra algoritmi (II)

I grafici mostrano la funzione $\log_2 \bar{N}$ al variare di n per $w \approx w_c$ e $r = 0.5$:



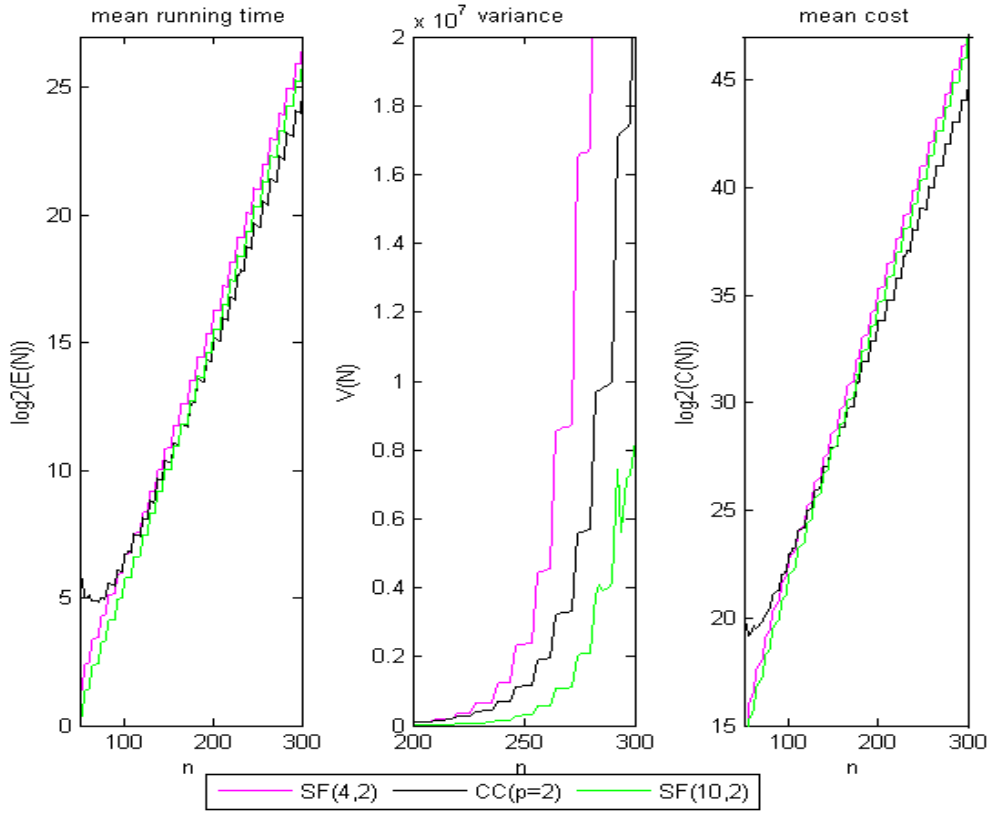
Simulazioni (I)

I grafici mostrano $\log_2 \bar{N}$ ottenuto nelle simulazioni con $T = 2$, al variare di $m = 2, 3, 4$:



Simulazioni (II)

Tempo medio con $T = 2$, al variare di $m = 4, 10$:



Approfondimenti

Fra le questioni da approfondire segnaliamo:

- l'ottimizzazione del costo computazionale;
- la scelta del numero massimo di iterazioni per dedurre che la parola cercata non esiste.

Approfondimenti

Fra le questioni da approfondire segnaliamo:

- l'ottimizzazione del costo computazionale;
- la scelta del numero massimo di iterazioni per dedurre che la parola cercata non esiste.

GRAZIE

Bibliografia


Berlekamp, E., McEliece, R., and VanTilborg, H. (1978).
On the inherent intractability of certain coding problems.
IEEE T. Inform. Theory, **IT-24**, 384–386.


Bosma, W., Cannon, J., and Playoust, C. (1997).
The Magma algebra system I. The user language.


Chabaud, A. and Canteaut, F. (1998).
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 T. Inform. Theory, **44**(1).


Vardy, A. (1997).
The intractability of computing the Minimum Distance of a Code.
IEEE Trans. Inf. Theory, **43**(6), 1757–1766.