Riordinare una matrice sparsa simmetrica per ridurne la taglia dell'inviluppo

Gemma Martini

8/04/2014

Indice

1 Prefazione

Prima di entrare nel vivo della trattazione è necessario soffermarsi sullo scopo dell'algoritmo che spiegheremo. Barnard, Pothen e Simon[**?**] forniscono un algoritmo interessante per la riduzione della taglia dell'inviluppo di una matrice sparsa simmetrica. Per chi non sapesse cos'è l'inviluppo di una matrice, ci limitiamo a fornire una definizione poco formale, ma immaginifica: l'insieme delle porzioni di riga precedute da soli zeri.

Perchè ridurre l'inviluppo di una matrice? I metodi frontali collegati alla conformazione dell'inviluppo sono metodi utilizzati in molti algoritmi (per esempio CSM della NASA oppure MSC/NASTRAN o ANSYS) per risolvere sistemi lineari su larga scala. In particolare questo algoritmo rappresenta un miglioramento in performance senza però effettuare cambiamenti sostanziali alle strutture dati.

2 Introduzione

Per effettuare calcoli con matrici sparse di taglia grande è molto importante riordinare tali matrici. Ci sono alcuni algoritmi, come l'"algoritmo del grado minimo" ed il "nested dissection", che permutano la matrice in modo tale da ridurre il numero di cifre non zero

della matrice L della fattorizzazione di Cholesky[1], così da rendere il calcolo notevolmente più veloce.

Gli algoritmi Gibbs-Poole-Stockmeyer (GPS), Cuthill-McKee (RCM) e Gibbs-King(GK) effettuano una ricerca locale nel grafo di adiacenza della matrice: tutti cercano di trovare uno pseudodiametro (non un diametro, ma qualcosa di lungo) nel grafo facendo una BFS e analizzandone i livelli di ricorsione, con lo scopo di ridurre la taglia dell'inviluppo.

Altri algoritmi, come RCM, GPS e GK, hanno lo scopo di ridurre l'inviluppo di una matrice sparsa, tuttavia si è visto che l'ordinamento RCM può essere usato per calcolare fattorizzazioni incomplete da usare per il precondizionamento di sistemi lineari. Inoltre tale algoritmo è implementato nella libreria di Octave, perciò sarà utilizzato come metro di paragone nel corso della nostra sperimentazione.

L'algoritmo proposto da Barnard, Pothen e Simon, che da ora in poi chiameremo **spectral**, prende in considerazione vettori ed è per natura non ricorsivo, si presta quindi ad effettuare operazioni in parallelo con un minimo sofrzo. Inoltre spectral si propone di lavorare su due fronti: ridurre l'inviluppo della matrice e, velocizzare la fattorizzazione di Cholesky.

3 Spectral

3.1 Definizioni

Definizione 3.1 (Definizioni utili per definire l'inviluppo)
Sia A una matrice simmetrica $n \times n$ con diagonale non nulla, di elementi a_{ij}.

- Denotiamo gli indici delle colonne degli elementi non nulli nella parte triangolare inferiore della i-esima riga con $\mathbf{row(i)} = \{j : a_{ij} \neq 0, 1 \leq j \leq i\}$.
- Per la i-esima riga di A definiamo $\mathbf{f_i(A)} = \min\{j : j \in row(i)\}$, che rappresenta l'indice della prima cifra non zero nella i-esima riga di A.
- Sempre per la i-esima riga di A definiamo $\mathbf{r_i(A)} = i - f_i(A)$ è chiamata **row-width** della i-esima riga di A.
- La **band-width** di A è il massimo delle row-width: $\mathbf{bw(A)} = \max\{r_i(A) : i = 1, ..., n\}$.

Definizione 3.2 (Inviluppo di una matrice)
Sia A una matrice simmetrica $n \times n$ con diagonale non nulla, di elementi a_{ij}. L'**inviluppo** di A è l'insieme degli indici delle colonne che stanno fra l'indice della colonna del primo non-zero e la diagonale in ogni riga: $\mathbf{Env(A)} = \{(i, j) : f_i(A) \leq j \leq i, i = 1, ...n\}$. Denotiamo **taglia dell'inviluppo Esize**$\mathbf{(A)} = |Env(A)|$.

[1]La fattorizzazione di Cholesky è la decomposizione di una matrice hermitiana (matrice quadrata complessa uguale alla sua trasposta coniugata), definita positiva nel prodotto di una matrice triangolare inferiore e la sua trasposta coniugata, utile per simulazioni Monte Carlo (metodo usato per trarre stime attraverso simulazioni: di base viene utilizzato per trovare soluzioni a problemi matematici che hanno molte variabili e che non possono essere risolti facilmente). Questa decomposizione è doppiamente efficiente rispetto alla decomposizione LU per la risoluzione di sistemi lineari.

Definizione 3.3
Sia A una matrice simmetrica $n \times n$ con diagonale non nulla, di elementi a_{ij}.

- Nel caso in cui A sia definita positiva è possibile effettuare la **fattorizzazione di Cholesky** di A, per la quale denoteremo una limitazione dall'alto del costo computazionale: $\mathbf{Ework}(\mathbf{A}) = \sum_{i=1}^{n} r_i^2$.
- Definiamo inoltre $\mathbf{Esize_{min}}(\mathbf{A})$ la minima taglia dell'inviluppo di A al variare delle permutazioni operate sulla matrice.
- Nello stesso spirito definiamo $\mathbf{Ework_{min}}(\mathbf{A})$ e $\mathbf{bw_{min}}(\mathbf{A})$.

I valori di tutti i parametri definiti dipendono strettamente dalla permutazione della matrice A (P^tAP).

Definizione 3.4 (Gli stessi parametri di prima nel caso di un grafo)
Sia A una matrice simmetrica $n \times n$ con diagonale non nulla, di elementi a_{ij} e sia $G(V, E)$ il suo grafo di adiacenza. Chiamando $\boldsymbol{\alpha}$ un ordinamento dei nodi del grafo:

- Denotiamo **vicini** di un nodo v $\mathbf{nbr}(\mathbf{v}) = \{v\} \cup adj\{v\}$.
- Al posto di r_i abbiamo $\mathbf{r}(\mathbf{v}, \boldsymbol{\alpha}) = \max\{\alpha(v) - \alpha(w) : w \in nbr(v),\ \alpha(w) \leq \alpha(v)\}$.
- $\mathbf{Esize}(\mathbf{G}, \boldsymbol{\alpha}) = \sum_{v \in V} r(v) = \sum_{v \in V} \max\{\alpha(v) - \alpha(w) : w \in nbr(v), \alpha(w) \leq \alpha(v)\}$.
- $\mathbf{Ework}(\mathbf{G}, \boldsymbol{\alpha}) = \sum_{v \in V} r(v)^2 = \sum_{v \in V} \max\{(\alpha(v) - \alpha(w))^2 : w \in nbr(v),\ \alpha(w) \leq \alpha(v)\}$.

L'obbiettivo, nel caso del grafo, è di scegliere una permutazione α che minimizza uno dei valori definiti; s'intende che i minimi di funzioni diverse non si verificheranno con lo stesso α.

Riscriviamo Envelope size ed Envelope work:

$$Esize(A) = |Env(A)| = \sum_{i=1}^{n} r_i = \sum_{i=1}^{n} (i - \min row(i)) = \sum_{i=1}^{n} \max_{j \in row(i)} (i - j)$$

$$Ework(A) = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (i - \min row(i))^2 = \sum_{i=1}^{n} \max_{j \in row(i)} (i - j)^2$$

Definizione 3.5 (1-somma e 2-somma)
Sia A una matrice simmetrica $n \times n$ con diagonale non nulla, di elementi a_{ij}.

- La **1-somma** è definita come $\boldsymbol{\sigma_1}(\mathbf{A}) = \sum_{i=1}^{n} \sum_{j \in row(i)} (i - j)$
- La **2-somma** è definita come $\boldsymbol{\sigma_2^2}(\mathbf{A}) = \sum_{i=1}^{n} \sum_{j \in row(i)} (i - j)^2$

Come detto in precedenza, $\boldsymbol{\sigma_{1,\mathbf{min}}}(\mathbf{A})$ e $\boldsymbol{\sigma^2_{2,\mathbf{min}}}(\mathbf{A})$ sono i valori minimi di questi parametri su tutte le permutazioni della matrice A.

Notoriamente minimizzare la bandwidth e la 1-somma sono problemi NP-completi, quindi vengno trattati dall'euristica. È stato recentemente dimostrato che il problema della taglia dell'inviluppo è strettamente legato al problema della 1-somma ed il problema del costo computazionale della fattorizzazione di Cholesky è collegato al problema della 2-somma.

3.2 La matrice Laplaciana ed alcune limitazioni sui parametri

Definizione 3.6
Sia A una matrice simmetrica $n \times n$ con diagonale non nulla, di elementi a_{ij}. La **matrice Laplaciana Q** di A è la differenza tra la matrice diagonale dei gradi e la matrice di adiacenza del grafo associato alla matrice. $Q = D - B$. Se chiamiamo q_{ij} gli elementi della Laplaciana si ha:

$$q_{ij} = \begin{cases} -1 & \text{se } i \neq j, m_{ij} \neq 0 \\ 0 & \text{se } i \neq j, m_{ij} = 0 \\ -\sum\limits_{\substack{j=1 \\ j \neq i}}^{n} q_{ij} & \text{se } i = j \end{cases}$$

Siano $\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, ..., \boldsymbol{\lambda_n}$ gli autovalori di Q tali che $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. Sia x_k un autvettore relativo al k-esimo autovalore, lo chiameremo **k-esimo autovettore di Q**.

Teorema 3.1
Sia A una matrice simmetrica $n \times n$ e sia Q la sua Laplaciana.

- *Q è una matrice singolare con autovalori non negativi.*
- *Q è irriducibile sse $\lambda_2 > 0$. Inoltre, per come è fatta la matrice, gli autvettori relativi all'autovalore 0 sono tutti i multipli di* $(1,\ 1,\ 1,\ \ldots,\ 1)$.

Dimostrazione. Ricordando la definizione di matrice riducibile: M si dice **riducibile** se esiste una matrice di cambiamento di base B tale che $B^{-1}AB$ è una matrice triangolare a blocchi: $B^{-1}AB = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$.
M si dice **irriducibile** se è non riducibile.

- Nella matrice Q la somma degli elementi su ogni riga è nulla, quindi Q è dominante diagonale e, per il primo teorema di Gerschgorin, gli autovalori sono ≥ 0.
- $\Leftarrow$) Hp: Q è riducibile. Essendo simmetrica è simile ad una matrice $Q' = \begin{pmatrix} L_1 & 0 \\ 0 & L_2 \end{pmatrix}$, con L_1 di taglia n_1 e L_2 di taglia n_2. Per come è definito il prodotto matrice-vettore $v_1 = (\underbrace{1,\ 1,\ \ldots,\ 1}_{n_1}, \underbrace{0,\ 0,\ \ldots,\ 0}_{n_2})^t$ e $v_2 = (\underbrace{0,\ 0,\ \ldots,\ 0}_{n_1}, \underbrace{1,\ 1,\ \ldots,\ 1}_{n_2})^t$ sono autovettori linearmente indipendenti entrambi relativi all'autovalore 0. Quindi $\lambda_1 = \lambda_2 = 0$.

  $\Rightarrow$) Hp: Q è irriducibile. Sia v un vettore tale che $Lv = 0$, allora $0 = v^tLv \overset{*}{=} \sum\limits_{(i,j) \in E} (v_i - v_j)^2$. Dove il passaggio contrassegnato con * è giustificato dalla forma della matrice Q.

$(i,j) \in E \Rightarrow v_i = v_j \Rightarrow v_i = v_1 \forall i \Rightarrow v = c(1,\ 1,\ 1,\ \ldots,\ 1)^t$.

□

Teorema 3.2 (Teorema di George e Pothen 1 [?])
Sia A una matrice simmetrica $n \times n$. Sia Δ il massimo numero di elementi non-nulli fuori dalla diagonale in una riga di A. Valgono le seguenti disuguaglianze:

- $Esize_{min}(A) \leq \sigma_{1,min}(A) \leq \Delta Esize_{min}(A)$.
- $Ework_{min}(A) \leq \sigma^2_{2,min}(A) \leq \Delta Ework_{min}(A)$.
- $\sigma_{2,min}(A) \leq \sigma_{1,min}(A) \leq \sqrt{(|E|)}\sigma_{2,min}(A)$.

Dimostrazione. Ricordando che $\Delta = \max_{\text{sulle righe}}$ (#di elementi non nulli fuori dalla diagonale) vediamo le dimostrazioni dei tre punti:

- È sufficiente scrivere le definizioni di $Esize(A) = \sum_{i=1}^{n} \max_{j \in row(i)} (i-j)$ e $\sigma_1(A) = \sum_{i=1}^{n} \sum_{j \in row(i)} (i-j)$, poichè se una disuguaglianza vale in generale vale anche per il minimo sulle permutazioni. Infatti $\max_{j \in row(i)} (i-j) \leq \sum_{j \in row(i)} (i-j)$ e $\sum_{j \in row(i)} (i-j) \leq \Delta \max_{j \in row(i)} (i-j)$.

- È sufficiente scrivere le definizioni di $Ework(A) = \sum_{i=1}^{n} \max_{j \in row(i)} (i-j)^2$ e $\sigma_2^2(A) = \sum_{i=1}^{n} \sum_{j \in row(i)} (i-j)^2$, poichè se una disuguaglianza vale in generale vale anche per il minimo sulle permutazioni. La disuguaglianza segue da quanto detto nel punto precendente opportunamente trasferito a $Ework_{min}(A)$, $\sigma^2_{2,min}(A)$ e $\Delta Ework_{min}(A)$.

- Siano E =insieme degli archi ed $|E|$ = #archi. Scrivendo σ_1 e σ_2 nel seguente modo $\sigma_1 = \sum_{(i,j) \in E} (i-j)$, $\sigma_2 = \sqrt{\sum_{(i,j) \in E} (i-j)^2}$ la prima disuguaglianza è facilmente provata, perchè $\sigma_2^2 \leq (\sigma_1)^2$. Per quanto riguarda la seconda ricordiamo la disuguaglianza tra media aritmetica e media quadratica: $\frac{\sum_{i=1}^{n} a_i}{n} \leq \sqrt{\frac{\sum_{i=1}^{n} a_i^2}{n}}$. Notiamo che $n = |E|$ e si ha la tesi.

□

Nonostante possa sembrare che parlare di autovalori della Laplaciana di una matrice sparsa non serva a nulla ai fini della riduzione della taglia dell'inviluppo, Miroslav Fiedler[**?**] [**?**] [**?**] (matematico Ceco famoso per il suo contributo in algebra lineare, teoria dei grafi e e teoria algebrica dei grafi) è stato pioniere nel collegare questi due concetti. Successivamente il suo lavoro è stato ripreso da Juvan e Mohar[**?**], che hanno suggerito l'uso del secondo autovettore laplaciano per riordinare matrici in modo da minimizzare la bandwidth, la 1-somma e la 2-somma.

Teorema 3.3 (Teorema di George e Pothen 2 [?])
Sia A una matrice simmetrica $n \times n$. Sia Δ il massimo numero di elementi non-nulli fuori dalla diagonale in una riga di A. Valgono le seguenti limitazioni per $Esize_{min}(A)$:

$$\frac{\lambda_2(Q)}{6\Delta}(n^2-1) \leq Esize_{min}(A) \leq \frac{\lambda_n(Q)}{6}(n^2-1)$$

Prima di addentrarci nella dimostrazione è necessario premettere alcune definizioni ed un lemma [**?**].

Definizione 3.7
Sia A una matrice simmetrica $n \times n$ con diagonale non nulla, di elementi a_{ij} e sia G(V,E) il suo grafo di adiacenza. Sia $S \subseteq V = \{v_1, v_2, \ldots, v_n\}$ e sia $V_j = \{v_1, v_2, \ldots, v_j\}$.

- Denotiamo con $\mathbf{adj(S)} = \{v \in V \setminus S : \exists u \in S \ t.c. (u,v) \in E\}$ l'insieme dei vertici che non stanno in S adiacenti ad almeno un nodo di S.
- Denotiamo con $\boldsymbol{\delta}\mathbf{(S)} = \{(u,v) \in E : u \in S, v \in V \setminus S\}$ l'insieme degli archi tra nodi di S nodi fuori da S.
- Chiamiamo **j-esima frontwidth** la quantità $|adj(V_j)|$, che corrisponde alla taglia della j-esima colonna dell'inviluppo di A.

Lemma 3.4
Nelle ipotesi della definizione **??** *vale la seguente catena di disuguaglianze:*

$$\lambda_2(Q)\frac{|S||V \setminus S|}{n} \overset{(1)}{\leq} |\delta(S)| \overset{(2)}{\leq} \lambda_n(Q)\frac{|S||V \setminus S|}{n}$$

Dimostrazione teorema **??**. Vale la seguente uguaglianza

$$Esize(A) = \sum_{j=1}^{n} |adj(V_j)|$$

infatti, per $k > j$, $v \in adj(V_j) \Leftrightarrow (k,i) \in E \Leftrightarrow a_{k,i} \neq 0, i \leq j$ (da notare che per $k \leq j$ l'elemento $a_{k,i}$ sta nella parte triangolare superiore, dunque non appartiene, per definizione, all'inviluppo).
Per dimostrare la prima disuguaglianza

$$
\begin{aligned}
Esize(A) &= \sum_{j=1}^{n} |adj(V_j)| \\
&\overset{*}{\geq} \sum_{j=1}^{n} \frac{|\delta(V_j)|}{\Delta} \\
&\overset{(1)}{\geq} \frac{\lambda_2(Q)}{n\Delta} \sum_{j=1}^{n} |V_j||V \setminus V_j| \\
&= \frac{\lambda_2(Q)}{n\Delta} \sum_{j=1}^{n} nj - j^2 \\
&= \frac{\lambda_2(Q)}{n\Delta} \left(n\frac{n(n+1)}{2} - \frac{n(n+1)(2n+1)}{6} \right) \\
&= \frac{\lambda_2(Q)}{6\Delta} \left(3n^2 + 3n - 2n^2 - 3n - 1 \right) \\
&= \frac{\lambda_2(Q)}{6}(n^2 - 1)
\end{aligned}
$$

dove il passaggio contrassegnato con * vale poichè $|adj(V_j)| \geq |\delta(V_j)|$.
Per quanto riguarda la seconda

$$
\begin{aligned}
Esize(A) &= \sum_{j=1}^{n} |adj(V_j)| \\
&\overset{**}{\geq} \sum_{j=1}^{n} |\delta(V_j)| \\
&\overset{(2)}{\geq} \frac{\lambda_n(Q)}{n} \sum_{j=1}^{n} |V_j||V \setminus V_j| \\
&= \frac{\lambda_n(Q)}{6}(n^2 - 1)
\end{aligned}
$$

dove il passaggio contrassegnato da ** vale perchè gli archi tra V_j e gli altri nodi sono di più dei nodi che stanno fuori. □

3.3 Euristica della riduzione dell'inviluppo

È questa la parte in cui vengono offerte giustificazioni formali della correttezza di spectral.
1) Far vedere che il secondo autvettore laplaciano x_2 risolve un rilassamento continuo del problema della 2-somma (NP-completo).
2) Far vedere che il vettore di permutazione calcolato da spectral è un vettore vicino a x_2(nel senso di $\|\cdot\|_2$).

1)

Per n dispari $\mathscr{P}$ è l'insieme dei vettori p di n componenti le cui componenti sono permutazioni dell'insieme $\{\frac{-(n-1)}{2}, \ldots, -1, 0, 1, \ldots \frac{(n-1)}{2}\}$. Inoltre sia $u = (1, 1, \ldots, 1)^t$ $\forall p \in \mathscr{P}$ $\ell = p^t p = (\frac{n}{12})(n+1)(n+2)$.

Per n pari $\mathscr{P}$ è l'insieme vettori p di n componenti le cui componenti sono permutazioni dell'insieme $\{\frac{-n}{2}, \ldots, -1, 0, 1, \ldots \frac{n}{2}\}$. Inoltre sia $u = (1, 1, \ldots, 1)^t$ $\forall p \in \mathscr{P}$ $\ell = p^t p = (\frac{n}{12})(n^2-1)$.

Chiamando x_i la i-esima componente di un vettore x, consideriamo il minimo della 2-somma di una matrice A simmetrica:

$$\begin{aligned}
\sigma^2_{2,min}(A) &= \min_{PAP^t} \sum_{i=1}^{n} \sum_{j \in row(i)} (i-j)^2 \\
&= \min_{x \in \mathscr{P}} \sum_{i=1}^{n} \sum_{j \in row(i)} (x_i - x_j)^2 \\
&= \frac{1}{2} \min_{x \in \mathscr{P}} \sum_{a_{ij} \neq 0} (x_i - x_j)^2
\end{aligned}$$

Questa uguaglianza vale solo nel caso pari(????), ma per n molto grande non è un problema. A questo punto l'idea è di rilassare il problema discreto in uno continuo per poi approssimare in un secondo momento. Dato un vettore $x \in \mathbb{R}^n$ possiamo definire un vettore di permutazione p indotto da s secondo la regola $p_i \leq p_j \Leftrightarrow x_i \leq x_j$, la permutazione è unica se non ci sono componenti uguali in x. Sia $\mathcal{X} = \{x \in t.c.\ x \neq 0, x^t u = 0, x^t x = \ell\}$ il problema di ottimizzazione continua è

$$\begin{aligned}
\frac{1}{2} \min_{x \in \mathcal{X}} \sum_{a_{ij} \neq 0} (x_i - x_j)^2 &= \min_{x \in \mathcal{X}} \Big(\sum_{i=1}^{n} d_i x_i^2 - 2 \sum_{\substack{j<i \\ a_{ij} \neq 0}} x_i x_j \Big) \\
&= \min_{x \in \mathcal{X}} \Big(x^t D x - x^t B x \Big) \\
&= \min_{x \in \mathcal{X}} x^t Q x \\
&= \lambda_2 x_2^t x_2 = \lambda_2 \ell
\end{aligned}$$

2) Resta da provare che il vettore di permutazione p_m indotto dal secondo autovettore laplaciano x_2 è un vettore vicino, nel senso della norma 2, a x_2 in $\mathscr{P}$.

Teorema 3.5
Sia A una matrice simmetrica sparsa $n \times n$. p_m il vettore indotto dal secondo autovettore laplaciano x_2 è un vettore permutazione vicino, nel senso della norma 2, a x_2. In altre parole: $p_m = arg \min_{p \in \mathscr{P}} \|p - x_2\|_2$

Prima di addentrarci nella dimostrazione è necessario premettere un lemma [**?**].

Lemma 3.6
Siano $a_1, a_2, b_1, b_2 \in \mathbb{R}$ t.c. $a_1 < a_2$, $b_1 < b_2$, $r = (a_1 - b_1)^2$ e $s = (a_1 - b_1)^2 + (a_2 - b_2)^2$. Allora $r > s$.

Dimostrazione.

$$\begin{aligned} r - s &= a_1^2 + b_2^2 - 2a_1b_2 + a_2^2 + b_1^2 - 2a_2b_1 \\ &\quad - a_1^2 - b_1^2 + 2a_1b_1 - a_2^2 - b_2^2 + 2a_2b_2 \\ &= 2a_1(b_1 - b_2) - 2_a(b_1 - b_2) \\ &= 2(a_1 - a_2)(b_1 - b_2) > 0 \end{aligned}$$

□

Dimostrazione teorema **??**. Sia $y \neq p_m$ un vettore di permutazione tale che $\exists u, v \in V$ *t.c.* $x_2(u) < x_2(v)$ e $y(u) > y(v)$. Sia z il vettore di permutazione tale che $z(u) = y(v)$, $z(v) = y(u)$ e $z(w) = y(w)$ $\forall w \neq v \neq u \in V$. Allora $\|y - x_2\|_2^2 - \|z - x_2\|_2^2 = (y(u) - x_2(u))^2 + (y(v) - x_2(v))^2 - (y(v) - x_2(u))^2 - (y(u) - x_2(v))^2 \overset{lemma}{>} 0$. Scambiando le componenti, abbiamo ottenuto che z è un vettore più vicino di y ad x_2. La tesi si ha dal fatto che scambiando tutte le componenti si ottiene p_m che coincide con il vettore più vicino ad x_2. □

3.4 Permutazioni di adiacenza???

Definizione 3.8
Sia A una matrice simmetrica $n \times n$ con diagonale non nulla, di elementi a_{ij} e sia G(V,E) il suo grafo di adiacenza. Sia $S \subseteq V = \{v_1, v_2, \ldots, v_n\}$ e sia $V_j = \{v_1, v_2, \ldots, v_j\}$. Diciamo che una permutazione è una **permutazione di adiacenza** se $v_{j+1} \in adj(V_j), j = 1, 2, \ldots, n$. Si noti che avere una permutazione di adiacenza è equivalente a dire che il grafo indotto da V_j è connesso.

Da quanto appena detto, richiamando la definizione **??**, risulta motivata la scelta di considerare ordini di adiacenza per ridurre la taglia dell'inviluppo: l'idea è infatti di ridurre la j-esima frontwidth scegliendo un vertice di grado basso $v_j \in adj(V)$. L'ordinamento indotto dal secondo vettore laplaciano non è una permutazione di adiacenza, ma ci sia avvicina per il teorema che segue (non ne riportiamo la dimostrazione, perchè molto lunga).

Teorema 3.7
Sia G un grafo connesso e sia $x = (x_1, x_2, \ldots, x_n)$ un secondo autovettore laplaciano di G. $\forall \rho \leq 0$ definiamo $\mathbf{S}(\boldsymbol{\rho}) = \{v_j \in V : x_j \geq \rho\}$. Allora il grafo indotto su $S(\rho)$ è connesso. Analogamente, se $\rho \geq 0$ allora $S'(\rho) = \{v_j \in V : x_j \leq \rho\}$ induce un sottografo connesso.

Riprendendo quanto detto sopra vediamo di chiarire perchè noi facciamo "quasi" una permutazione di adiacenza. Nella notazione del teorema, siano $v_j \in V$ ordinati in modo tale che $j \leq k$ sse $x_j \leq x_k$. Definiamo $P = \{v_j : x_j > 0\}$, numeraat da 1 a k $Z = \{v_j : x_j = 0\}$, numerati da k + 1 a p - 1 e $N = \{v_j : x_j < 0\}$, numerati da p a n. Si ha che $k < p$, dunque, per il teorema **??**, si ha $\forall j = p - 1, \ldots, n$ $v_{j+1} \in adj(V_j)$.
Si conclude che se i vertici che corrispondono a componenti positive sono inseriti in ordine crescente e quelli che corrispondono a componenti negative sono inseriti in ordine decrescente la permutazione ha la proprietà di essere una permutazione di adiacenza. Tuttavia, ci sono esempi semplici in cui questo non avviene e dunque la permutazione non è di adicenza.

3.5 Algoritmo

Per quanto detto finora, l'algoritmo che abbiamo implementato in Octave, suppondendo che la matrice sia irriducibile, consta tre passi:

1. Data la matrice M in forma sparsa creare la matrice Laplaciana L
2. Calcolare il secondo autovalore x_2 di L
3. Ordinare le componenti di x_2 in ordine non crescente e non decrescente e riordinare la matrice M con matrici di permutazione ottenute da entrambi gli ordinamenti di x_2. Scegliere la permutaazione che fornisce la taglia dell'inviluppo più piccola.

Dal punto di vista dell'implementazione la parte difficile è stata rappresentata dal secondo punto: l'algoritmo standard per calcolare gli autovalori di una grande matrice simmetrica sparsa è quello di Lanczos.

3.5.1 Algoritmo di Lanczos

L'algoritmo di Lanczos è un algoritmo iterativo messo a punto da Cornelius Lanczos, matematico e fisico ungherese vissuto a cavallo del XIX e del XX secolo. Questo algoritmo è un adattamento del metodo delle potenze per trovare i più importanti autovalori ed i relativi autovettori di una matrice $n \times n$, con n molto grande, in $m \ll n$ operazioni.

Anche se computazionalmente efficiente in principio, il metodo nella sua formulazione originale non era utile, poichè numericamente instabile. Per poter sfruttare il suo potenziale, altri matematici[**?**] lo hanno modificato e migliorato; attualmente il metodo di Lanczos è largamente usato in molti settori diversi ed ha un grande numero di varianti.

L'algoritmo vero e proprio è qui descritto e si applica a matrici hermitiane, ma è necessario tenere a mente che questo metodo, ovvero quello originale di Lanczos, non è numericamente stabile e non è quello di fatto implementato in Octave con il nome di `eigs`:

1. Il passo i-esimo dell'algoritmo trasforma la matrice A $n \times n$ in una matrice tridiagonale T_{mm} simile ad A.
2. Nel trasformare la matrice A nella matrice T_{mm} vengono creati dei vettori v_j detti **vettori di Lanczos** che sono le colonne della matrice di trasformazione e $V_m = (v_1, v_2, \cdots, v_m)$ servono per calcolare gli autovettori.
3. Dalla matrice T_{mm} è possibile calcolare gli autovalori e gli autovettori, ad esempio con il metodo QR.

Questo è lo pseudocodice del passo 1:
$v_1 \leftarrow$ `random vector with norm 1`
$v_0 \leftarrow 0$
$\beta_1 \leftarrow 0$
`for` $j = 1, 2, \ldots, m-1$
$w_j \leftarrow Av_j$
$\alpha_j \leftarrow < w_j, v_j >$
$w_j \leftarrow w_j - \alpha_j v_j - \beta_j v_{j-1}$

$\beta_{j+1} \leftarrow \|w_j\|$
$v_{j+1} \leftarrow w_j/\beta_{j+1}$
`endfor`
$w_m \leftarrow Av_m$
$\alpha_m \leftarrow w_m \cdot v_m$

che restituisce la matrice $T_{mm} = \begin{pmatrix} \alpha_1 & \beta_2 & & & & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \ddots & & \\ & & \ddots & \ddots & \beta_{m-1} & \\ & & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & & & & \beta_m & \alpha_m \end{pmatrix}$.

Nell'articolo di Barnard, Pothen e Simon è presentato inoltre un altro algoritmo, messo a punto da due[**?**] dei tre autori dell'articolo stesso, chiamato **metodo multilivello**.

3.5.2 Metodo multilivello

Questo metodo si propone come miglioramento in velocità rispetto al metodo di Lanczos applicato direttamente alla matrice, infatti opera sul grafo per ottenere una forma che renda più veloce applicare Lanczos. Il metodo multilivello si articola in tre parti:

Contrazione Si costruiscono dei piccoli grafi che mantengano la struttura di quello grande.

Interpolazione Dato il secondo autovettore di un grafo contratto, si interpola questo vettore con il grafo subito più grande, per ottenere una buona approssimazione del secondo autovettore del grafo più grande.

Raffinamento Dato il secondo autovettore del grafo iniziale, si calcola in modo efficiente un autovettore più accurato.

La contrazione è messa a punto trovando per prima cosa il massimo insieme di vertici indipendenti del grafo (*non-adiacenti* due a due), che vanno a formare il grafo contratto. Gli archi vengono trovati mediante una BFS che parte dai nodi del grafo contratto, aggiungendo un arco se si verifica la seguente condizione: da ogni nodo ci si espande ai nodi adiacenti e quando, tramite questa espansione, si raggiunge uno stesso vertice da due nodi distinti si aggiunge l'arco che collega i due nodi. Vengono creati grafi contratti fino a che il numero di vertici non raggiunge un limite prefissato.

Viene dunque applicato il metdo di Lanczos ed avviene l'interpolazione, che è raffinata dall'algoritmo chiamato "Rayleigh Quotient Iteration". Il processo di interpolazione-raffinamento viene iterato finchè non si sono esaurite tutte le possibilità, così da ottenere il secondo autovettore del grafo di partenza.

3.6 Risultati ottenuti

Nella sezione **??** è riportato il codice con il quale è stato implementato l'algoritmo Spectral e confrontato, come suggerisce l'articolo di Barnard, Pothen e Simon, con il metodo RCM implementato in Octave. Di seguito i confronti con quanto ottenuto dagli autori sopracitato ed alcune considerazioni sui dati riportati.

Matrice	Tempo (s)	Esize	Algoritmo	Val
	0	2159155	Null	3
Barth4	0.00804019	691734	RCM	2
	0.120163	326498	Spectral	1
	0	434798	Null	2
bcsstk13	0.011761	454503	RCM	3
	0.214249	418319	Spectral	1
	0	7441166	Null	1
bcsstk29	0.06831	7561056	RCM	2
	1.27442	73245227	Spectral	3
	0	15686968	Null	2
bcsstk30	0.269773	22003216	RCM	3
	2.89372	8712321	Spectral	1
	0	110527392	Null	3
bcsstk32	0.211628	52152181	RCM	2
	3.27401	23624011	Spectral	1
	0	3571395	Null	1
bcsstk33	0.073132	4007528	RCM	2
	2.43721	19534387	Spectral	3
	0	447931	Null	3
Random1000	0.00654602	12246	RCM	1
	0.0328641	15148	Spectral	2
	0	446773	Null	3
Random1001	0.00943398	12157	RCM	1
	0.0265691	24748	Spectral	2
	0	4223455	Null	3
Random3000	0.02005	61467	RCM	1
	0.0790141	73787	Spectral	2
	0	7563624	Null	3
Random4000	0.029094	94101	RCM	1
	0.123751	127584	Spectral	2

Nella tabella le prime 6 matrici sono state usate da Barnard, Pothen e Simon nell'articolo come prova dell'efficacia del loro algoritmo, mentre le successive sono matrici create con il programma **??**. Si può notare che i tempi di elaborazione di spectral sono notevolmente superiori a quelli di elaborazione di RCM, ma questa differenza è motivata dal fatto che RCM è implementato direttamente in Octave, mentre spectral è frutto di una function scritta da noi. Occupiamoci ora della taglia dell'inviluppo: nelle matrici proposte dai tre ricercatori spectral supera, tranne in due casi, RCM come efficienza. Tuttavia, nelle matrici campione costruite in modo da soddisfare le condizioni di simmetria, sparsità ed irriducibilità, spectral si dimostra a dir poco deludente, infatti oltre

ad essere lento nell'elaborazione è sempre secondo ad RCM nella taglia dell'inviluppo, di circa un fattore 2. A supporto dei dati numerici riportiamo delle figure che mostrano alcune matrici prima e dopo i due metodi qui analizzati (??? Figura1).

Finora le evidenze sperimentali da noi riportate non collimano con quanto sostenuto da Barnard, Pothen e Simon, tuttavia resta da affrontare la velocità nella fattorizzazione di Cholesky, accennata nella sezione **??**. Di seguito i dati raccolti:

Matrice	$\text{Tempo}_{\text{tot}}$(s)	$\text{Tempo}_{\text{parz}}$	Esize	Algoritmo	$\text{Val}_{\text{tot,parz}}$	
Barth4	0.0206459	0.0206459	2159155	Null	2	3
	0.0145202	0.00647998	691734	RCM	1	2
	0.1251960	0.00503302	326498	Spectral	3	1
bcsstk29	0.81697	0.81697	7441166	Null	2	3
	0.088956	0.058362	7561056	RCM	1	1
	1.720327	0.445907	73245227	Spectral	3	2
bcsstk30	0.182387	0.182387	15686968	Null	1	2
	0.441944	0.172171	22003216	RCM	2	3
	2.997399	0.103679	8712321	Spectral	3	1
bcsstk32	0.497944	0.497944	110527392	Null	1	3
	0.530944	0.319316	52152181	RCM	2	2
	3.494141	0.220131	23624011	Spectral	3	1
bcsstk33	0.0511069	0.0511069	3571395	Null	1	2
	0.114197	0.041065	4007528	RCM	2	1
	2.58454	0.14733	19534387	Spectral	3	3

La prima cosa che salta agli occhi guardando la tabella è che è più corta della precendente, infatti la fattorizzazione di Cholesky è possibile solo su matrici definite positive e non tutte quelle prese precedentemente in considerazione godono di questa proprietà. Dalla tabella si può notare come, sebbene nei tempi parziali la fattorizzazione di Cholesky a partire dalla matrice ordinata da spectral sia spesso la più veloce, questo primato venga perso non appena si valutano i tempi totali (tempo per la fattorizzazione di Cholesky + tempo per l'ordinamento).

3.7 Codice Octave

L'algoritmo della sezione **??** è stato implementato in diverse funzioni qui elencate.

3.7.1 sparsePD

[serve solo per generare matrici]

```
1 function M = sparsePD(n)
2         lens = round(rand(1, n) .* sqrt(0:(n-1)));
```

```
        lens(2:n) += 1;
        lens = min(lens, 0:(n-1));
        pos = 1;
        tot = sum(lens);
        c = zeros(1, tot);
        r = zeros(1, tot);
        for i = 1:n
                c(pos:(pos+lens(i)-1)) = ones(1, lens(i)) * i;
                r(pos:(pos+lens(i)-1)) = i - (1:lens(i));
                pos += lens(i);
        endfor
        M = sparse(r, c, rand(1, tot), n, n, 0);
        M = M + M';
        M = M + diag(sum(M) + 1);
        p = randperm(n);
        M = M(p, p);
endfunction
```

3.7.2 rcm.m

```
function M = rcm(A)
        p = symrcm(A);
        M = A(p, p);
endfunction
```

3.7.3 Esize.m

```
function n = Esize(B)
        dim = size(B)(1);
        [x, y] = find(B);
        v = y;
        v(1) = 0;
                v(2:end) = y(1: (end - 1));
        y = y - v;
        v = find(y);
        rowpos = x(v);
        n = sum((1:dim)' - rowpos);
endfunction
```

3.7.4 spectral.m

```
function M = spectral(B)
        A = (B != 0);
```

```
        A = A - diag(diag(A));
        D = diag(sum(A));
        L = D-A;
        opts.tol = 1e-25;
        opts.nit = 1000;
        [V, _] = eigs(L, 2, "sm", opts);
        x2 = V(:,1);
        [_, t] = sort(x2);
        S1 = B(t, t);
        [_, t] = sort(-x2);
        S2 = B(t, t);
        if Esize(S1) < Esize(S2)
                M = S1;
        else
                M = S2;
        endif
endfunction
```

Queste funzioni sono utilizzate in un programma Octave chiamato **test.m**, qui riportato:

```
#!/usr/bin/octave -qf
B = mmread(argv(){1});
sizeB = Esize(B)
spy(B, 1);
tic
[_, p] = chol(B);
if p
        toc
end
input("", "s");

tic
R = rcm(B);
toc
sizeR = Esize(R)
spy(R, 1);
tic
[_, p] = chol(R);
if p
        toc
end

input("", "s");

```

```
tic
S = spectral(B);
toc
sizeS = Esize(S)
spy(S, 1);
tic
[_, p] = chol(S);
if p
        toc
end

input("", "s");
```

4 Conclusioni

Si dubita, inoltre, che l'inefficacia del metodo sia da attribuirsi ad errori di implementazione perchè non solo l'algoritmo è stato provato su casi in cui era possibile effettuare i calcoli a mano, ma i dati raccolti collimano (per quanto l'apparecchiatura usata da Barnard, Pothen e Simon fosse più obsoleta di quella utilizzata in questa sperimentazione) con quelli riportati nell'articolo di riferimento. Da un'accurata analisi degli algoritmi proposti e dei dati raccolti si può concludere che spectral rappresenta sì un approccio originale al problema della riduzione della taglia dell'inviluppo di una matrice sparsa simmetrica, ma di fatto non effettua sostanziali miglioramenti allo standard in nessuno dei due ambiti presi in esame.

Figura 1: Si nota bene che la taglia dell'inviluppo viene ridotta di più con l'algoritmo RCM che con spectral.

Riferimenti bibliografici

[1] Stephen T. Barnard, Alex Pothen, Horst D. Simon, "A spectral algorithm for envelope reduction of sparse matrices", Shorter version in *Supercomputing '93* October 1993.

[2] J. A. George and A. Pothen, "An analysis of the spectral approach to envelope reduction via quadratic assignment problems" *http://www.cs.odu.edu/ pothen/papers.html* 1994.

[3] M. Fiedler, "Algebraic connectivity of graphs" *Czech. Math. J., 23(1973)*, pp. 298-305.

[4] M. Fiedler, "Algebraische zusammenhangszahl der graphen und ihre numerische bedeutung" *Numerische Methoden bei graphentheoretischen und kombinatorischen Problemen, Oberwolfach 1974, ISNM, L. Collatz, H. Werner and G. Meinardus, eds., vol. 29, Birkhauser Varlag, 1975*, pp. 69-85, in German.

[5] M. Fiedler, "A property of eigenvectors f non-negative symmetric matrices and its applicatin to graph theory" *Czech. Math. J., 25 (1975)*, pp. 619-633.

[6] M. Juvan and B. Mohar "Optimal linear labelings and aigenvalues of graphs" *Discr. Appl. Math., 36 (1992)*, pp.153-168.

[7] I.U. Ojalvo and M. Newman,"Vibration modes of large structures by an automatic matrix-reduction method", *AIAA J., 8 (7), 1234-1239 (1970).*

[8] S.T Barnard and H. D. Simon, "A fast multilevel implementtion of recursive spectral bisection for partitioning unstructured problems" *Tech. report RNR-092-033, NASA Ames Research Center, Moffett Field, CA 94035, Nov. 1992.*