Grafi Expanders e Codici di Correzione d'Errore

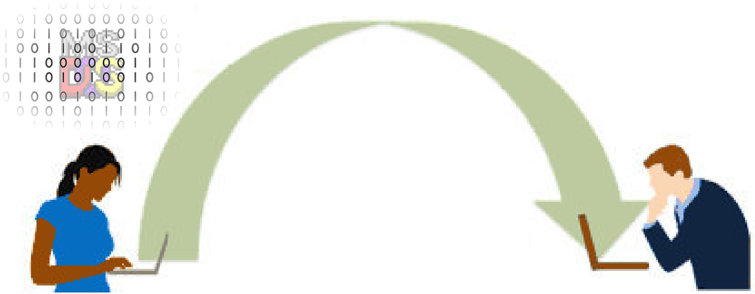
Barbarino Giovanni

Università di Pisa

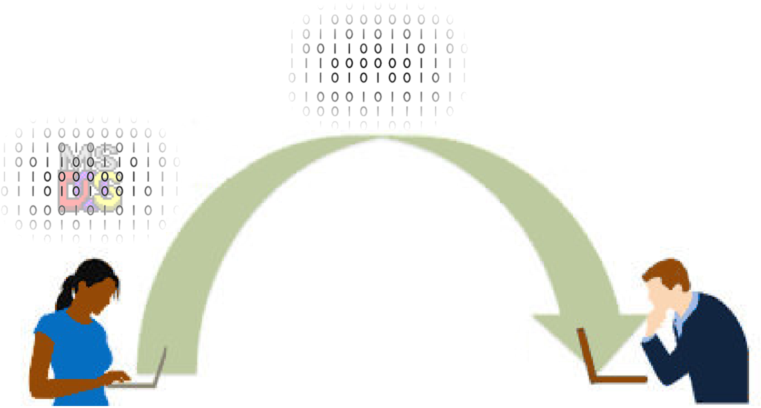
19 Settembre 2014

Trasmissione di Dati

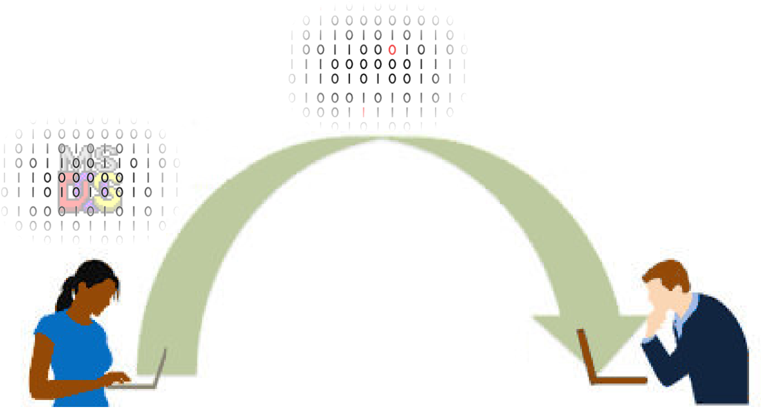
Trasmissione di Dati



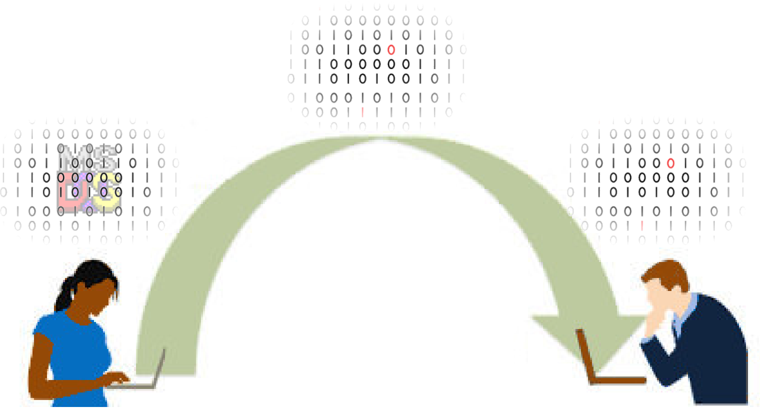
Trasmissione di Dati



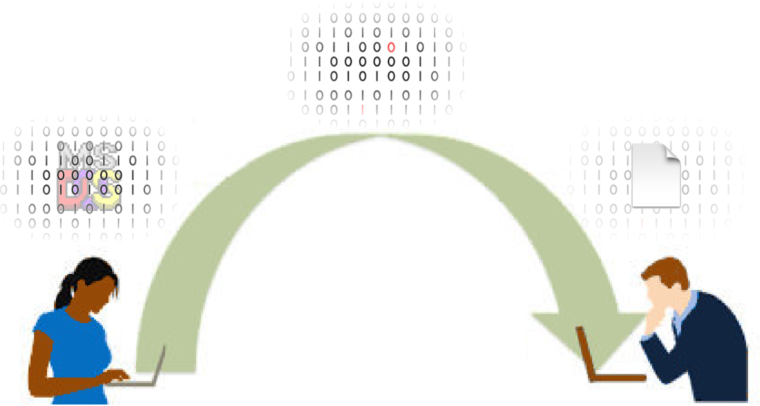
Trasmissione di Dati



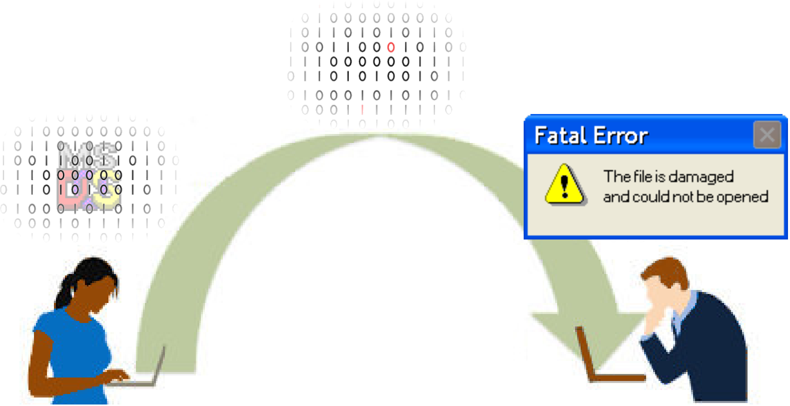
Trasmissione di Dati



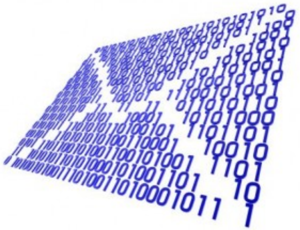
Trasmissione di Dati



Trasmissione di Dati



Le Specifiche



- Il Mittente trasmette un messaggio binario di lunghezza n

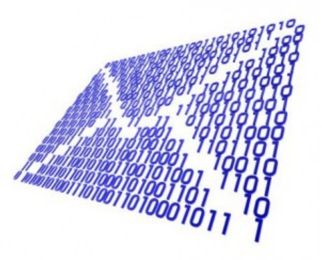
$$\underbrace{101010111010110}_{n}$$

- Il messaggio viene corrotto al massimo di un fattore $p < 1$

$$\underbrace{10101\overbrace{10}^{\leq pn}11010110}_{n}$$

- Come fa il Destinatario a correggere l'errore?

Le Specifiche



- Il Mittente trasmette un messaggio binario di lunghezza n

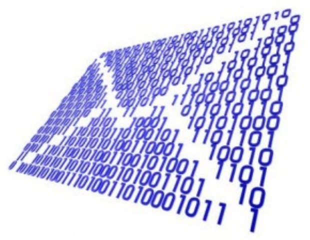
$$\underbrace{101010111010110}_{n}$$

- Il messaggio viene corrotto al massimo di un fattore $p < 1$

$$\underbrace{10101\ \overbrace{10}^{\leq pn}\ 11010110}_{n}$$

- Come fa il Destinatario a correggere l'errore?

Le Specifiche



- Il Mittente trasmette un messaggio binario di lunghezza n

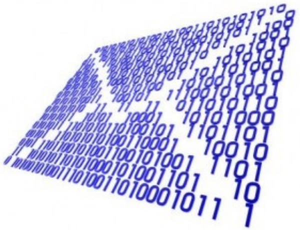
$$\underbrace{101010111010110}_{n}$$

- Il messaggio viene corrotto al massimo di un fattore $p < 1$

$$\underbrace{10101\overbrace{\color{red}{10}}^{\leq pn}11010110}_{n}$$

- Come fa il Destinatario a correggere l'errore?

Le Specifiche



- Il Mittente trasmette un messaggio binario di lunghezza n

$$\underbrace{101010111010110}_{n}$$

- Il messaggio viene corrotto al massimo di un fattore $p < 1$

$$\underbrace{10101\overbrace{\color{red}{10}}^{\leq pn}11010110}_{n}$$

- Come fa il Destinatario a correggere l'errore?

La Storia del Forward Error Correction

- Nel 1948 Shannon e Hamming presentano il problema, dando vita ai primi **Codici Lineari di Detenzione e Correzione dell'Errore**
- Nel 1960, Gallager creò i **low-density parity-check code (LDPC)**, usando grafi sparsi bipartiti
- Berlekamp e Massey, nel 1969, trovarono un algoritmo di decodifica efficiente dei **Reed–Solomon codes (RSC)**, promuovendone la diffusione
- I **Turbo Codes** nacquero nel 1993 grazie a Berrou, Glavieux, e Thitimajshima, sorpassando i precedenti
- Grazie agli studi sui grafi di Bassalygo, Pinsker e Margulis, nel 1996 i LDPC divennero molto più efficienti, rivaleggiando l'avanzata dei Turbo Codes

La Storia del Forward Error Correction

- Nel 1948 Shannon e Hamming presentano il problema, dando vita ai primi **Codici Lineari di Detenzione e Correzione dell'Errore**
- Nel 1960, Gallager creò i **low-density parity-check code (LDPC)**, usando grafi sparsi bipartiti
- Berlekamp e Massey, nel 1969, trovarono un algoritmo di decodifica efficiente dei **Reed–Solomon codes (RSC)**, promuovendone la diffusione
- I **Turbo Codes** nacquero nel 1993 grazie a Berrou, Glavieux, e Thitimajshima, sorpassando i precedenti
- Grazie agli studi sui grafi di Bassalygo, Pinsker e Margulis, nel 1996 i LDPC divennero molto più efficienti, rivaleggiando l'avanzata dei Turbo Codes

La Storia del Forward Error Correction

- Nel 1948 Shannon e Hamming presentano il problema, dando vita ai primi **Codici Lineari di Detenzione e Correzione dell'Errore**
- Nel 1960, Gallager creò i **low-density parity-check code (LDPC)**, usando grafi sparsi bipartiti
- Berlekamp e Massey, nel 1969, trovarono un algoritmo di decodifica efficiente dei **Reed–Solomon codes (RSC)**, promuovendone la diffusione
- I **Turbo Codes** nacquero nel 1993 grazie a Berrou, Glavieux, e Thitimajshima, sorpassando i precedenti
- Grazie agli studi sui grafi di Bassalygo, Pinsker e Margulis, nel 1996 i LDPC divennero molto più efficienti, rivaleggiando l'avanzata dei Turbo Codes

La Storia del Forward Error Correction

- Nel 1948 Shannon e Hamming presentano il problema, dando vita ai primi **Codici Lineari di Detenzione e Correzione dell'Errore**
- Nel 1960, Gallager creò i **low-density parity-check code (LDPC)**, usando grafi sparsi bipartiti
- Berlekamp e Massey, nel 1969, trovarono un algoritmo di decodifica efficiente dei **Reed–Solomon codes (RSC)**, promuovendone la diffusione
- I **Turbo Codes** nacquero nel 1993 grazie a Berrou, Glavieux, e Thitimajshima, sorpassando i precedenti
- Grazie agli studi sui grafi di Bassalygo, Pinsker e Margulis, nel 1996 i LDPC divennero molto più efficienti, rivaleggiando l'avanzata dei Turbo Codes

La Storia del Forward Error Correction

- Nel 1948 Shannon e Hamming presentano il problema, dando vita ai primi **Codici Lineari di Detenzione e Correzione dell'Errore**
- Nel 1960, Gallager creò i **low-density parity-check code (LDPC)**, usando grafi sparsi bipartiti
- Berlekamp e Massey, nel 1969, trovarono un algoritmo di decodifica efficiente dei **Reed–Solomon codes (RSC)**, promuovendone la diffusione
- I **Turbo Codes** nacquero nel 1993 grazie a Berrou, Glavieux, e Thitimajshima, sorpassando i precedenti
- Grazie agli studi sui grafi di Bassalygo, Pinsker e Margulis, nel 1996 i LDPC divennero molto più efficienti, rivaleggiando l'avanzata dei Turbo Codes

La Storia del Forward Error Correction

- Nel 1948 Shannon e Hamming presentano il problema, dando vita ai primi **Codici Lineari di Detenzione e Correzione dell'Errore**
- Nel 1960, Gallager creò i **low-density parity-check code (LDPC)**, usando grafi sparsi bipartiti
- Berlekamp e Massey, nel 1969, trovarono un algoritmo di decodifica efficiente dei **Reed–Solomon codes (RSC)**, promuovendone la diffusione
- I **Turbo Codes** nacquero nel 1993 grazie a Berrou, Glavieux, e Thitimajshima, sorpassando i precedenti
- Grazie agli studi sui grafi di Bassalygo, Pinsker e Margulis, nel 1996 i LDPC divennero molto più efficienti, rivaleggiando l'avanzata dei Turbo Codes

Distanza di Hamming

Definizione

Dato lo spazio $\mathbb{F}_2^n$ delle stringhe binarie di lunghezza n, la **Distanza di Hamming** tra due elementi è il numero di bit per cui differiscono.

$$d_H(x,y) = sum(xor(x,y)) = \|x-y\|_1$$

$$x = 000100101000$$

$$y = 010010111001$$

$$d_H(x,y) = 5$$

Distanza di Hamming

Definizione

Dato lo spazio $\mathbb{F}_2^n$ delle stringhe binarie di lunghezza n, la **Distanza di Hamming** tra due elementi è il numero di bit per cui differiscono.

$$d_H(x, y) = sum(xor(x, y)) = \|x - y\|_1$$

$$x = 000100101000$$

$$y = 010010111001$$

$$d_H(x, y) = 5$$

Distanza di Hamming

Definizione

Dato lo spazio $\mathbb{F}_2^n$ delle stringhe binarie di lunghezza n, la **Distanza di Hamming** tra due elementi è il numero di bit per cui differiscono.

$$d_H(x, y) = sum(xor(x, y)) = \|x - y\|_1$$

$$x = 000100101000$$

$$y = 010010111001$$

$$d_H(x, y) = 5$$

Algoritmo

Problema I messaggi M sono stringhe di k bit con percentuale di errore p

Codifica Si cerca un **codice** $C \subseteq \mathbb{F}_2^n$, con $n \geq k$, in corrispondenza biunivoca con M

Trasmissione Il Mittente sceglie un messaggio in M, e trasmette la stringa in C corrispondente

Correzione Il Destinatario riceve il messaggio corrotto, e cerca in C la stringa con distanza minore

Decodifica Ricostruita la stringa corretta in C, essa individuerà il messaggio originale in M

Algoritmo

Problema I messaggi M sono stringhe di k bit con percentuale di errore p

Codifica Si cerca un **codice** $C \subseteq \mathbb{F}_2^n$, con $n \geq k$, in corrispondenza biunivoca con M

Trasmissione Il Mittente sceglie un messaggio in M, e trasmette la stringa in C corrispondente

Correzione Il Destinatario riceve il messaggio corrotto, e cerca in C la stringa con distanza minore

Decodifica Ricostruita la stringa corretta in C, essa individuerà il messaggio originale in M

Algoritmo

Problema I messaggi M sono stringhe di k bit con percentuale di errore p

Codifica Si cerca un **codice** $C \subseteq \mathbb{F}_2^n$, con $n \geq k$, in corrispondenza biunivoca con M

Trasmissione Il Mittente sceglie un messaggio in M, e trasmette la stringa in C corrispondente

Correzione Il Destinatario riceve il messaggio corrotto, e cerca in C la stringa con distanza minore

Decodifica Ricostruita la stringa corretta in C, essa individuerà il messaggio originale in M

Algoritmo

Problema I messaggi M sono stringhe di k bit con percentuale di errore p

Codifica Si cerca un **codice** $C \subseteq \mathbb{F}_2^n$, con $n \geq k$, in corrispondenza biunivoca con M

Trasmissione Il Mittente sceglie un messaggio in M, e trasmette la stringa in C corrispondente

Correzione Il Destinatario riceve il messaggio corrotto, e cerca in C la stringa con distanza minore

Decodifica Ricostruita la stringa corretta in C, essa individuerà il messaggio originale in M

Algoritmo

Problema I messaggi M sono stringhe di k bit con percentuale di errore p

Codifica Si cerca un **codice** $C \subseteq \mathbb{F}_2^n$, con $n \geq k$, in corrispondenza biunivoca con M

Trasmissione Il Mittente sceglie un messaggio in M, e trasmette la stringa in C corrispondente

Correzione Il Destinatario riceve il messaggio corrotto, e cerca in C la stringa con distanza minore

Decodifica Ricostruita la stringa corretta in C, essa individuerà il messaggio originale in M

Algoritmo

Teorema

L'algoritmo è corretto se e solo se per ogni coppia di stringhe distinte nel codice C si ha

$$d_H(x, y) > 2pn$$

Dimostrazione.

Se x è la stringa trasmessa, $\tilde{x}$ quella corrotta, e y una qualsiasi stringa in C diversa da x, allora

$$d_H(x, \tilde{x}) \leq pn$$

$$d_H(y, \tilde{x}) \geq d_H(x, y) - d_H(x, \tilde{x}) > 2pn - pn = pn$$

Dunque la stringa in C più vicina a $\tilde{x}$ è x $\square$

Algoritmo

Teorema

L'algoritmo è corretto se e solo se per ogni coppia di stringhe distinte nel codice C si ha

$$d_H(x, y) > 2pn$$

Dimostrazione.

Se x è la stringa trasmessa, $\tilde{x}$ quella corrotta, e y una qualsiasi stringa in C diversa da x, allora

$$d_H(x, \tilde{x}) \leq pn$$

$$d_H(y, \tilde{x}) \geq d_H(x, y) - d_H(x, \tilde{x}) > 2pn - pn = pn$$

Dunque la stringa in C più vicina a $\tilde{x}$ è x $\square$

Percentuale di Errore: $\frac{1}{5}$

	Originale
s_1	10101
s_2	10001
s_3	10111
s_4	10010
s_5	00101
s_6	00000
s_7	11011
s_8	01001

Percentuale di Errore: $\frac{1}{5}$

	Originale	Codice
s_1	10101	000000
s_2	10001	111000
s_3	10111	100110
s_4	10010	010101
s_5	00101	001011
s_6	00000	110011
s_7	11011	011110
s_8	01001	101101

Percentuale di Errore: $\frac{1}{5}$

	Originale	Codice
s_1	10101	000000
s_2	10001	111000
s_3	10111	100110
s_4	10010	010101
s_5	00101	001011
s_6	00000	110011
s_7	11011	011110
s_8	01001	101101

Percentuale di Errore: $\frac{1}{5}$

	Originale	Codice
s_1	10101	000000
s_2	10001	111000
s_3	10111	100110
s_4	10010	010101
s_5	00101	001011
s_6	00000	110011
s_7	11011	011110
s_8	01001	101101

Messaggio Originale
10010
$\downarrow$
Messaggio Codificato
010101
$\downarrow$
Messaggio Corrotto
011101

Percentuale di Errore: $\frac{1}{5}$

	Originale	Codice
s_1	10101	000000
s_2	10001	111000
s_3	10111	100110
s_4	10010	010101
s_5	00101	001011
s_6	00000	110011
s_7	11011	011110
s_8	01001	101101

Messaggio Originale
10010
$\downarrow$
Messaggio Codificato
010101
$\downarrow$
Messaggio Corrotto
011101

Percentuale di Errore: $\frac{1}{5}$

Messaggio Corrotto
01**1**101
↓
Messaggio Corretto
010101
↓
Messaggio Decodificato
10010

	Originale	Codice
s_1	10101	000000
s_2	10001	111000
s_3	10111	100110
s_4	10010	010101
s_5	00101	001011
s_6	00000	110011
s_7	11011	011110
s_8	01001	101101

Percentuale di Errore: $\frac{1}{5}$

Messaggio Corrotto
011101
$\downarrow$
Messaggio Corretto
010101
$\downarrow$
Messaggio Decodificato
10010

d_H	Corrotto	Codice
4		000000
3		111000
5		100110
1	011101	010101
3		001011
4		110011
2		011110
2		101101

Percentuale di Errore: $\frac{1}{5}$

Messaggio Corrotto

011101

↓

Messaggio Corretto

010101

↓

Messaggio Decodificato

10010

	Originale	Codice
s_1	10101	000000
s_2	10001	111000
s_3	10111	100110
s_4	10010	010101
s_5	00101	001011
s_6	00000	110011
s_7	11011	011110
s_8	01001	101101

Distanza

Distanza

Dato $C \subseteq \{0,1\}^n$ un codice, definiamo la sua **distanza** come la minima distanza tra due stringhe distinte fratto la loro lunghezza

$$D(C) = \frac{\min_{c_1 \neq c_2}^{c_1, c_2 \in C} d_H(c_1, c_2)}{n}$$

- Ogni codice che risolva il problema ha $D(C) > 2p$
- La distanza dà una misura di *correttezza* del codice

Distanza

Distanza

Dato $C \subseteq \{0,1\}^n$ un codice, definiamo la sua **distanza** come la minima distanza tra due stringhe distinte fratto la loro lunghezza

$$D(C) = \frac{\min_{c_1 \neq c_2}^{c_1, c_2 \in C} d_H(c_1, c_2)}{n}$$

- Ogni codice che risolva il problema ha $D(C) > 2p$
- La distanza dà una misura di *correttezza* del codice

Distanza

Distanza

Dato $C \subseteq \{0,1\}^n$ un codice, definiamo la sua **distanza** come la minima distanza tra due stringhe distinte fratto la loro lunghezza

$$D(C) = \frac{\min^{c_1, c_2 \in C}_{c_1 \neq c_2} d_H(c_1, c_2)}{n}$$

- Ogni codice che risolva il problema ha $D(C) > 2p$
- La distanza dà una misura di *correttezza* del codice

Distanza

Distanza

Dato $C \subseteq \{0,1\}^n$ un codice, definiamo la sua **distanza** come la minima distanza tra due stringhe distinte fratto la loro lunghezza

$$D(C) = \frac{\min_{c_1 \neq c_2}^{c_1, c_2 \in C} d_H(c_1, c_2)}{n}$$

- Ogni codice che risolva il problema ha $D(C) > 2p$
- La distanza dà una misura di *correttezza* del codice

Rate

Definizione

Dato $C \subseteq \{0,1\}^n$ un codice, definiamo il suo **rate** come

$$R(C) = \frac{\log_2 |C|}{n}$$

- Più grande è $|C|$, più messaggi possiamo codificare
- n è il numero di bit trasmessi, che vorremmo essere piccolo
- $R(C)$ è una misura dell'*efficienza* del codice

Rate

Definizione

Dato $C \subseteq \{0,1\}^n$ un codice, definiamo il suo **rate** come

$$R(C) = \frac{\log_2 |C|}{n}$$

- Più grande è $|C|$, più messaggi possiamo codificare
- n è il numero di bit trasmessi, che vorremmo essere piccolo
- $R(C)$ è una misura dell'efficienza del codice

Rate

Definizione

Dato $C \subseteq \{0,1\}^n$ un codice, definiamo il suo **rate** come

$$R(C) = \frac{\log_2 |C|}{n}$$

- Più grande è $|C|$, più messaggi possiamo codificare
- n è il numero di bit trasmessi, che vorremmo essere piccolo
- $R(C)$ è una misura dell'*efficienza* del codice

Rate

Definizione

Dato $C \subseteq \{0,1\}^n$ un codice, definiamo il suo **rate** come

$$R(C) = \frac{\log_2 |C|}{n}$$

- Più grande è $|C|$, più messaggi possiamo codificare
- n è il numero di bit trasmessi, che vorremmo essere piccolo
- $R(C)$ è una misura dell'*efficienza* del codice

Rate

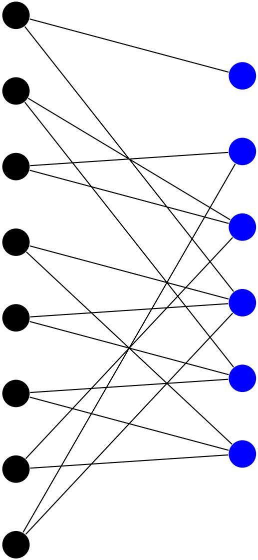
Definizione

Dato $C \subseteq \{0,1\}^n$ un codice, definiamo il suo **rate** come

$$R(C) = \frac{\log_2 |C|}{n}$$

- Più grande è $|C|$, più messaggi possiamo codificare
- n è il numero di bit trasmessi, che vorremmo essere piccolo
- $R(C)$ è una misura dell'*efficienza* del codice

Grafi Bipartiti



Grafo $G = (V, E)$

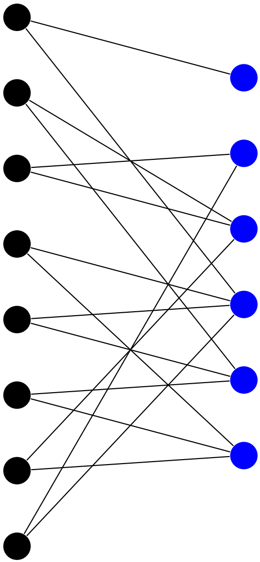
Bipartito $L \coprod R = V$

Codice Generato

C è l'insieme delle stringhe $x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8$ tali che

$$\begin{cases} x_1 = 0 \\ x_3 + x_8 = 0 \\ x_2 + x_3 + x_7 = 0 \\ x_1 + x_4 + x_5 + x_8 = 0 \\ x_2 + x_5 + x_6 = 0 \\ x_4 + x_6 + x_7 = 0 \end{cases}$$

Grafi Bipartiti



Grafo $G = (V, E)$

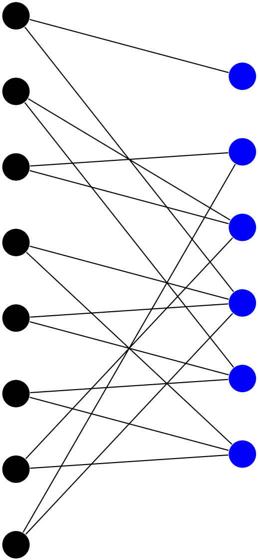
Bipartito $L \coprod R = V$

Codice Generato

C è l'insieme delle stringhe $x_1x_2x_3x_4x_5x_6x_7x_8$ tali che

$$\begin{cases} x_1 = 0 \\ x_3 + x_8 = 0 \\ x_2 + x_3 + x_7 = 0 \\ x_1 + x_4 + x_5 + x_8 = 0 \\ x_2 + x_5 + x_6 = 0 \\ x_4 + x_6 + x_7 = 0 \end{cases}$$

Grafi Bipartiti



Grafo $G = (V, E)$

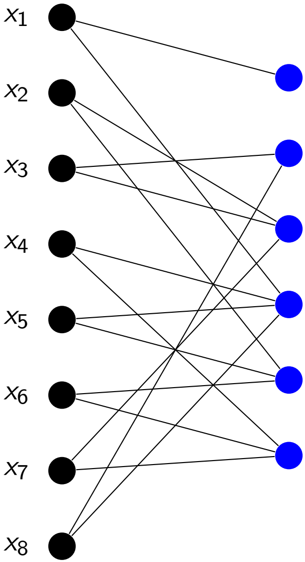
Bipartito $L \coprod R = V$

Codice Generato

C è l'insieme delle stringhe $x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8$ tali che

$$\begin{cases} x_1 = 0 \\ x_3 + x_8 = 0 \\ x_2 + x_3 + x_7 = 0 \\ x_1 + x_4 + x_5 + x_8 = 0 \\ x_2 + x_5 + x_6 = 0 \\ x_4 + x_6 + x_7 = 0 \end{cases}$$

Grafi Bipartiti



Grafo $G = (V, E)$

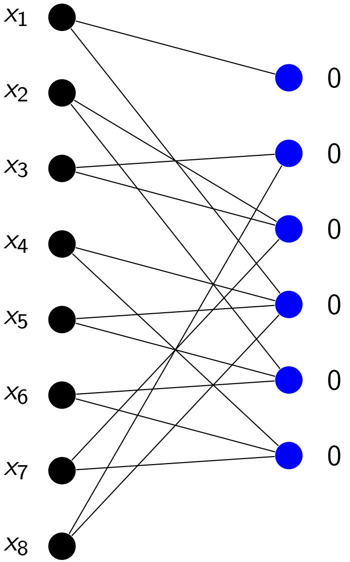
Bipartito $L \coprod R = V$

Codice Generato

C è l'insieme delle stringhe $x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8$ tali che

$$\begin{cases} x_1 = 0 \\ x_3 + x_8 = 0 \\ x_2 + x_3 + x_7 = 0 \\ x_1 + x_4 + x_5 + x_8 = 0 \\ x_2 + x_5 + x_6 = 0 \\ x_4 + x_6 + x_7 = 0 \end{cases}$$

Grafi Bipartiti



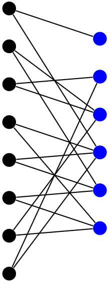
Grafo $G = (V, E)$
Bipartito $L \coprod R = V$

Codice Generato

C è l'insieme delle stringhe $x_1x_2x_3x_4x_5x_6x_7x_8$ tali che

$$\begin{cases} x_1 = 0 \\ x_3 + x_8 = 0 \\ x_2 + x_3 + x_7 = 0 \\ x_1 + x_4 + x_5 + x_8 = 0 \\ x_2 + x_5 + x_6 = 0 \\ x_4 + x_6 + x_7 = 0 \end{cases}$$

Grafi Magici e Left Expansion



$(8, 6, 2)$
grafo magico

Definizione

Dato G bipartito, allora è detto un (n, m, d)-**grafo magico** se valgono le seguenti:

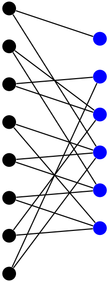
- $|L| = n, \quad |R| = m, \quad d$-regolare a sinistra
- $S \subseteq L, \quad |S| \leq n/10d \implies |N(S)| \geq 5d|S|/8$
- $S \subseteq L, \quad n/10d < |S| \leq n/2 \implies |N(S)| \geq |S|$

Definizione

Dato G bipartito, la sua Left Vertex Expansion Ratio è

$$L(G, k) = \min_{\substack{0<|S|\leq k \\ S \subseteq L}} \frac{|N(S)|}{|S|}$$

Grafi Magici e Left Expansion



$(8, 6, 2)$
grafo magico

Definizione

Dato G bipartito, allora è detto un (n, m, d)-**grafo magico** se valgono le seguenti:

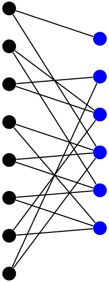
- $|L| = n, \quad |R| = m, \quad d$-regolare a sinistra
- $S \subseteq L, \quad |S| \leq n/10d \implies |N(S)| \geq 5d|S|/8$
- $S \subseteq L, \quad n/10d < |S| \leq n/2 \implies |N(S)| \geq |S|$

Definizione

Dato G bipartito, la sua **Left Vertex Expansion Ratio** è

$$L(G, k) = \min_{\substack{0<|S|\leq k \\ S\subseteq L}} \frac{|N(S)|}{|S|}$$

Grafi Magici e Left Expansion



$(8, 6, 2)$
grafo magico

Definizione

Dato G bipartito, allora è detto un (n, m, d)-**grafo magico** se valgono le seguenti:

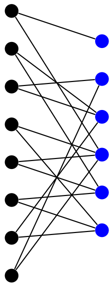
- $|L| = n, \quad |R| = m, \quad d$-regolare a sinistra
- $S \subseteq L, \quad |S| \leq n/10d \implies |N(S)| \geq 5d|S|/8$
- $S \subseteq L, \quad n/10d < |S| \leq n/2 \implies |N(S)| \geq |S|$

Definizione

Dato G bipartito, la sua **Left Vertex Expansion Ratio** è

$$L(G, k) = \min_{\substack{0<|S|\leq k \\ S\subseteq L}} \frac{|N(S)|}{|S|}$$

Grafi Magici e Left Expansion



$(8, 6, 2)$
grafo magico

Definizione

Dato G bipartito, allora è detto un (n, m, d)-**grafo magico** se valgono le seguenti:

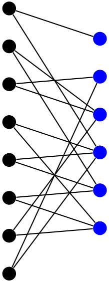
- $|L| = n, \quad |R| = m, \quad d$-regolare a sinistra
- $S \subseteq L, \quad |S| \leq n/10d \implies |N(S)| \geq 5d|S|/8$
- $S \subseteq L, \quad n/10d < |S| \leq n/2 \implies |N(S)| \geq |S|$

Definizione

Dato G bipartito, la sua **Left Vertex Expansion Ratio** è

$$L(G, k) = \min_{\substack{0<|S|\leq k \\ S \subseteq L}} \frac{|N(S)|}{|S|}$$

Grafi Magici e Left Expansion



$(8, 6, 2)$
grafo magico

Definizione

Dato G bipartito, allora è detto un (n, m, d)-**grafo magico** se valgono le seguenti:

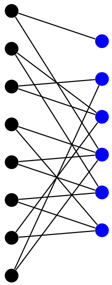
- $|L| = n, \quad |R| = m, \quad d$-regolare a sinistra
- $S \subseteq L, \quad |S| \leq n/10d \implies |N(S)| \geq 5d|S|/8$
- $S \subseteq L, \quad n/10d < |S| \leq n/2 \implies |N(S)| \geq |S|$

Definizione

Dato G bipartito, la sua **Left Vertex Expansion Ratio** è

$$L(G, k) = \min_{\substack{0<|S|\leq k \\ S\subseteq L}} \frac{|N(S)|}{|S|}$$

Grafi Magici e Left Expansion



$(8, 6, 2)$
grafo magico

Definizione

Dato G bipartito, allora è detto un (n, m, d)-**grafo magico** se valgono le seguenti:

- $|L| = n, \quad |R| = m, \quad d$-regolare a sinistra
- $S \subseteq L, \quad |S| \leq n/10d \implies |N(S)| \geq 5d|S|/8$
- $S \subseteq L, \quad n/10d < |S| \leq n/2 \implies |N(S)| \geq |S|$

Definizione

Dato G bipartito, la sua **Left Vertex Expansion Ratio** è

$$L(G, k) = \min_{\substack{0<|S|\leq k \\ S\subseteq L}} \frac{|N(S)|}{|S|}$$

Famiglia di Codici

Teorema

Preso un $(n, 3n/4, d)$-grafo magico, sia C il codice generato. Allora

$$D(C) > \frac{1}{10d} \qquad R(C) \geq \frac{1}{4}$$

- Solitamente, p è molto piccolo, dunque troviamo d tale che

$$\frac{1}{10d} \geq 2p$$

- $D(C)$ e $R(C)$ non dipendono da n, quindi possiamo trovare codici per messaggi di qualsiasi lunghezza

Costruzione Random

Dato un grafo bipartito con $|L| = n$, $|R| = m$, d-regolare a sinistra e costruito casualmente, è al 90% un (n, m, d)-grafo magico

Famiglia di Codici

Teorema

Preso un $(n, 3n/4, d)$-grafo magico, sia C il codice generato. Allora

$$D(C) > \frac{1}{10d} \qquad R(C) \geq \frac{1}{4}$$

- Solitamente, p è molto piccolo, dunque troviamo d tale che

$$\frac{1}{10d} \geq 2p$$

- $D(C)$ e $R(C)$ non dipendono da n, quindi possiamo trovare codici per messaggi di qualsiasi lunghezza

Costruzione Rapida

Dato un grafo bipartito con $|L| = n$, $|R| = m$, d-regolare a sinistra, e costruito casualmente, è al 90% un (n, m, d)-grafo magico

Famiglia di Codici

Teorema

Preso un $(n, 3n/4, d)$-grafo magico, sia C il codice generato. Allora

$$D(C) > \frac{1}{10d} \qquad R(C) \geq \frac{1}{4}$$

- Solitamente, p è molto piccolo, dunque troviamo d tale che
$$\frac{1}{10d} \geq 2p$$
- $D(C)$ e $R(C)$ non dipendono da n, quindi possiamo trovare codici per messaggi di qualsiasi lunghezza

Costruzione Rapida

Dato un grafo bipartito con $|L| = n$, $|R| = m$, d-regolare a sinistra, e costruito casualmente, è al 90% un (n, m, d)-grafo magico

Famiglia di Codici

Teorema

Preso un $(n, 3n/4, d)$-grafo magico, sia C il codice generato. Allora

$$D(C) > \frac{1}{10d} \qquad R(C) \geq \frac{1}{4}$$

- Solitamente, p è molto piccolo, dunque troviamo d tale che

$$\frac{1}{10d} \geq 2p$$

- $D(C)$ e $R(C)$ non dipendono da n, quindi possiamo trovare codici per messaggi di qualsiasi lunghezza

Costruzione Rapida

Dato un grafo bipartito con $|L| = n$, $|R| = m$, d-regolare a sinistra, e costruito casualmente, è al 90% un (n, m, d)-grafo magico

Famiglia di Codici

Teorema

Preso un $(n, 3n/4, d)$-grafo magico, sia C il codice generato. Allora

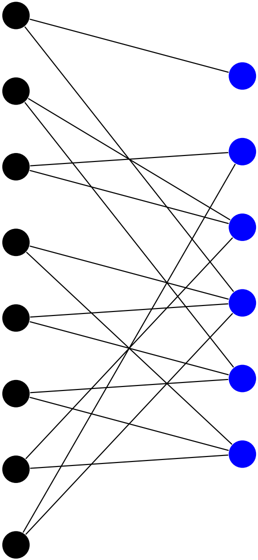
$$D(C) > \frac{1}{10d} \qquad R(C) \geq \frac{1}{4}$$

- Solitamente, p è molto piccolo, dunque troviamo d tale che
$$\frac{1}{10d} \geq 2p$$
- $D(C)$ e $R(C)$ non dipendono da n, quindi possiamo trovare codici per messaggi di qualsiasi lunghezza

Costruzione Rapida

Dato un grafo bipartito con $|L| = n$, $|R| = m$, d-regolare a sinistra, e costruito casualmente, è al 90% un (n, m, d)-grafo magico

Euristica



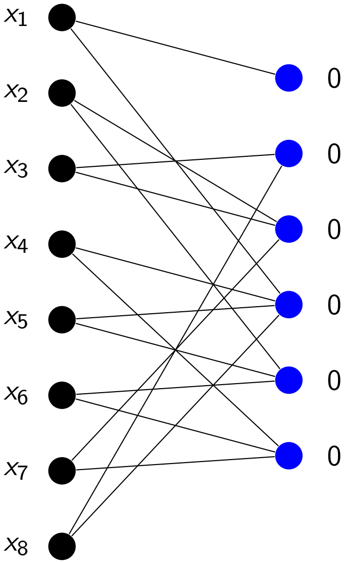
Problema: Come correggiamo le stringhe corrotte?

Algoritmo: cambiamo le componenti con *la maggior parte* dei controlli sbagliati

Finitezza: Il numero di controlli sbagliati decresce sempre

Correttezza?: Riusciamo a ricostruire la stringa originale del codice?

Euristica



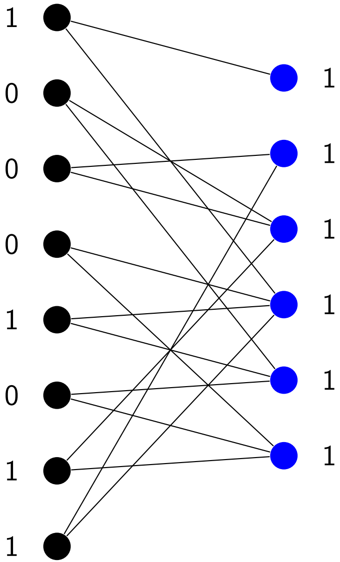
Problema Come correggiamo le stringhe corrotte?

Algoritmo cambiamo le componenti con *la maggior parte* dei controlli sbagliati

Finitezza Il numero di controlli sbagliati decresce sempre

Correttezza? Riusciamo a ricostruire la stringa originale del codice?

Euristica



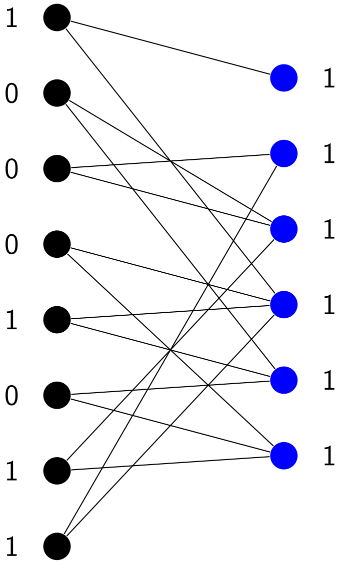
Problema Come correggiamo le stringhe corrotte?

Algoritmo cambiamo le componenti con *la maggior parte* dei controlli sbagliati

Finitezza Il numero di controlli sbagliati decresce sempre

Correttezza? Riusciamo a ricostruire la stringa originale del codice?

Euristica



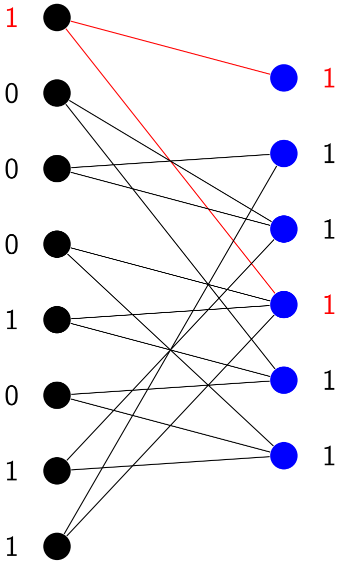
Problema Come correggiamo le stringhe corrotte?

Algoritmo cambiamo le componenti con *la maggior parte* dei controlli sbagliati

Finitezza Il numero di controlli sbagliati decresce sempre

Correttezza? Riusciamo a ricostruire la stringa originale del codice?

Euristica



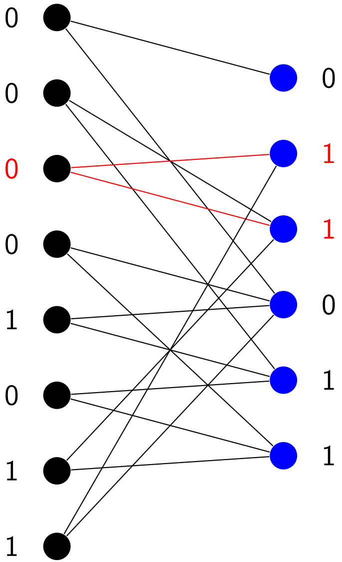
Problema Come correggiamo le stringhe corrotte?

Algoritmo cambiamo le componenti con *la maggior parte* dei controlli sbagliati

Finitezza Il numero di controlli sbagliati decresce sempre

Correttezza? Riusciamo a ricostruire la stringa originale del codice?

Euristica



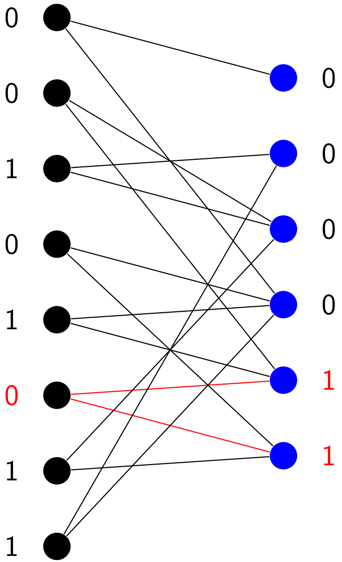
Problema Come correggiamo le stringhe corrotte?

Algoritmo cambiamo le componenti con *la maggior parte* dei controlli sbagliati

Finitezza Il numero di controlli sbagliati decresce sempre

Correttezza? Riusciamo a ricostruire la stringa originale del codice?

Euristica



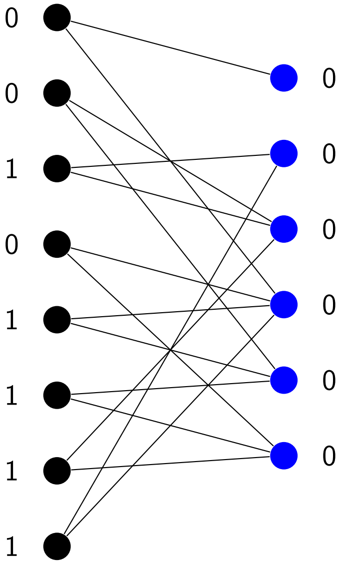
Problema Come correggiamo le stringhe corrotte?

Algoritmo cambiamo le componenti con *la maggior parte* dei controlli sbagliati

Finitezza Il numero di controlli sbagliati decresce sempre

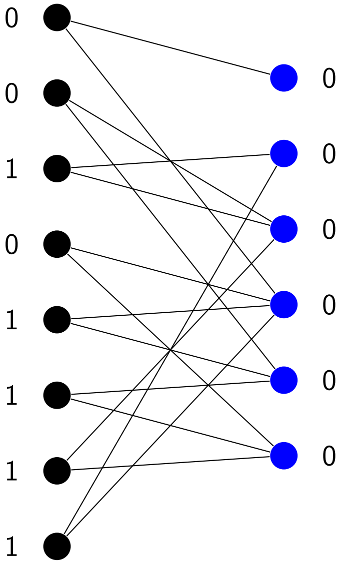
Correttezza? Riusciamo a ricostruire la stringa originale del codice?

Euristica



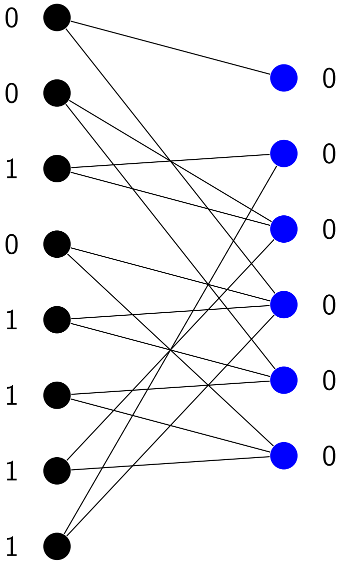
Problema	Come correggiamo le stringhe corrotte?
Algoritmo	cambiamo le componenti con *la maggior parte* dei controlli sbagliati
Finitezza	Il numero di controlli sbagliati decresce sempre
Correttezza?	Riusciamo a ricostruire la stringa originale del codice?

Euristica



Problema	Come correggiamo le stringhe corrotte?
Algoritmo	cambiamo le componenti con *la maggior parte* dei controlli sbagliati
Finitezza	Il numero di controlli sbagliati decresce sempre
Correttezza?	Riusciamo a ricostruire la stringa originale del codice?

Euristica



Problema Come correggiamo le stringhe corrotte?

Algoritmo cambiamo le componenti con *la maggior parte* dei controlli sbagliati

Finitezza Il numero di controlli sbagliati decresce sempre

Correttezza? Riusciamo a ricostruire la stringa originale del codice?

Belief Propagation

Teorema

Dato G un grafo bipartito d-regolare a sinistra, supponiamo che la percentuale di corruzione del messaggio sia al massimo p.
Se $k = \lfloor pn \rfloor$ e $L(G, 2k) > 3d/4$, allora il messaggio originale viene ripristinato in $O(n)$ ripetizioni dell'euristica.

Nota Bene

- $3d/4 > 5d/8$, dunque un grafo che soddisfi $L(G, 2k) > 3d/4$ è anche un grafo magico, *ma non è vero il viceversa*.
- Nel 2002 sono stati costruiti per la prima volta dei grafi con $L(G, 2k) > (1 - \delta)d$ per ogni $\delta > 0$
- La *Belief Propagation* ha una complessità totale di $O(n)$

Belief Propagation

Teorema

Dato G un grafo bipartito d-regolare a sinistra, supponiamo che la percentuale di corruzione del messaggio sia al massimo p.
Se $k = \lfloor pn \rfloor$ e $L(G, 2k) > 3d/4$, allora il messaggio originale viene ripristinato in $O(n)$ ripetizioni dell'euristica.

Nota Bene

- $3d/4 > 5d/8$, dunque un grafo che soddisfi $L(G, 2k) > 3d/4$ è anche un grafo magico, *ma non è vero il viceversa*
- Nel 2002 sono stati costruiti per la prima volta dei grafi con $L(G, 2k) > (1 - \delta)d$ per ogni $\delta > 0$
- La *Belief Propagation* ha una complessità totale di $O(n)$

Belief Propagation

Teorema

Dato G un grafo bipartito d-regolare a sinistra, supponiamo che la percentuale di corruzione del messaggio sia al massimo p.
Se $k = \lfloor pn \rfloor$ e $L(G, 2k) > 3d/4$, allora il messaggio originale viene ripristinato in $O(n)$ ripetizioni dell'euristica.

Nota Bene

- $3d/4 > 5d/8$, dunque un grafo che soddisfi $L(G, 2k) > 3d/4$ è anche un grafo magico, *ma non è vero il viceversa*
- Nel 2002 sono stati costruiti per la prima volta dei grafi con $L(G, 2k) > (1 - \delta)d$ per ogni $\delta > 0$
- La *Belief Propagation* ha una complessità totale di $O(n)$

Belief Propagation

Teorema

Dato G un grafo bipartito d-regolare a sinistra, supponiamo che la percentuale di corruzione del messaggio sia al massimo p.
Se $k = \lfloor pn \rfloor$ e $L(G, 2k) > 3d/4$, allora il messaggio originale viene ripristinato in $O(n)$ ripetizioni dell'euristica.

Nota Bene

- $3d/4 > 5d/8$, dunque un grafo che soddisfi $L(G, 2k) > 3d/4$ è anche un grafo magico, *ma non è vero il viceversa*
- Nel 2002 sono stati costruiti per la prima volta dei grafi con $L(G, 2k) > (1 - \delta)d$ per ogni $\delta > 0$
- La *Belief Propagation* ha una complessità totale di $O(n)$

Belief Propagation

Teorema

Dato G un grafo bipartito d-regolare a sinistra, supponiamo che la percentuale di corruzione del messaggio sia al massimo p.
Se $k = \lfloor pn \rfloor$ e $L(G, 2k) > 3d/4$, allora il messaggio originale viene ripristinato in $O(n)$ ripetizioni dell'euristica.

Nota Bene

- $3d/4 > 5d/8$, dunque un grafo che soddisfi $L(G, 2k) > 3d/4$ è anche un grafo magico, *ma non è vero il viceversa*
- Nel 2002 sono stati costruiti per la prima volta dei grafi con $L(G, 2k) > (1 - \delta)d$ per ogni $\delta > 0$
- La *Belief Propagation* ha una complessità totale di $O(n)$

Belief Propagation

Teorema

Dato G un grafo bipartito d-regolare a sinistra, supponiamo che la percentuale di corruzione del messaggio sia al massimo p.
Se $k = \lfloor pn \rfloor$ e $L(G, 2k) > 3d/4$, allora il messaggio originale viene ripristinato in $O(n)$ ripetizioni dell'euristica.

Nota Bene

- $3d/4 > 5d/8$, dunque un grafo che soddisfi $L(G, 2k) > 3d/4$ è anche un grafo magico, *ma non è vero il viceversa*
- Nel 2002 sono stati costruiti per la prima volta dei grafi con $L(G, 2k) > (1 - \delta)d$ per ogni $\delta > 0$
- La *Belief Propagation* ha una complessità totale di $O(n)$

Altri Utilizzi

I Grafi Expanders vengono usati in molti contesti, quali

- Convergenza veloce di Catene di Markov e derandomizzazione di algoritmi
- Studio di gruppi tramite Grafi di Caley
- Applicazioni alla teoria della complessità e approssimazioni di algoritmi Np-hard o completi
- Studio di ricoprimenti universali di grafi e alberi infiniti
- Costruzione di superconcentratori e complessità relativa a matrici super regolari
- Applicazioni nella teoria degli embedding in spazi metrici, e utilizzi in algoritmi per problemi di taglio

Altri Utilizzi

I Grafi Expanders vengono usati in molti contesti, quali

- Convergenza veloce di Catene di Markov e derandomizzazione di algoritmi
- Studio di gruppi tramite Grafi di Caley
- Applicazioni alla teoria della complessità e approssimazioni di algoritmi Np-hard o completi
- Studio di ricoprimenti universali di grafi e alberi infiniti
- Costruzione di superconcentratori e complessità relativa a matrici super regolari
- Applicazioni nella teoria degli embedding in spazi metrici, e utilizzi in algoritmi per problemi di taglio

Altri Utilizzi

I Grafi Expanders vengono usati in molti contesti, quali

- Convergenza veloce di Catene di Markov e derandomizzazione di algoritmi
- Studio di gruppi tramite Grafi di Caley
- Applicazioni alla teoria della complessità e approssimazioni di algoritmi Np-hard o completi
- Studio di ricoprimenti universali di grafi e alberi infiniti
- Costruzione di superconcentratori e complessità relativa a matrici super regolari
- Applicazioni nella teoria degli embedding in spazi metrici, e utilizzi in algoritmi per problemi di taglio

Altri Utilizzi

I Grafi Expanders vengono usati in molti contesti, quali

- Convergenza veloce di Catene di Markov e derandomizzazione di algoritmi
- Studio di gruppi tramite Grafi di Caley
- Applicazioni alla teoria della complessità e approssimazioni di algoritmi Np-hard o completi
- Studio di ricoprimenti universali di grafi e alberi infiniti
- Costruzione di superconcentratori e complessità relativa a matrici super regolari
- Applicazioni nella teoria degli embedding in spazi metrici, e utilizzi in algoritmi per problemi di taglio

Altri Utilizzi

I Grafi Expanders vengono usati in molti contesti, quali

- Convergenza veloce di Catene di Markov e derandomizzazione di algoritmi
- Studio di gruppi tramite Grafi di Caley
- Applicazioni alla teoria della complessità e approssimazioni di algoritmi Np-hard o completi
- Studio di ricoprimenti universali di grafi e alberi infiniti
- Costruzione di superconcentratori e complessità relativa a matrici super regolari
- Applicazioni nella teoria degli embedding in spazi metrici, e utilizzi in algoritmi per problemi di taglio

Altri Utilizzi

I Grafi Expanders vengono usati in molti contesti, quali

- Convergenza veloce di Catene di Markov e derandomizzazione di algoritmi
- Studio di gruppi tramite Grafi di Caley
- Applicazioni alla teoria della complessità e approssimazioni di algoritmi Np-hard o completi
- Studio di ricoprimenti universali di grafi e alberi infiniti
- Costruzione di superconcentratori e complessità relativa a matrici super regolari
- Applicazioni nella teoria degli embedding in spazi metrici, e utilizzi in algoritmi per problemi di taglio

Altri Utilizzi

I Grafi Expanders vengono usati in molti contesti, quali

- Convergenza veloce di Catene di Markov e derandomizzazione di algoritmi
- Studio di gruppi tramite Grafi di Caley
- Applicazioni alla teoria della complessità e approssimazioni di algoritmi Np-hard o completi
- Studio di ricoprimenti universali di grafi e alberi infiniti
- Costruzione di superconcentratori e complessità relativa a matrici super regolari
- Applicazioni nella teoria degli embedding in spazi metrici, e utilizzi in algoritmi per problemi di taglio

Altri Utilizzi

I Grafi Expanders vengono usati in molti contesti, quali

- Convergenza veloce di Catene di Markov e derandomizzazione di algoritmi
- Studio di gruppi tramite Grafi di Caley
- Applicazioni alla teoria della complessità e approssimazioni di algoritmi Np-hard o completi
- Studio di ricoprimenti universali di grafi e alberi infiniti
- Costruzione di superconcentratori e complessità relativa a matrici super regolari
- Applicazioni nella teoria degli embedding in spazi metrici, e utilizzi in algoritmi per problemi di taglio