UNIVERSITÀ DEGLI STUDI DI CAGLIARI

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in
Ingegneria Elettrica, Elettronica, Informatica

Analisi di reti complesse mediante indici di centralità

Relatore
Prof. Giuseppe Rodriguez

Studente
Giulia Gramignano
Matr. N. 70/89/00183

ANNO ACCADEMICO 2020/21

Indice

Elenco delle figure

Introduzione

L'importanza dello studio delle reti nasce dal fatto che i sistemi complessi hanno come caratteristica fondamentale la connettività tra elementi, inoltre siamo capaci di rappresentare come reti anche concetti astratti dove le entità corrispondono ai nodi e le relazioni o interazioni tra esse agli archi. Possiamo quindi descrivere mediante grafi sia le interazioni fisiche, come quelle ad esempio che avvengono tra le proteine, sia dei collegamenti fisici, come quelli di una rete autostradale, che connessioni astratte, come quelle sociali.
In questa tesi definiremo inizialmente gli strumenti matematici necessari per poter svolgere un'analisi sulle reti, poi passeremo a illustrare alcuni metodi utili per misurare l'importanza dei nodi in base a diversi punti di vista; infine, svolgeremo l'analisi della rete di autobus CTM della città di Cagliari, andando a individure nel caso reale e ideale quali sono le fermate di maggiore interesse dal punto di vista della connettività.

Capitolo 1

Reti e matrici

1.1 Reti

Le reti possono essere viste come un insieme di elementi connessi tra loro.
Si può definire una **rete** G la coppia (V,E), dove **V**=$(v_1, v_2, ..., v_n)$ è l'insieme dei *nodi* mentre **E**=$(e_1, e_2, ..., e_n)$ è l'insieme degli *archi*:

- se E è simmetrico [1], allora G è una rete **non orientata**, altrimenti è **orientata**;
- mentre se E è simmetrico, anti-riflessivo [2] e non contiene duplicati dei nodi, allora G è una rete **semplice**;
- se per ogni coppia di nodi $(u, v) \in V$, esiste un cammino che collega u a v, allora il grafo si dice **connesso**.

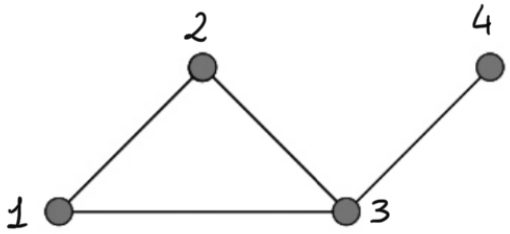


Figura 1.1: *Un esempio di rete semplice e undirected con V=(1,2,3,4) e E=[(1,2), (1,3), (2,1), (2,3), (3,1), (3,2), (3,4), (4,3)]*

[1] se $(v_1, v_2) \in E \iff (v_2, v_1) \in E$.
[2] se per ogni v in V, $(v, v) \notin E$.

1.2 Matrici

1.2.1 Matrice di adiacenza

Possiamo rappresentare una rete mettendo in risalto le *connessioni* tra i suoi elementi, ottenendo quindi la matrice di adiacenza.
Supponiamo che $G = (V, E)$ sia una rete semplice dove $V = (1, 2, ..., n)$ e con $1 \leq i, j \leq n$, definiamo:

$$a_{ij} = \begin{cases} 1, & (i,j) \in E, \\ 0 & (i,j) \notin E. \end{cases}$$

Allora la matrice quadrata $A = (a_{ij})$ è detta matrice di adiacenza di G.
Quindi per costruire A si pone 1 se il nodo i è connesso al nodo j, altrimenti si pone 0.
Ad esempio, la matrice di adiacenza per la rete in figura 1.1 sarà

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

1.2.2 Matrice di incidenza

La matrice di incidenza, invece, adotta un altro punto di vista rispetto a quella di adiacenza, concentrandosi maggiormente sugli *elementi* di una rete. Supponiamo che $G = (V, E)$ sia una rete dove $V = (1, 2, ..., n)$ e $E = (e_1, e_2, \ldots, e_m)$, con $e_i = (u_i, v_i)$. Per $1 \leq i \leq m$ e $1 \leq j \leq n$, definiamo:

$$b_{ij} = \begin{cases} 1, & u_i = j, \\ -1, & v_i = j, \\ 0 & altrimenti. \end{cases}$$

allora la matrice quadrata $B = b_{ij}$ è detta matrice di incidenza di G.
Riferendoci alla rete in figura 1.2, la sua matrice di incidenza sarà

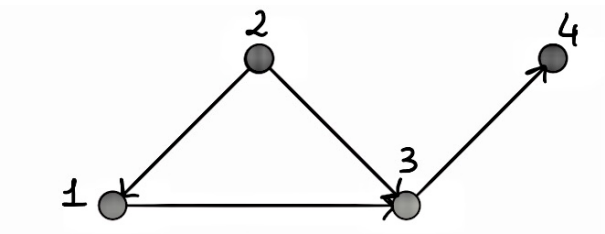


Figura 1.2: *Analogo caso di prima di rete semplice e undirected ma questa volta orientata*

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 0 & -1 & 0 \\ 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

1.3 Grafi

In MATLAB, si può creare un grafo in due modi:

- attraverso la **matrice di adiacenza**: dove gli elementi non nulli indicano una connessione tra due nodi. Si usa il comando *G=graph(A)*, dove A deve essere simmetrica se si ha una rete non orientata, altrimenti si usa il comando *digraph* che origina un grafo orientato. Viceversa, si può risalire alla matrice di adiacenza dato un certo grafo, attraverso la funzione *adjacency*;
- attraverso la **lista degli archi**: dove si indicano le colonne dei nodi source e target. Per le reti orientatiquesta distinzione è fondamentale, mentre per quelle non orientate le colonne sono interscambiabili. Prendendo la rete orientata in figura 1.2, i comandi MATLAB sarebbero:

```
>> source_nodes={'1','2','2','3'}

source_nodes =

  1×4 cell array

    {'1'}    {'2'}    {'2'}    {'3'}

>> target_nodes={'3','1','3','4'}

target_nodes =

  1×4 cell array

    {'3'}    {'1'}    {'3'}    {'4'}

>> G=graph(source_nodes, target_nodes)
```

Per **accedere ai nodi** di un grafo possiamo utilizzare il loro *indice numerico* o il loro *nome*. Per default, le funzioni graph e digraph numerano i nodi e questo va

a costituire il loro indice numerico, altrimenti possiamo individuarli con un alias grazie alla proprietà *G.Nodes* che contiene la variabile *Name*. Inoltre, attarverso la funzione *findnode* è possibile trovare l'indice numerico di un nodo dato il suo nome, oppure viceversa, trovare il nome di un nodo dato un certo indice attraverso *G.Nodes.Name*. Sempre per la rete in figura 1.2 abbiamo lo script in figura 1.3
Per **disegnare un grafo** si usa la funzione *plot* che mostra per default i nodi con il loro indice o nome come etichetta (tranne per le reti molto grandi). Se si chiama la funzione plot specificando un output *p=plot(G)*, questa restituirà un puntatore p ad un oggetto *GraphPlot* che può essere utilizzare per cambiare colore o stile delle connessioni e nodi, accedendo alle varie proprietà (*p.NodeColor='red'*). Al fine di effettuare un'analisi sulle varie centralità dei nodi di una rete, può essere utile la **colorazione proporzionale al nodo**, che permette di colorare il nodo in base al loro grado di centralità (o qualsiasi altra misura di centralità). Prendiamo ad esempio la rete in figura 1.1 e la sua matrice di adiacenza A:

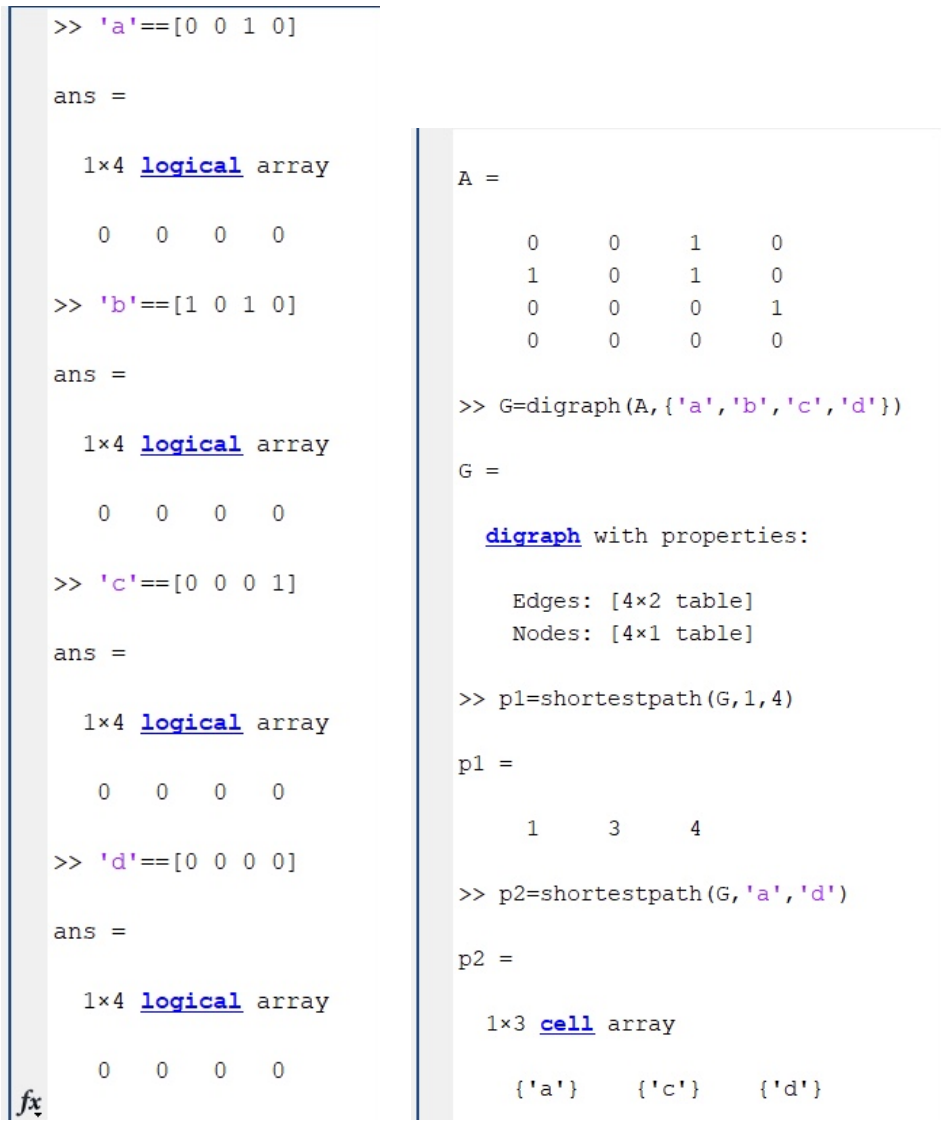


Figura 1.3: *Nella colonna di sinistra attribuiamo ai vari nodi un nome, assegnando a ciascuno una riga della matrice di adiacenza.Come si nota dalla figura possiamo assegnare i nodi ad una funzione sia con il loro nome, come avviene per p1, che con il loro indice, come avviene per p2, infatti i due vettori risultanti p1 e p2 sono uguali.*

```
>> A=[0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0]
G=graph(A)
p=plot(G)
G.Nodes.NodeColors=degree(G)
p.NodeCData = G.Nodes.NodeColors
colorbar
```

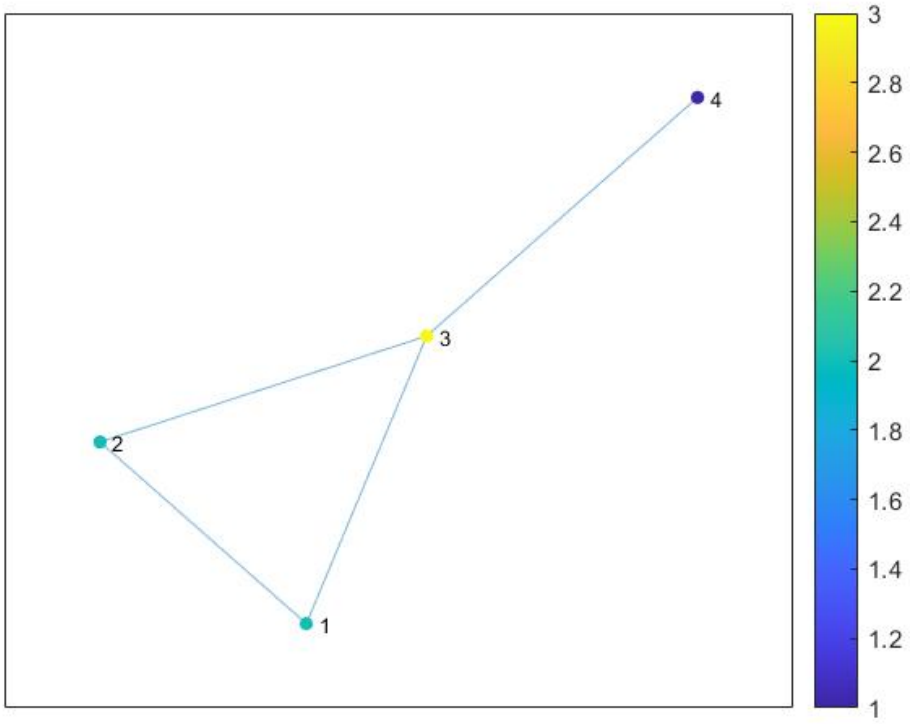


Figura 1.4: *Si nota che il massimo grado di centralità è 3 ed è associato al nodo 3, mentre i nodi 1 e 2 hanno grado 2 e il nodo 4 è quello meno centrale con grado 1.*

1.4 Matrici sparse

Le matrici sparse sono matrici nelle quali la maggior parte degli elementi sono nulli, al contrario di quelle che invece hanno la maggior parte degli elementi diversi da zero, dette matrci dense.

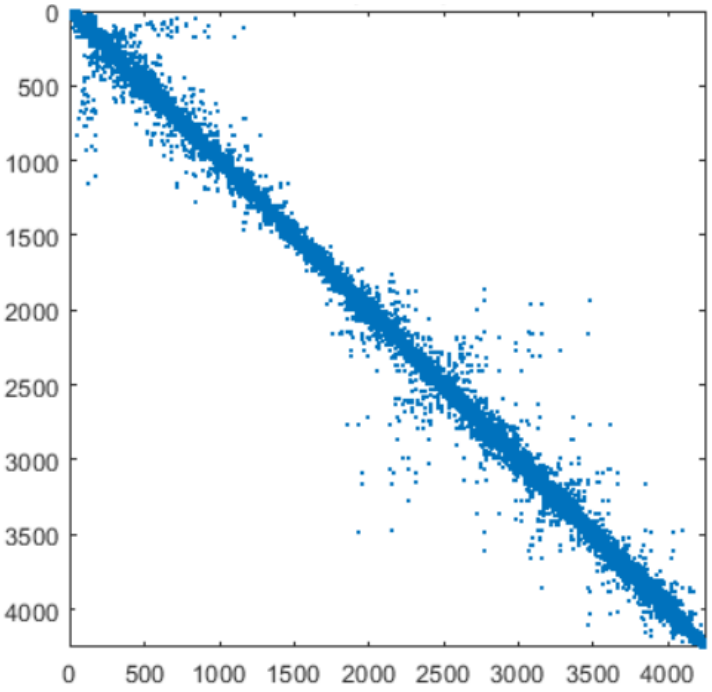


Figura 1.5: *I puntini blu indicano tutti gli elementi non nulli all'interno della matrice sparsa*

Il numero degli elementi non nulli diviso il numero totale degli elementi (righe x colonne), ci fornisce la **densità** della matrice. Il **grado di sparsità** invece si definisce come:

$$\text{sparsità} = 1 - \text{densità}$$

e per una matrice sparsa questo valore è di circa 1, essendo la densità molto bassa.
Creiamo su MATLAB una matrice random sparsa
Per determinare densità e sparsità occorre conoscere il numero di elementi non nulli, attraverso:
Siamo quindi in grado di calcolare la densità della matrice come:

$$\textit{densità} = \frac{nnz}{(nxm)} = \frac{15}{64} \approx 0.3$$

```
>> M_sparse = sprand(8,8,0.3)

M_sparse =

   (2,1)        0.0344
   (7,1)        0.4456
   (8,1)        0.7547
   (2,2)        0.4387
   (4,3)        0.7655
   (7,3)        0.6463
   (8,4)        0.2760
   (1,6)        0.9502
   (5,6)        0.1869
   (8,6)        0.6797
   (3,7)        0.3816
   (4,7)        0.7952
   (8,7)        0.6551
   (6,8)        0.4898
   (7,8)        0.7094
```

```
>> nnz(M_sparse)

ans =

    15
```

E perciò la sua sparsità sarà:

$$sparsità = 1 - 0.3 \approx 0.7$$

Il concetto di sparsità è utile nel campo delle network theory, che solitamente ha a che fare con reti molto grandi in cui gli elementi non sono densamente connessi tra loro, e quindi la densità della matrice è bassa,

1.4.1 Vantaggi computazionali

Quando si devono effettuare calcoli basati su matrici sparse attraverso strumenti di calcolo come ad esempio i computer, si devono usare degli **algoritmi specializzati** che sfruttano a proprio vantaggio le caratteristiche delle matrici sparse.
Se non si usano questi algoritmi le operazioni di calcolo possono prendere molto tempo oppure fallire; inoltre, sempre sfruttando la caratteristica di avere la maggior parte degli elementi nulli, si ha un vantaggio anche dal punto di vista della **memorizzazione** della matrice.
In MATLAB l'attributo *sparse* consente di:

- memorizzare solo gli elementi diversi da zero, assieme ai loro indici di riga e colonna;
- ridurre il tempo di computazione eliminando le operazioni su elementi nulli.

Ipotizziamo di prendere la matrice sparsa considerata precedentemente e convertirla in una matrice densa con il comando $full$. Utilizzando poi il comando *whos* è possibile visualizzare:

```
>> whos
  Name          Size              Bytes  Class     Attributes

  M_full        8x8                 512  double
  M_sparse      8x8                 312  double    sparse
```

Dall'immagine si nota che il numero di bytes usati nel caso di matrice sparsa è molto minore rispetto a quello di una matrice densa, proprio perchè vengono conservati solo gli elementi non nulli.

1.4.2 Memorizzazione

MATLAB attraverso l'attributo *sparse* memorizza solo gli elementi non nulli della matrice sparsa, assieme alle relative coordinate (riga, colonna), come si nota infatti dalla matrice sparsa creata precedentemente:

```
(2,1)        0.0344
(7,1)        0.4456
(8,1)        0.7547
(2,2)        0.4387
(4,3)        0.7655
(7,3)        0.6463
(8,4)        0.2760
(1,6)        0.9502
(5,6)        0.1869
(8,6)        0.6797
(3,7)        0.3816
(4,7)        0.7952
(8,7)        0.6551
(6,8)        0.4898
(7,8)        0.7094
```

Figura 1.6: *Si nota che gli elementi sono in ordine di colonna*

Oltre a questo metodo, ne esistono altri che sfruttano il ridotto numero di elementi non nulli, come i metodi CSC e CSR.

Metodo CSC

Sia $A(nxn)$ una matrice sparsa, per memorizzarla si può utilizzare il metodo CSC, ovvero *Compressed Sparse Columns* format che consiste nel memorizzare la matrice tramite 3 array di A:

- R: contiene gli elementi non nulli di A memorizzati per colonna;
- I: I_k contiene l'indice della riga nella matrice A corrispondente all'elemento R_k;
- C: l'elemento C_k contiene la posizione nell'array R del 1° elemento non nullo della k-esima colonna in A.

$$A = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix}$$

$$R = \begin{pmatrix} 1 & 3 & 6 & 4 & 7 & 10 & 2 & 5 & 8 & 11 & 9 & 12 \end{pmatrix}$$

$$I = \begin{pmatrix} 1 & 2 & 3 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 3 & 5 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 4 & 5 & 7 & 11 \end{pmatrix}$$

Metodo CSR

Il metodo CSR, ovvero *Compressed Sparse Row* format, rappresenta come il CSC una matrice A(mxn) attraverso 3 array:

- R: contiene gli elementi non nulli di A memorizzati per riga;
- C: C_k contiene l'indice della colonna nella matrice A corrispondente all'elemento R_k;
- I: di lunghezza m+1, dove I_k contiene l'indice in R del primo elemento non nullo della k-esima riga di A, e come ultimo elemento il valore di nz+1 (numero elementi non nulli incremenetato di uno);

Prendiamo come esempio la matrice A utilizzata nel metodo CSC. I tre array saranno:

$$R = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 4 & 1 & 2 & 4 & 1 & 3 & 4 & 5 & 3 & 4 & 5 \end{pmatrix}$$

$$I = \begin{pmatrix} 1 & 3 & 6 & 10 & 12 & 13 \end{pmatrix}$$

1.5 Metodi iterativi

I metodi iterativi sono convenienti per matrici di grandi dimensioni, soprattutto se sparse, perchè a differenza dei metodi diretti, non alterano la struttura della matrice o aumentano il numero degli elementi non nulli.
Una strategia usata comunemente per la costruzione di metodi iterativi lineari è quella dello *splitting additivo*, che consiste nello scrivere la matrice del sistema nella forma:

$$A = P - N$$

dove P è la *matrice di precondizionamento* e non singolare.

Sostituendo questa relazione nel sistema lineare $Ax = b$, si ottiene il sistema equivalente:

$$Px = Nx + b$$

che porta alla definizione del metodo iterativo

$$Px^{(k+1)} = Nx^{(k)} + b$$

Si nota subito che un metodo così costruito è *consistente* [3] ed essendo P non singolare, allora $\det(P) \neq 0$, e possiamo scrivere:

$$x^{(k+1)} = P^{-1}Nx^{(k)} + P^{-1}b = Bx^{(k)} + f$$

con $B = P^{-1}N$ e $f = P^{-1}b$.

Il **metodo di Jacobi** corrisponde alla scelta

$$P = D,\ N = L + U$$

dove D, L e U sono rispettivamente la diagonale, il triangolo inferiore e quello superiore di A cambiati di segno, ovvero:

$$D_{ij} = \begin{cases} a_{ij}, & i = j \\ 0 & i \neq j, \end{cases} \quad L_{ij} = \begin{cases} -a_{ij}, & i > j \\ 0 & i \leq j, \end{cases} \quad U_{ij} = \begin{cases} -a_{ij}, & i < j \\ 0 & i \geq j \end{cases}$$

Il metodo iterativo diventa quindi:

$$Dx^{(k+1)} = Lx^{(k)} + Ux^{(k)} + b$$

da cui

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b = B_J x^{(k)} + f$$

La matrice di iterazione è

$$B_J = P^{-1}N = D^{-1}(L + U) = I - D^{-1}A$$

che determina la *convergenza.* [4]

[3] Un metodo iterativo si dice consistente se $x^{(k)} = x \Rightarrow x^{(k+1)} = x$

[4] Un metodo iterativo lineare è convergente se e solo se il raggio spettrale $\rho(B)$ della matrice di iterazione B è minore di 1.

Capitolo 2

Misure di centralità

Il concetto di centralità viene usato per determinare il nodo più importante in una rete, le cui caratteristiche principali sono la sua abilità di *comunicare direttamente* con altri nodi, la *closeness centrality* (vicinanza ad essi) e la sua *betweenness centrality* (ruolo di mediatore) tra diverse parti di una rete.

2.1 Grado di centralità

Il grado di centralità semplice misura l'abilità di un nodo di comunicare **direttamente** con alri nodi. Il grado di un nodo i in una rete semplice G è definito usando la matrice di adiacenza della rete:

$$k_i = \sum_{j=1}^{n} a_{ij} = (e^T A)_i = (Ae)_i$$

dove e è un vettore di soli 1. Quindi in una rete, il nodo i è più centrale di un nodo j se $k_i > k_j$.
In una **rete orientata**, dove il grado può essere diverso in base all'orientamento dell'arco tra i nodi, utilizziamo due differenti misure di centralità:

$$k_i^{in} = \sum_{i=1}^{n} a_{ji} = (A^T e)_i \ , \ k_i^{out} = \sum_{j=1}^{n} a_{ij} = (Ae)_i$$

Considerando la rete in figura 2.1

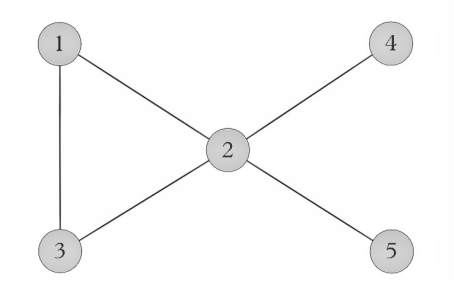


Figura 2.1: *Un esempio di rete semplice*

la sua matrice di adiacenza è

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

mentre il suo vettore del grado di centralità è:

$$k = \begin{pmatrix} 2 & 4 & 2 & 1 & 1 \end{pmatrix}$$

e da esso quindi, $k(1)=k(3)=2$, $k(2)=4$ e $k(4)=k(5)=1$, perciò il nodo centrale è il nodo 2.
Se ipotizziamo di prendere la stessa rete ma orientata come in figura 2.2:

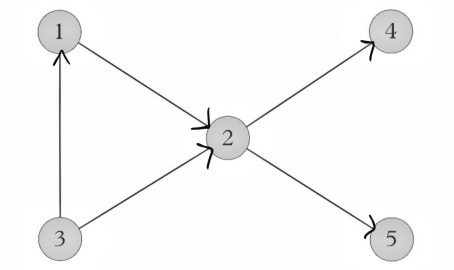


Figura 2.2: *Un esempio di rete semplice orientata*

la sua matrice di adiacenza, costruita con $a_{ij}=1$ se arco orientato in uscita dal nodo

i, altrimenti a_{ij}=0, diventa quindi:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

i suoi vettori del grado di centralità sono:

$$k^{in} = A^T e = \begin{pmatrix} 1 & 2 & 0 & 1 & 1 \end{pmatrix},\ k^{out} = Ae = \begin{pmatrix} 2 & 4 & 2 & 1 & 1 \end{pmatrix}$$

2.2 Closeness centrality

La closeness centrality di un nodo indica quanto è *vicino* agli altri nodi della rete, ed è misurata in termini di **cammino più breve**:

$$CC(i) = \frac{(n-1)}{s(i)}$$

dove la distanza totale $s(i)$ è calcolata come

$$s(i) = \sum_{j \in V(G)} d(i,j)$$

dove $d(i,j)$ è la distanza tra il nodo i e il nodo j.
Nelle reti orientate anche la closeness centrality è differente in base all'orientamento dell'arco tra i due nodi:

- *out-closeness* centrality indica quanto un nodo è vicino a quelli a cui invia le informazioni;

- *in-closeness* centrality indica quanto un nodo è vicino a quelli da cui riceve le informazioni.

In queste reti, il cammino più breve è una sorta di pseudo-distanza a causa della possibile mancanza di simmetria.

Considerando la rete in figura 2.1, costruiamo la matrice di distanza [1]:

$$D = \begin{pmatrix} 0 & 1 & 1 & 2 & 2 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 2 \\ 2 & 1 & 2 & 0 & 2 \\ 2 & 1 & 2 & 2 & 0 \end{pmatrix}$$

il vettore della distanza totale [2] è

$$s = \begin{pmatrix} 6 & 4 & 6 & 7 & 7 \end{pmatrix}$$

calcoliamo la closeness centrality ad esempio del nodo 1:

$$CC(1) = \frac{(5-1)}{6} \approx 0.67$$

Per calcolare la closeness centrality possiamo pensare di implementare un **algoritmo** che sfrutta il fatto che la matrice esponenziale A^k va ad identificare i percorsi di lunghezza k associati ad ogni nodo della rete. Ogni qualvolta che si ha un elemento $(a_{ij})^k \neq 0$ laddove $d_{ij} = 0$, assegneremo all'elemento della matrice di distanza il valore k, e così via fino a quando tutti gli elementi di D saranno non nulli. Dato che però per definizione gli elementi della diagonale della matrice di distanza devono essere nulli, provvederemo poi ad annullarli. Il codice dell'algoritmo quindi è:

[1] dove d_{ij} è la lunghezza del percorso più breve tra il nodo i e il nodo j.
[2] definito come s=De.

```
function [cc,D] = mycloscen(A)
% funzione che data in ingresso la matrice di adiacenza A restituisce la
% matrice delle distanze D e il vettore delle closenesscentrality cc

D=zeros(size(A));
[row, col]=size(A);
B=A;
k=1;
tot=row*col;
while nnz(D)~=tot %continuare il ciclo fino a quando non si è riempita la matrice D
    for i=1:row %ciclo for che itera sulle righe della matrice, ovvero su ogni nodo della rete
        vv = (A(i,:)~=0) & (D(i,:)==0); %vettore logico che contiene il risultato dell'operazione
                                        % logica AND e quindi individua le
                                        % colonne dove si può riempire D



        D(i,vv) = k; %assegnazione del percorso di lunghezza k
    end
    A=A*B; %incremento esponente matrice A per individuare percorsi di lunghezza k da ogni nodo
    k=k+1;
end
d = diag(D); %matrice che ha come elementi della diagonale gli stessi di D
D = D-diag(d);  % annulla la diagonale di D
cc = (row-1) ./ (D*ones(row,1)); %calcolo closeness centrality
```

2.3 Betweenness centrality

La betweenness centrality indica quanto un nodo è importante nella comunicazione tra diverse parti della rete, ed è definita come:

$$BC(i) = \sum_j \sum_k \frac{p(i,j,k)}{p(j,k)}, i \neq j \neq k$$

dove *p(j,k)* è il numero di cammini più brevi che collegano il nodo j al nodo k, mentre *p(i,j,k)* è il numero di cammini più brevi tra i due nodi che passano anche per il nodo i.
Se la rete è orientata, il termina $p(i,j,k)$ si riferisce al numero di cammini orientati dal nodo j al nodo k che passano per il nodo i, e $p(j,k)$ al numero totale di cammini orientati dal nodo j al nodo k.
Consideriamo sempre la rete in figura 2.1 e calcoliamo la betweenness centrality del nodo i=2:

(j,k)	p(j,2,k)	p(j,k)	p(j,2,k)/p(j,k)
1,3	0	1	0
1,4	1	1	1
1,5	1	1	1
3,4	1	1	1
3,5	1	1	1
4,5	1	1	1

Quindi la betweenness centrality del nodo 2, data dalla somma degli elementi dell'ultima colonna è:

$$BC(2) = 0 + 1 + 1 + 1 + 1 + 1 = 5$$

2.4 Position centrality

Permette di determinare il nodo che occupa la posizione più centrale nella rete, in modo che se deve inviare delle informazioni a tutti gli altri nodi, allora potrà farlo nel minore tempo possibile, rispetto a tutti gli altri nodi.
Sia τ uno ***spanning-tree*** del grafo G, che inizia dal nodo v e sia $\upsilon_1, \upsilon_2, \ldots, \upsilon_l$ il percorso di lunghezza l determinato dall'albero. La position centrality P_c di v nel grafo è la somma delle lunghezze dei cammini da v a tutti gli altri nodi:

$$P_c = \sum_{k=1}^{l-1} k(\#\upsilon_{k+1})$$

dove $\#\upsilon_i$ indica la cardinalità (ovvero numero di elementi) dell'insieme υ_i. Minore sarà questo valore P_c, più sarà centrale la posizione del nodo v.
Consideriamo il grafico in figura 2.3. Per calcolare la position centrality di un certo nodo, consideriamo lo spanning-tree [3] che ha come radice tale nodo.
Prendiamo ad esempio il nodo 2, abbiamo: $\upsilon_1 = v_2$, $\upsilon_2 = v_3$, $\upsilon_3 = v_4$, $\upsilon_4 = \{v_1, v_5, v_8\}$ e $\upsilon_5 = \{v_6, v_7, v_9, v_{10}\}$. Dato che la rete è non orientata, la distanza tra due nodi ha lunghezza unitaria. Segue che la position centrality del nodo è data da:

$$P_c = 1x1 + 2x1 + 3x3 + 4x4 = 28$$

[3] è un albero che contiene tutti i nodi del grafo e soltanto gli archi necessari per connettere tra loro tutti i vertici con uno e un solo cammino

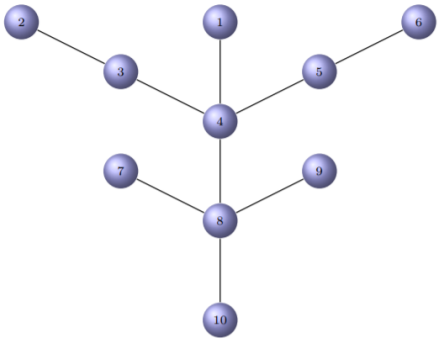


Figura 2.3: *Esempio di rete per cui determinare il nodo con posizione più centrale*

Analizziamo poi il nodo 4: $\upsilon_1 = v_4$, $\upsilon_2 = \{v_1, v_3, v_5, v_8\}$, $\upsilon_3 = \{v_2, v_6, v_7, v_9, v_{10}\}$, $\upsilon_4 = 0$, $\upsilon_5 = 0$, quindi si ha:

$$P_c = 1x4 + 2x5 + 3x0 + 4x0 = 14$$

Il valore di position centrality in questo caso è minore e se lo calcolassimo anche per tutti gli altri nodi vedremmo ch alla fine il nodo 4 è quello con valore di P_c minore e perciò nella rete occupa la posizione più centrale (come si poteva intuire anche dal grafico).
Nell'esempio considerato, il nodo con posizione più centrale appartiene ad uno dei rami più lunghi dello spanning-tree e questo accade anche nella maggior parte dei casi, quindi per semplificare possiamo decidere di calcolare il valore P_c solo per i nodi dei rami più lunghi anzichè per tutti (questa però è una regola euristica, perchè non è detto che il nodo con posizione più centrale debba appartenere per forza ad uno dei rami più lunghi.
Il seguente codice MATLAB ci permette di calcolare la il nodo con posizione centrale maggiore nella rete e il relativo valore di position centrality:

centralnode.m | getconcomp.m | graphvisit.m | prova1.m

```
function [cnod, pcen] = centralnode(A,nod)
%CENTRALNODE approximate center node of A
%    [cnod,pcen]=apprcnode(A,nod), where nod is the starting node

cnod = nod;
pcen = inf;
flag = 1;
while flag
    [ptree,V] = spantree(A,cnod);
    [len, list] = getlbranches(ptree);
    nlist = length(list);
    pc = zeros(len,nlist);
    for k = 1:nlist
        for i = 1:len
            [ptree,V] = spantree(A,list{k}(i));
            pc(i,k) = poscen(V);
        end
    end
    [m1 i1] = min(pc);
    [m2 i2] = min(m1);
    if m2<pcen
        cnod = list{i2}(i1(i2));
        pcen = m2;
    else
        flag = 0;
    end
end
```

centralnode.m | getconcomp.m | graphvisit.m | prova1.m

```
29      % spanning tree [p,V]=spantree(A,k)
30      %    A adjacency matrix
31      %    k starting node
32      %    p spanning tree
33      %    V chain structure determined by p
34      function [p,V] = spantree(A,k)
35
36      n = size(A,1);
37      V = cell(n,1);
38      V{1} = k;
39      nodes = V{1};
40      dpth = 1;
41      lnod = length(nodes);
42      while lnod<n
43          dpth = dpth+1;
44          v = [];
45          for nod = V{dpth-1}
46              w = find(A(nod,:));
47              lst = ismember(w,nodes);
48              w(lst) = [];
49              v = [v w];
50              nodes = [nodes w];
51          end
52          nodes = sort(nodes);
53          V{dpth} = sort(v);
54          if length(nodes)==lnod
55              warning('The graph is not connected.')
```

centralnode.m | getconcomp.m | graphvisit.m | prova1.m

```
55 -            warning('The graph is not connected.')
56 -            warning('The output tree spans only a connected component.')
57 -            break
58 -        end
59 -        lnod = length(nodes);
60 -    end
61 -    V = V(1:dpth);
62
63 -    p = zeros(1,n);
64 -    for i = dpth:-1:2
65 -        v = V{i};
66 -        for j = 1:length(v);
67 -            nod = v(j);
68 -            u = V{i-1};
69 -            w = A(nod,u);
70 -            z = find(w);
71 -            p(v(j)) = u(z(1));
72 -        end
73 -    end
74 -    p(V{1}) = 0;
75
76      function [len,list] = getlbranches(tree,node)
77      %GETLBRANCHES get all longest branches from tree
78      %    [len,list]=getlbranches(tree) returns the length of the branch and a cell
79      %    array containing a list of nodes for each longest branch
80
81 -    if nargin<2; node = find(0==tree); end
```

centralnode.m | getconcomp.m | graphvisit.m | prova1.m

```
81 -    if nargin<2; node = find(0==tree); end
82
83 -    list{1} = [];
84 -    nl = 1;
85 -    len = 0;
86 -    children = find(node==tree);
87 -    for child = children    % for all children of node
88 -        [len1,list1] = getlbranches(tree,child);
89 -        if len1>len
90 -            len = len1;
91 -            clear list
92 -            nl = 0;
93 -            for i = 1:length(list1)
94 -                nl = nl+1;
95 -                list{nl} = list1{i};
96 -            end
97 -        elseif len1==len
98 -            for i = 1:length(list1)
99 -                nl = nl+1;
100 -               list{nl} = list1{i};
101 -           end
102 -       end
103 -   end
104 -   for i = 1:nl
105 -       list{i} = [node;list{i}];
106 -   end
107 -   len = 1+len;
108
109     function pc = poscen(V)
110     % position centrality of root node in V
```

```
109     function pc = poscen(V)
110     % position centrality of root node in V
111
112 -   n = length(V);
113 -   pc = 0;
114 -   for i = 1:n-1
115 -       pc = pc+i*length(V{i+1});
116 -   end
117
118
```

centralnode.m | getconcomp.m | graphvisit.m | prova1.m | +

```
1   function  U = getconcomp(A,opts)
2   %GETCONCOMP determine the connected components of a graph.
3   %   U = getconcomp(A) returns a cell array containing the vector of indices
4   %   which identify the connected components of the graph corresponding to
5   %   the adjacency matrix A.
6   %   U = getconcomp(A,opts) optionally passes a set of options.
7   %   Warning: to reduce stack usage, a global matrix AGLOB is used.
8   %
9   %   options:
10  %   opts.maxncomp bound for the number of connected components (def. 100).
11  %
12  %   See also spectrsort, pspectrsort.
13
14  %   A. Concas, C. Fenu & G. Rodriguez, University of Cagliari, Italy
15  %   Email: concas.anna@gmail.com, kate.fenu@gmail.com, rodriguez@unica.it
16  %
17  %   Last revised Aug 20, 2017
18
19  global AGLOB; % declare as global the input matrix
20  AGLOB = A;
21  AGLOB = abs(AGLOB) | abs(AGLOB');
22
23  if nargin < 2 || isempty(opts), opts = struct('empty','empty'); end
24
25  nam = fieldnames(opts);
26  % max number of connected components
27  if ~any(strcmpi('maxncomp',nam)), opts.maxncomp = 100; end
28
29  n = size(A,1);
```

Figura 2.4: *getconcomp.m è estratta dal pacchetto PQser [7]*

centralnode.m | getconcomp.m | graphvisit.m | prova1.m

```
28
29 -    n = size(A,1);
30 -    root = 1;
31 -    list = root;
32 -    flag = 1;
33 -    i = 0;
34 -    chlist = [];
35 -    U = cell(1,opts.maxncomp); % preallocation
36
37 -    while flag
38 -        i = i+1;
39 -        list = graphvisit(root,list);
40 -        U{i} = list;
41 -        chlist = sort([chlist list]);
42 -        flag = (length(chlist)~= n);
43 -        if flag
44              % root computation in the case of a new connected component
45 -            root = find(diff(chlist)-1,1) + 1;
46 -            if isempty(root) % root restore for a 'complete' sorted chlist
47 -                root = chlist(end) + 1; % (to avoid loops)
48 -            end
49 -            list = [root];
50 -        end
51 -        if ~rem(i,100)
52 -            U{i+opts.maxncomp} = {};
53 -        end
54 -    end
55 -    U = U(1:i); % remove the empty components
56
57
```

centralnode.m | getconcomp.m | graphvisit.m | prova1.m

```
function list = graphvisit(root,list)
%GRAPHVISIT visit a graph starting from a node.
%   list = graphvisit(node) visits a graph defined by an adjacency matrix,
%   starting from node "node". Returns a list of the visited nodes.
%   This function is called by getconcomp.
%   Warning: to reduce stack usage, a global matrix AGLOB is used.
%
%   See also getconcomp.

%   A. Concas, C. Fenu & G. Rodriguez, University of Cagliari, Italy
%   Email: concas.anna@gmail.com, kate.fenu@gmail.com, rodriguez@unica.it
%
%   Last revised Aug 20, 2017

global AGLOB;

l = find(AGLOB(root,:)); % list of the indices of the nodes connected to root
newlist = []; % nodes absent in list

for i = 1:length(l)
    if ~any(l(i)==list)
        list = [list l(i)]; % add every element of l not in the list
        newlist = [newlist l(i)]; % save node in newlist
    end
end

if length(newlist) > 0
    list = sort(list);
    for i = 1:length(newlist)
```

```
    list = sort(list);
    for i = 1:length(newlist)
        list = graphvisit(newlist(i),list);
    end
end
```

Fornendo come paramentro la matrice di adiacenza della rete in figra 2.3, infatti abbiamo come output:

centralnode.m | getconcomp.m | graphvisit.m | prova1.m

```
% matrice di adiacenza di un grafo non orientato
A1 = [0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0
1 0 1 0 1 0 0 1 0 0
0 0 0 1 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0 1 1
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1 0 0];

U = getconcomp(A1)  % componenti connesse
v = U{1}        % nodi nella prima componente connessa
A = A1(v,v)     % matrice di adiacenza della componente connessa
[cnode poscen] = centralnode(A,1)
```

Command Window

```
     0     0     0     0     0     0     0     1     0     0


cnode =

     4


poscen =

    14

fx >>
```

Ovvero il nodo con posizione più centrale è il nodo 4 con P_c=14.

2.5 Esempio completo

Prendiamo in esame la rete in figura 2.5

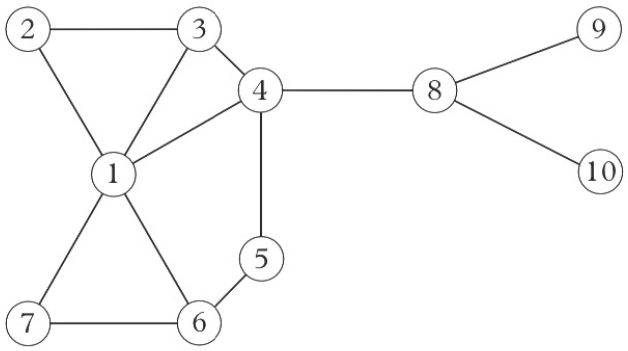


Figura 2.5: *Esempio di rete per la quale calcoliamo le diverse misure di centralità*

Calcoliamo per ogni nodo grado di centralità, closeness centrality e betweenness centrality. La matrice di adiacenza della rete è:

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

dunque il vettore dei gradi di centralità è:

$$k = Ae = \begin{pmatrix} 5 & 2 & 3 & 4 & 2 & 3 & 2 & 3 & 1 & 1 \end{pmatrix}$$

Come viene confermato da MATLAB utilizzando la funzione **centrality** che prende in ingresso il grafo di (attraverso il comando $G = graph(A)$) A e il tipo di centrality che vogliamo calcolare:

```
>> degree=centrality(G, 'degree')

degree =

     5
     2
     3
     4
     2
     3
     2
     3
     1
     1
```

Invece la matrice di distanza è:

$$D = \begin{pmatrix} 0 & 1 & 1 & 1 & 2 & 1 & 1 & 2 & 3 & 3 \\ 1 & 0 & 1 & 2 & 3 & 2 & 2 & 3 & 4 & 4 \\ 1 & 1 & 0 & 1 & 2 & 2 & 2 & 2 & 3 & 3 \\ 1 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 2 & 2 \\ 2 & 3 & 2 & 1 & 0 & 1 & 2 & 2 & 3 & 3 \\ 1 & 2 & 2 & 2 & 1 & 0 & 1 & 3 & 4 & 4 \\ 1 & 2 & 2 & 2 & 2 & 1 & 0 & 3 & 4 & 4 \\ 2 & 3 & 2 & 1 & 2 & 3 & 3 & 0 & 1 & 1 \\ 3 & 4 & 3 & 2 & 3 & 4 & 4 & 1 & 0 & 2 \\ 3 & 4 & 3 & 2 & 3 & 4 & 4 & 1 & 2 & 0 \end{pmatrix}$$

confermato da MATLAB utilizzando l'algoritmo in figura **??**, dove si arriva a A^5, mentre il vettore distanza totale sarà:

$$s = De = \begin{pmatrix} 15 & 22 & 17 & 14 & 19 & 20 & 21 & 18 & 26 & 26 \end{pmatrix}$$

da cui si ottiene quello di closeness centrality:

$$CC(i) = \frac{(n-1)}{s(i)} = \begin{pmatrix} 0.6 & 0.4 & 0.53 & 0.64 & 0.47 & 0.45 & 0.43 & 0.5 & 0.34 & 0.34 \end{pmatrix}$$

In MATLAB non otteniamo esattamente lo stesso vettore, perchè l'algoritmo usato

calcola la closeness centrality come: $CC(i) = \frac{1}{s(i)}$, senza tenere quindi in considerazione la grandezza della rete data dal fattore $(n-1)$, infatti si ha:

```
>> closeness=centrality(G, 'closeness')

closeness =

    0.0667
    0.0455
    0.0588
    0.0714
    0.0526
    0.0500
    0.0476
    0.0556
    0.0385
    0.0385
```

Figura 2.6: *Closeness centrality senza tenere in considerazione l'estensione della rete*

In ogni caso si preferisce usare una misura "pesata". come appunto quella che si ottiene dalla nostra definizione, in quanto ci fornisce un'interpretazione completa della centralità del nodo, tenendo in considerazione anche l'estensione della rete. Le due definizioni coincidono quando si moltiplica il vettore ottenuto con MATLAB per $(n-1)$, ovvero il numero totale dei nodi della rete diminuito di uno, infatti otteniamo lo stesso vettore $CC(i)$ che abbiamo calcolato:

```
>> closeness=closeness*9

closeness =

    0.6000
    0.4091
    0.5294
    0.6429
    0.4737
    0.4500
    0.4286
    0.5000
    0.3462
    0.3462
```

Figura 2.7: *Closeness centrality tenendo in considerazione l'estensione della rete*

Continuando con l'analisi, il vettore di betweenness centrality è:

$$BC = \begin{pmatrix} 12.67 & 0 & 2.33 & 20.17 & 2 & 1.83 & 0 & 15 & 0 & 0 \end{pmatrix}$$

```
>> betweenness=centrality(G, 'betweenness')

betweenness =

   12.6667
         0
    2.3333
   20.1667
    2.0000
    1.8333
         0
   15.0000
         0
         0
```

Riassumendo i dati in forma di tabella:

nodo	grado di centralità	closeness	betweenness
1	***5***	0.6	12.67
2	2	0.4	0
3	3	0.53	2.33
4	4	**0.64**	**20.17**
5	2	0.47	2
6	3	0.45	1.83
7	2	0.39	0
8	3	0.5	15
9	1	0.34	0
10	1	0.34	0

Si può notare quindi che in questo caso grado di centralità, closeness e betweenness non coincidono, infatti sebbene il nodo 1 sia quello con grado di centralità più alto, non è quello con closeness e betweenness centrality più alte, che è invece il nodo 4. Quindi da questo risultato si può dedurre che il nodo 1 è quello maggiormente connesso *direttamente*, ma il nodo 4 è quello più vicino agli altri e che svolge da mediatore tra diverse parti nella rete. Quindi corrisponde a quello che complessivamente riesce a comunicare meglio con gli altri nodi della rete, infatti utilizzando il codice MATLAB mostrato prima per determinare il nodo con posizione più centrale nella rete, abbiamo che questo corrisponde al nodo 4 con valore P_c=14.

Capitolo 3

Centralità spettrale

Fino ad ora abbiamo esaminato misure di centralità che tengono conto solo dell'immediata vicinanza tra vari nodi di una rete, ma per un'analisi completa può essere utile andare oltre questo concetto, studiando il ruolo di un nodo in relazione all'intera rete, e per farlo sfruttiamo quelle che vengono chiamate proprietà spettrali di una rete.

3.1 Katz centrality

Il grado di centralità k_i di un nodo i indica il numero di cammini di lunghezza unitaria dal nodo i a tutti gli altri della rete. Katz estese questo concetto andando a contare non solo i cammini di lunghezza unitaria, ma ogni cammino che ha origine dal nodo i. Possiamo intuire che i nodi più vicini hanno maggiore influenza sul nodo i rispetto a quelli più distanti, perciò, combinando cammini di ogni lunghezza con un *fattore di attenuazione* α, in modo che sia data più importanza ai cammini brevi, possiamo definire l' **indice di Katz** come:

$$K_i = [(\alpha^0 A^0 + \alpha A + \alpha^2 A^2 + \cdots + \alpha^k A^k + \dots)e]_i = \left(\sum_{k=0}^{\infty} (\alpha^k A^k)e\right)_i \text{, dove } A^0 = I$$

che sfrutta il fatto che gli elementi a_{ij} di A^k siano il *numero di cammini* di lunghezza k tra i nodi i e i nodi j. Inoltre $f(A) = \sum_{k=0}^{\infty} (\alpha^k A^k)$ corrisponde ad una *media pesata* di tutti i cammini che connettono i a j, e descrive la facilità con cui è possibile viaggiare tra essi.

La serie nell'espressione ha come funzione risolvente $(I - \alpha A)^{-1}$ e sappiamo

che essa converge se $\alpha < \frac{1}{\rho(A)}$ [1].In tal caso allora l'indice di Katz può essere scritto come:

$$K_i = [(I - \alpha A)^{-1} e]_i$$

Oppure possiamo esprimerlo in termini di *autovalori* e *autovettori* [2] della matrice di adiacenza, sfruttando la fattorizzazione spettrale $A = QDQ^T$:

$$K_i = \sum_l \sum_j q_j(i) q_j(l) \frac{1}{1 - \alpha\lambda_j}$$

Nel caso di **reti orientate** dovremo considerare il numero di archi in ingresso e in uscita dal nodo e l'indice di Katz diventa:

$$K_i^{out} = [(I - \alpha A)^{-1} e]_i, \quad K_i^{in} = [e^T (I - \alpha A)^{-1}]_i$$

3.2 Eigenvector centrality

L'intuizione che sta dietro questa misura di centralità è che un nodo è tanto più importante quanto più è collegato ad altri nodi altrettanto importanti all'interno della rete. Viene definita come:

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} a_{ij} x_j$$

Preso un sistema lineare $Ax = \lambda x$, con x autovettore (*eigenvector*) e λ autovalore, dato che $a_{ij} \geq 0$ e vogliamo far sì che anche $x_i \geq 0$, in modo che sia normalizzato ovvero di lunghezza unitaria, perciò il teorema di Perron-Frobenius [3] dimostra che l'unico autovalore possibile è quello di valore massimo.
Nel caso di **rete orientata**, usiamo gli *autovettori sinistri e destri.* Se $Ax = \lambda_1 x$ e $A^T y = \lambda_{max} y$ allora gli elementi di x e y forniscono rispettivamente le eigenvector

[1] la serie geometrica $\sum_{k=0}^{\infty} a^k = \frac{1}{1-a}$ per $|a| < 1$. Allo stesso modo la serie $\sum_{k=0}^{\infty} \alpha^k A^k = (I - \alpha A)^{-1}$, perchè la condizione di convergenza equivale a $\|\alpha A\| < 1$, ovvero $\|\alpha A\|_2 = \alpha \|A\|_2 = \alpha\rho(A) < 1$, se A è Hermitana ovvero $A = A^T$, da cui si ottiene $\alpha < \frac{1}{\rho(A)}$

[2] si definiscono rispettivamente autovalore e autovettore di una matrice A, uno scalare $\lambda \in \mathbb{C}$ ed un vettore $x \neq 0$ che verifichino la relazione $Ax = \lambda x$.

[3] la matrice di adiacenza è non negativa e supponendo che $A \in \mathbb{R}^{nxn}$, allora A ha autovalori λ che soddisfano la proprietà:

- se $Ay = \lambda y$ allora $y = \alpha x$ dove $x > 0$ e $\alpha \in \mathbb{C}$

centrality: quella destra descrive l'importanza di un nodo in base ai nodi a cui invia le informazioni, mentre quella sinistra descrive l'importanza di un nodo sulla base dei nodi da cui riceve le informazioni.

3.3 PageRank centrality

Quando effettuiamo una ricerca su internet, il meccanismo che si occupa di trovare i risultati fa in modo di mettere in cima alla nostra pagina i siti web che coincidono con i nostri parametri di ricerca. Possiamo vedere il World Wide Web come una vasta rete orientata, dove i nodi sono le *pagine web* e gli archi sono gli *hyperlinks* tra esse.
Il PageRank è strettamente correlato alla eigenvector centrality, e misura l'importanza di una pagina web in base all'importanza delle altre pagine che indirizzano ad essa. In altri termini quindi, il PageRank di una pagina non è altro che la somma delle PageRank centrality di tutte le pagine che puntano ad essa.
Dato che il WWW è una rete estremamente complessa ed è nota per essere disconnessa, per poter applicare degli strumenti matematici come il teorema di Perron-Frobenius, dobbiamo fare delle modifiche alla matrice di adiacenza A. Quella più semplice è riconnettere i nodi che non hanno collegamenti in uscita, quindi rimpiazziamo la matrice A con una nuova matrice H definita come:

$$H_{ij} = \begin{cases} a_{ij}, & k_i^{out} > 0 \\ 1 & k_i^{out} = 0 \end{cases}$$

Il PageRank usa come modello un utente che naviga in rete e che raggiunge il sito deisgnato dopo diversi click, per poi spostarsi su una pagina casuale. Il valore di PageRank di una certa pagina riflette la probabilità che un utente arrivi su di essa, cliccando su un link di qualche altra pagina. Se l'utente arriva su una pagina che non ha link esterni (detta *sink page*), allora attraverso un URL casuale verrà reindirizzato, in modo da continuare la navigazione.
La frequenza relativa di visite ad una pagina, può essere misurata dagli elementi del principale autovettore sinistro della *matrice stocastica*:

$$S = D^{-1}H$$

dove D è una matrice diagonale contenente gli out-degrees k_i^{out} (grado di centralità per una rete orientata) della rete che ha come matrice di adiacenza H.
Dato che però calcolare gli autovettori di una rete grande come il WWW è mol-

to complicato, utilizziamo un parametro addizionale α e anzichè lavorare con la matrice S, lavoriamo con:

$$P = \alpha S + \frac{1-\alpha}{n} e e^T$$

Capitolo 4

Esempi reti reali

Prendiamo in esame degli esempi di reti non orientate reali di diverse dimensioni:

- *autobahn* (1168 nodi): descrive il sistema autostradale tedesco, dove i nodi rappresentano delle località e gli archi sono le strade che le connettono. Equivale ad una rete reale di *piccole dimensioni*;
- *yeast* (2114 nodi): descrive la rete di interazione delle proteine del lievito: ogni arco rappresenta l'interazione tra due proteine. Equivale ad una rete reale di *medie dimensioni*;
- *facebook* (63 731 nodi): descrive tutte le connessioni tra utenti (*amicizie*) di New Orleans. Equivale ad una rete reale di *grandi dimensioni.*

Di ciascuna calcoliamo le 4 diverse misure di centralità (grado, closeness, betweenness e position) individuando il nodo più centrale relativo alla singola misura e inoltre mostriamo il grafico della rete con una colorazione dei nodi proporzionale alla misura di centralità.

4.1 Autobahn

Attraverso il comando *spy* di MATLAB siamo in grado di visualizzare la sparsità della matrice.
Come si può vedere dall'immagine sotto 4.1, abbiamo che la maggior parte degli elementi non nulli sta sulla diagonale della matrice e inoltre il numero di elementi non nulli nnz=2486, mentre il numero di elementi totali è pari a 1364224, dunque la densità della matrice è pari a:

$$\text{densità} = \frac{nnz}{tot} = \frac{2486}{1364224} = 0.0018 << 1$$

perciò la matrice sarà sparsa.

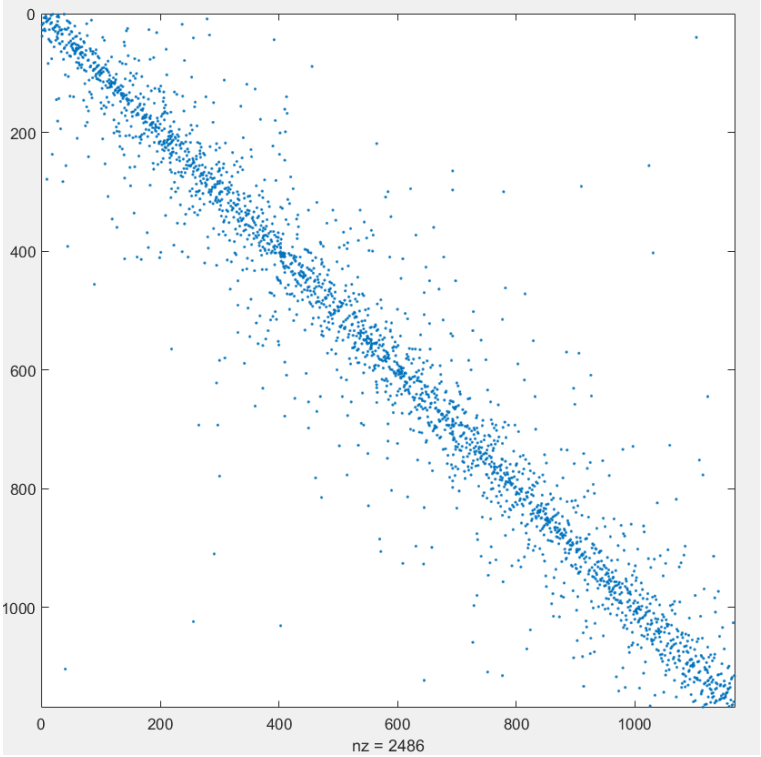
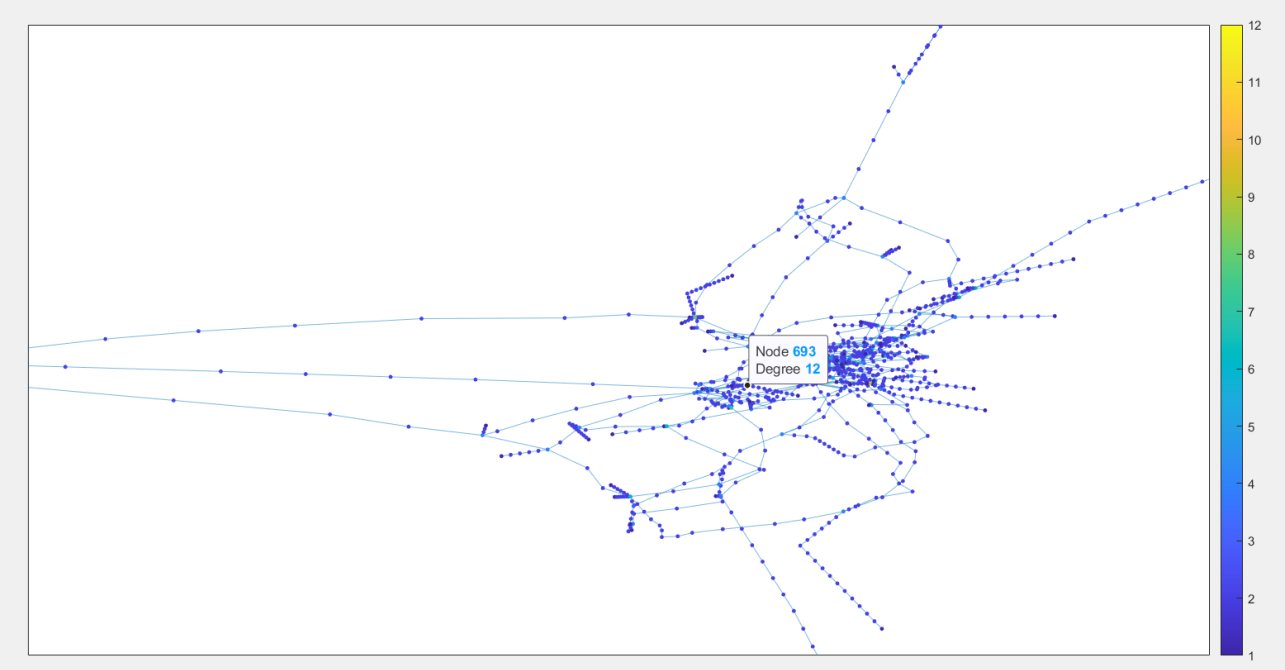


Figura 4.1: *Matrice di adiacenza della rete autostradale tedesca*

Calcoliamo il grado di centralità e individuiamo il valore massimo possibile e l'indice del nodo a cui appartiene:

```
>> [max_val, pos_idx]=max(degree)

max_val =

    12


pos_idx =

   693
```

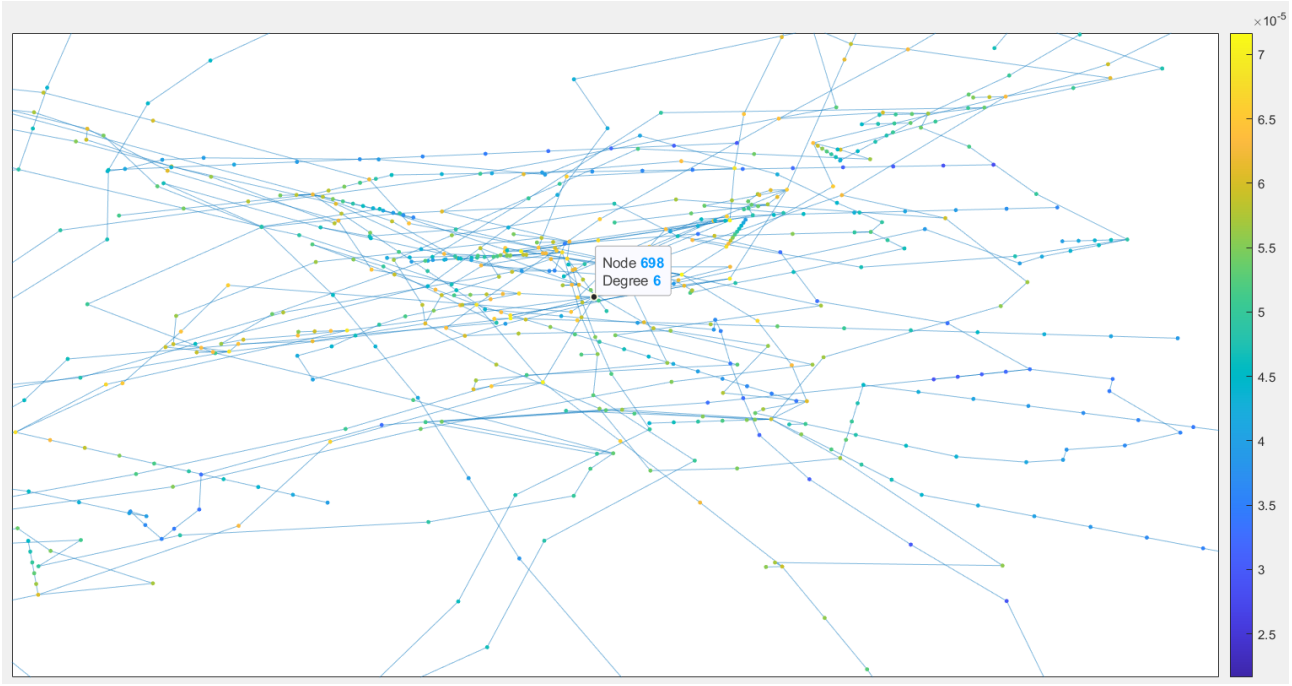
Individuiamo poi il nodo con grado di centralità maggiore nella rete:



Calcoliamo poi la closeness centrality e individuiamo il nodo con valore maggiore:

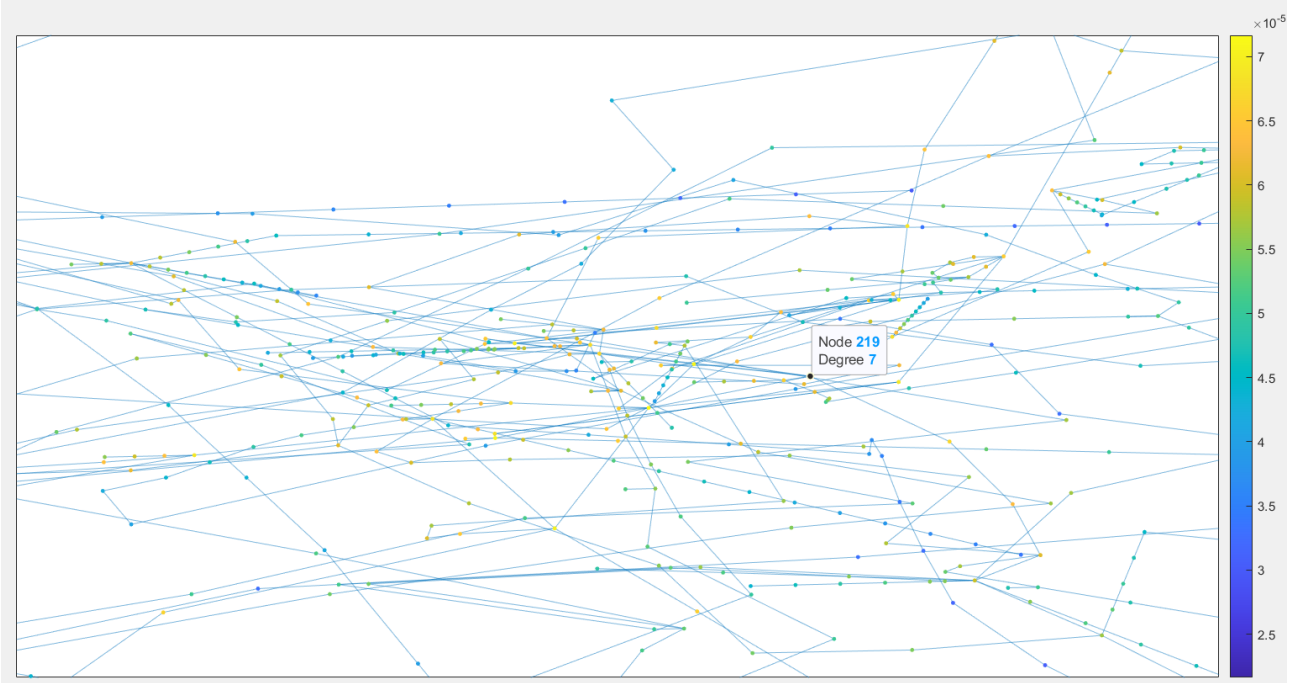
```
>> [max_val, pos_idx]=max(clos)

max_val =

   7.1664e-05


pos_idx =

   698
```

Individuiamolo sempre nella rete grazie alla colorbar:



Facciamo la stessa cosa con la betweenness centrality:

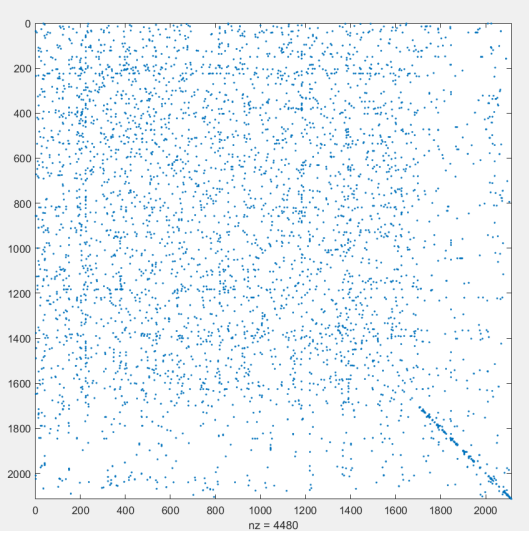
```
>> [max_val, pos_idx]=max(betw)

max_val =

   1.6311e+05


pos_idx =

   219
```



4.2 Yeast

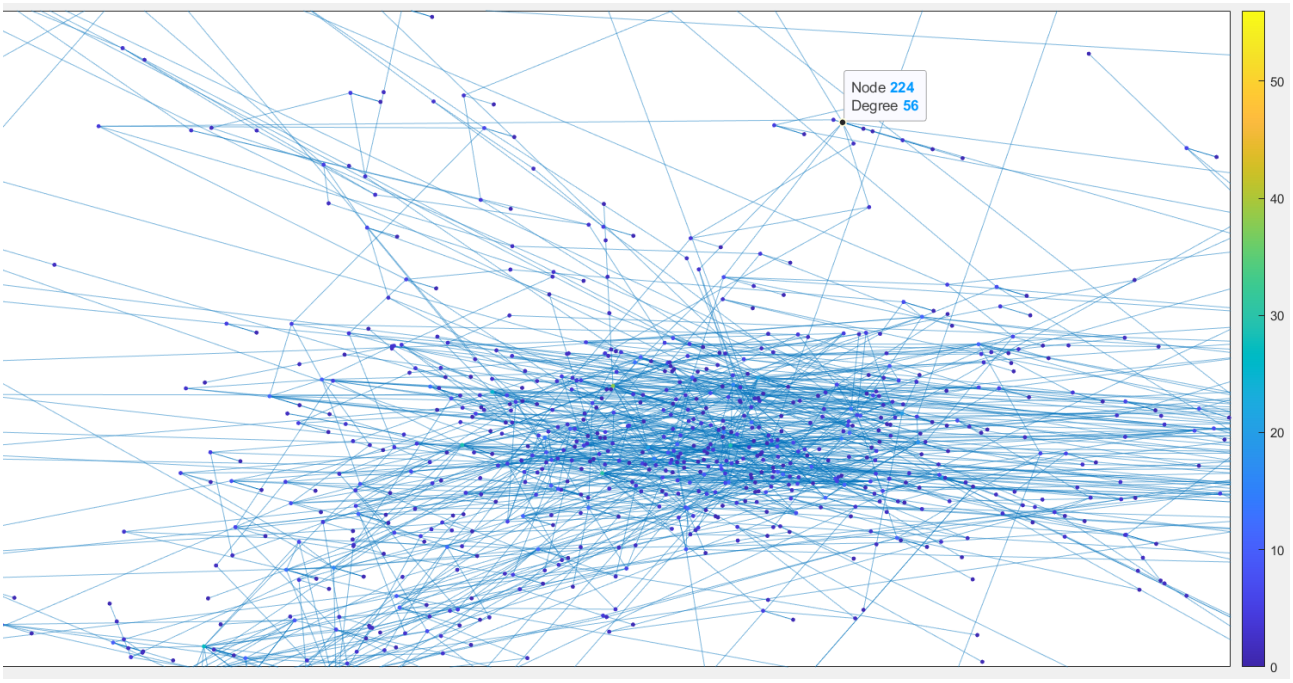
Anche in questo caso abbiamo una matrice sparsa dato che:

$$\text{densità} = \frac{4480}{2114^2} = 0,001 << 1$$



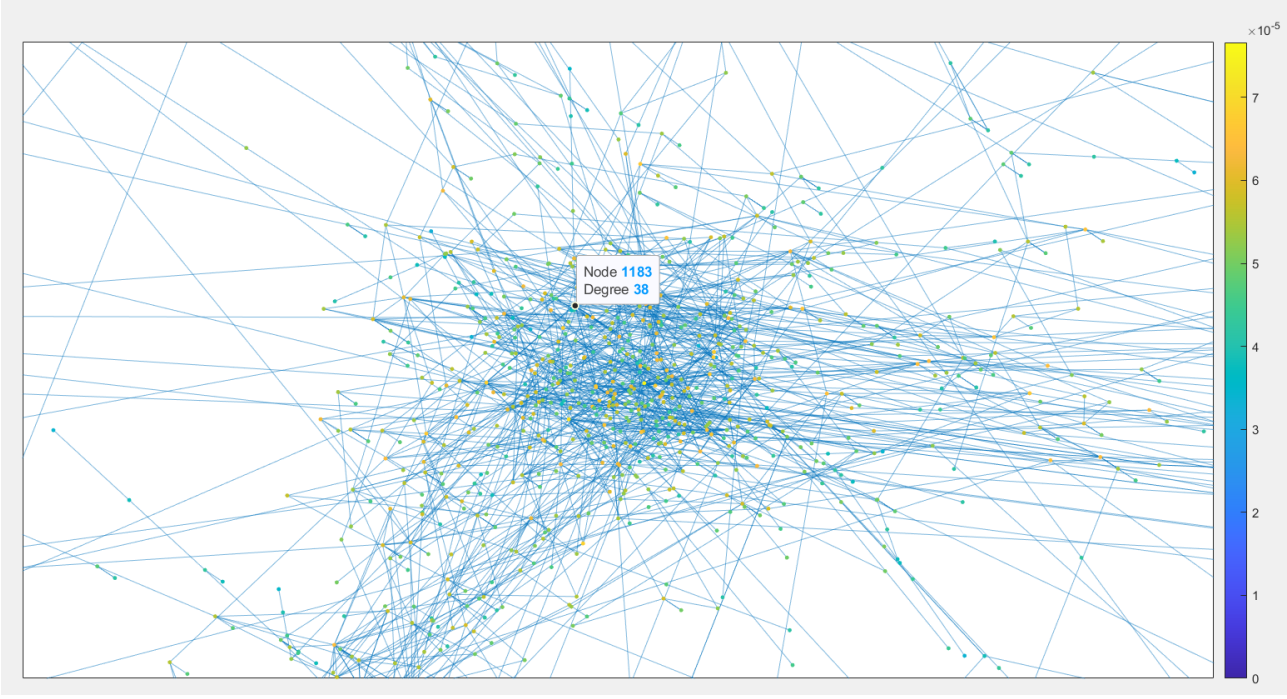
Calcoliamo il grado di centralità e individuiamo il nodo con valore maggiore:

```
>> [max_val,pos_idx]=max(degree)

max_val =

    56


pos_idx =

   224

>>
```



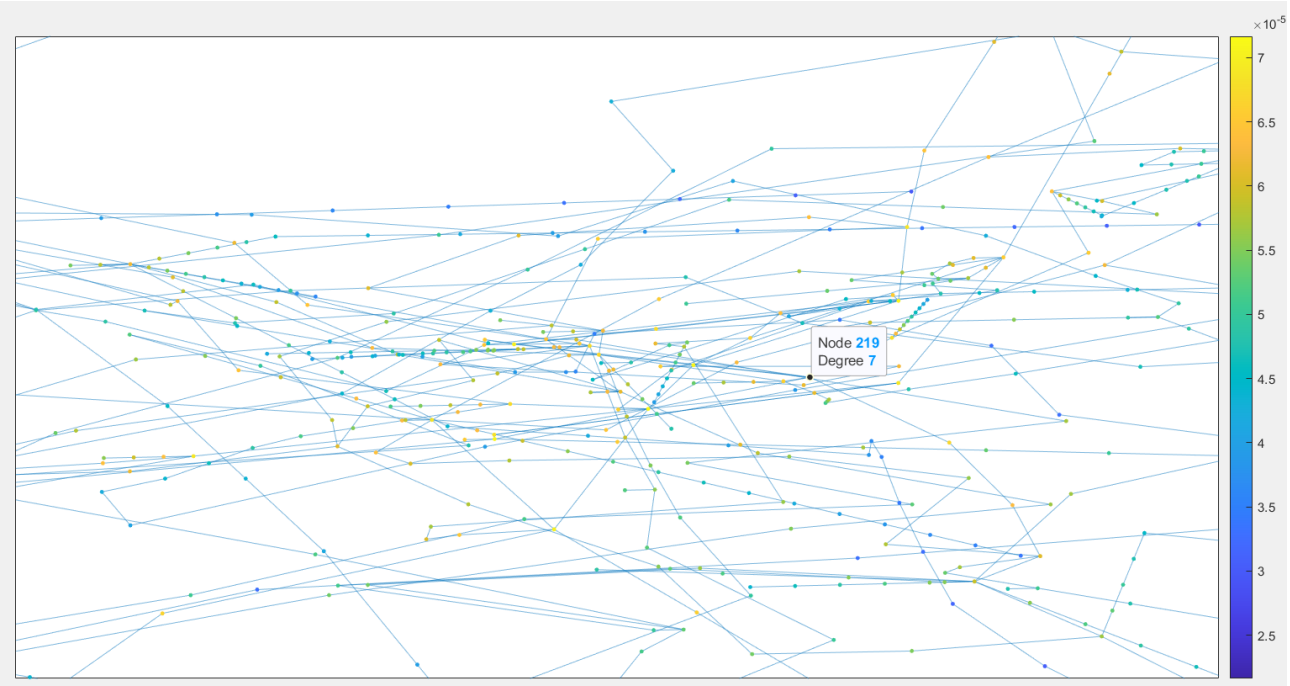
Calcoliamo la closeness centrality e individuiamo il nodo con valore maggiore:

```
>> [max_val, pos_idx]=max(clos)

max_val =

   7.6614e-05


pos_idx =

        1183
```



Infine facciamo la stessa cosa per la betweenness centrality:

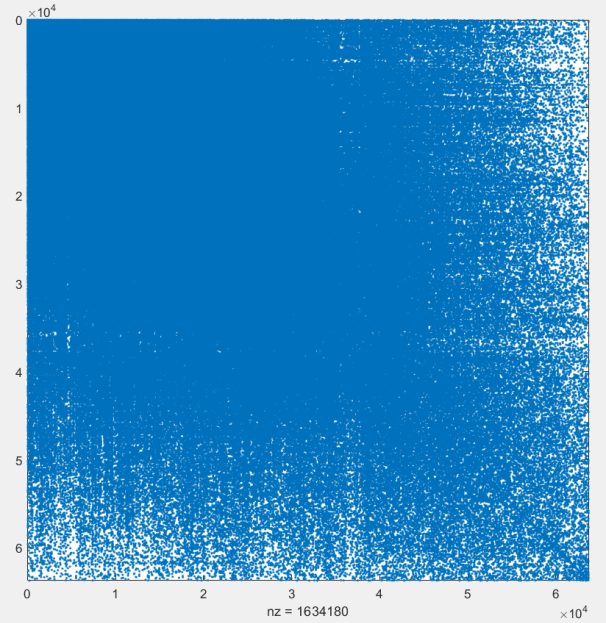
```
>> [max_val, pos_idx]=max(betw)

max_val =

   1.6311e+05


pos_idx =

   219
```



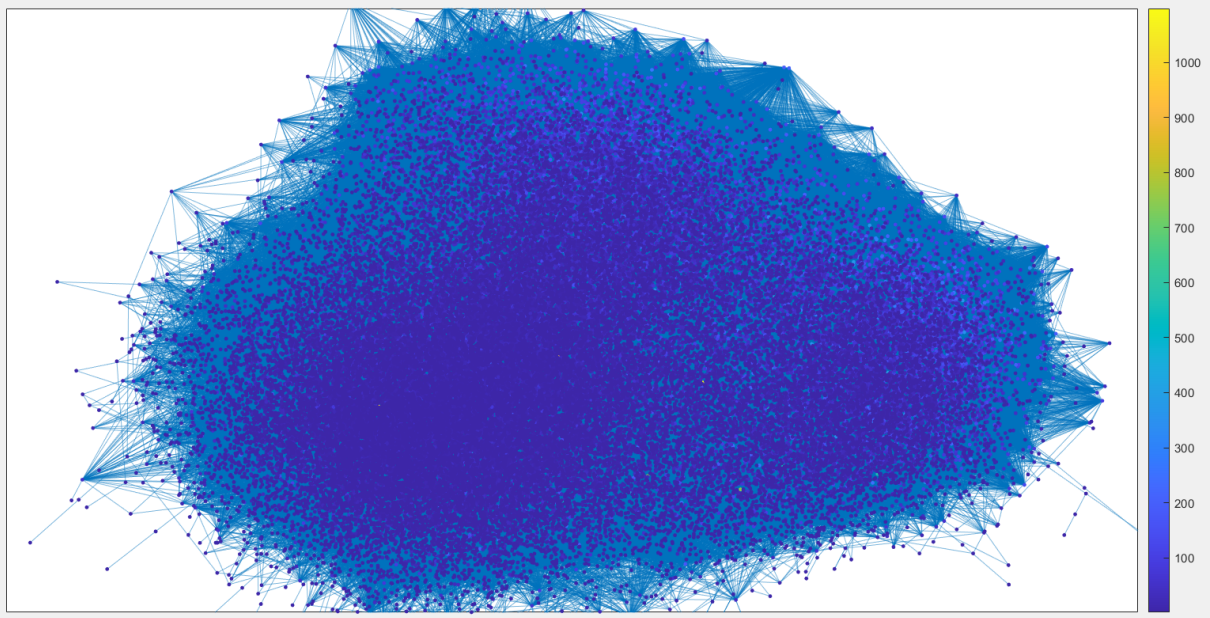
4.3 Facebook

Abbiamo sempre una matrice sparsa dato che:

$$\text{densità} : \frac{1634180}{63731^2} << 1$$



Essendo una rete molto grande, calcoleremo solo i valori massimi delle varie centralità, senza andare ad individuare i nodi nella rete.



Per il grado di centralità si ha massimo valore:

```
max_val =

        1098


pos_idx =

        2332
```

Per la closeness centrality si ha:

```
>> [max_val, pos_idx]=max(clos)

max_val =

   5.5696e-06


pos_idx =

        2332
```

E per la betweenness centrality si ha:

```
>> [max_val, pos_idx]=max(betw)

max_val =

   6.7151e+07


pos_idx =

   554
```

Dato che il nodo con indice 2332 ha sia grado di centralità che closeness centrality maggiore, è probabile che sia il nodo più centrale complessivamente della rete.
Per riassumere tutte le nostre analisi in maniera ordinata indicando anche il tempo impiegato da MATLAB per calcolare ciascun valore (comandi: *tic; istruzione; toc;*), abbiamo;

rete	degree	closeness	betweenness	position
Autobahn	12 $(0.0045s)$	$7.16x10^{-5}$ $(0.0293s)$	$1.63x10^{5}$ $(0.0290s)$	14 036 $(3.62s)$
Yeast	56 $(0.0016s)$	$7.66x10^{-5}$ $(0.015s)$	$1.63x10^{5}$ $(0.034s)$	7252* $(9.57s)$
Facebook	1098 $(0.023s)$	$5.56x10^{-6}$ $(233s)$	$6.71x10^{7}$ $(932s)$	//

* Per calcolare la position centrality della Yeast non abbiamo considerato la prima componente connessa (come è scritto nel codice MATLAB) ma quella più grande.

Si vede subito dalla tabella che per calcolare le varie misure di centralità che per una rete molto grande come quella di Facebook (da oltre 63 000 nodi), i tempi di calcolo sono molto maggiori: se per la degree è 5-10 volte più grande, nel caso della closeness e betweenness non ha nemmeno senso fare un paragone perchè si tratta non di centesimi di secondo, come nei primi due casi, ma di diversi minuti. Nel caso della position, addirittura, non abbiamo un tempo di calcolo o un valore, in quanto la rete è troppo grande.

Capitolo 5

Analisi Rete CTM

La matrice di adiacenza della rete autobus di Cagliari ovviamente non è simemtrica, in quanto non tutte le fermate hanno sia l'andata che il ritorno della stessa linea bus (che tradotto in nodi e archi significa che alcuni nodi hanno solo archi orientati in uscita o in ingresso). Perciò la rete in realtà è orientata e avrà una matrice di adiacenza non simmetrica, ma per semplificare le analisi, almeno inizialmente, rendiamola simmetrica. Inoltre abbiamo che essa è formata da $970x970$ elementi, perciò leggeremente più piccola della rete Autobahn vista in precedenza, e possiede l'attributo *sparse*, infatti attraverso il comando *spy* è possibile notare la classica struttura di una matrice sparsa in figura 5.1.
Le informazioni in nostro possesso sono:

- *route*: elenco linee di 7 campi che contiene in ordine id,codice linea ctm, nome, colore, codice linea, disponibile e prenotabile;
- *mbusline*: altro elenco linee di 9 campi che contiene id, andata/ritorno, codice linea ctm, colore, descrizione, id nome codice, ordine, linea percorso;
- *mroutepoint*: rotta di ciascuna linea di 4 campi con id, point id, id linea bus, stop order;
- *fermate*: contiene l'elenco delle fermate di 9 campi con id, indirizzo, bus stop ctm, descrizione, latitudine, longitudine, tipo, starred, fermata disabili.

Vogliamo determinare quali siano le fermate principali della rete calcolando le varie misure di centralità:

- **degree**: il nodo con grado di centralità maggiore è quello con indice 884, che nella tabella "fermate" corrisponde alla fermata *Abruzzi (ang.via Campeda)*

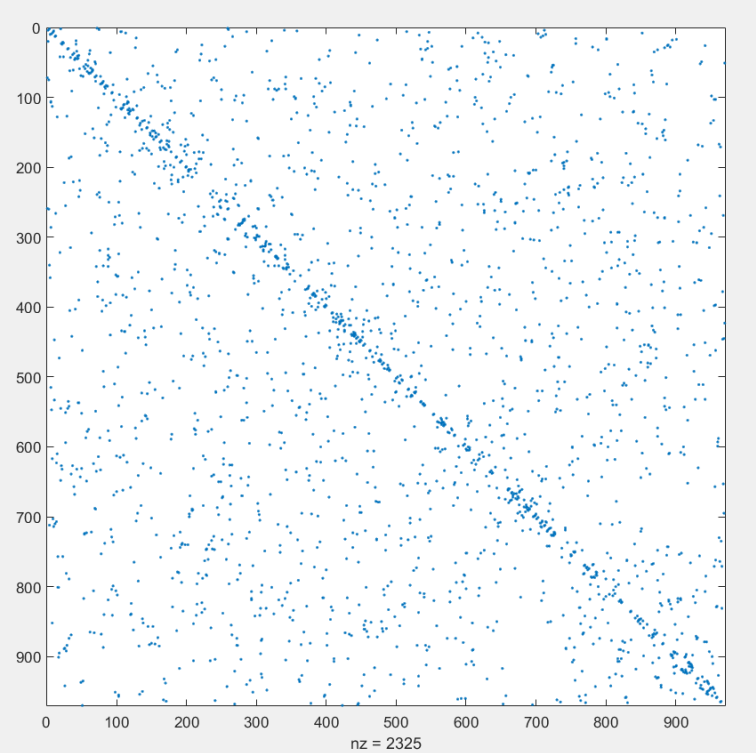


Figura 5.1: *Output del comando spy che permette di vedere la densità della matrice di adiacenza.*

mroutepoint | mbusline | fermate | route

970x9 cell

	1	2	3	4	5	6	7	8	9
881	881	'Campania ...	'CA0085'	'Campania ...	39.2336	9.1118	'BUSTOP'	'*'	1
882	882	'Campania ...	'CA0083'	'Campania ...	39.2323	9.1139	'BUSTOP'	'*'	1
883	883	'Campania ...	'CA0082'	'Campania ...	39.2325	9.1138	'BUSTOP'	'*'	2
884	884	'Abruzzi (a...	'AB0109'	'Abruzzi (a...	39.2405	9.1007	'BUSTOP'	'*'	1
885	885	'Binaghi (a...	'AU1142'	'Binaghi (a...	39.2543	9.1004	'BUSTOP'	'*'	1
886	886	'San Salvat...	'SS0946'	'San Salvat...	39.2582	9.1700	'BUSTOP'	'*'	1
887	887	'Diaz (uffici...	'AD0265'	'Diaz (uffici...	39.2102	9.1196	'BUSTOP'	'*'	1
888	888	'Giorgino (...	'GN2099'	'Giorgino (...	39.2083	9.0847	'BUSTOP'	'*'	0
889	889	'Giorgino (t...	'GN2098'	'Giorgino (t...	39.2083	9.0845	'BUSTOP'	'*'	0
890	890	'Yenne (fro...	'PY0056'	'Yenne (fro...	39.2180	9.1136	'BUSTOP'	'*'	0
891	891	'Pitz" e Serr...	'PZ0996'	'Pitz" e Serr...	39.2405	9.2191	'BUSTOP'	'*'	0
892	892	'Mazzini (p...	'MZ0670'	'Mazzini (p...	39.2539	9.1719	'BUSTOP'	'*'	0
893	893	'Margine R...	'MR0409'	'Margine R...	39.2266	9.2385	'BUSTOP'	'*'	0

Ricordiamo che il **grado di centralità** (*degree*) misura quanto un nodo riesce a comunicare **direttamente** con gli altri nodi, ovvero il numero di cammini di lunghezza unitaria. Abbiamo determinato ora che la fermata Abruzzi è quella con misura di valore maggiore, con *degree*=7, infatti dall'immagine 5.2 vediamo che il nodo 884 comunica direttamente con i nodi di indice:

- 29: fermata Lunigiana (Hotel Sardegna);
- 272: San Michele (Chiesa);
- 401: Marongiu (fronte San Michele);
- 530: del Fangario (ang. viale Elmas);
- 542: Emilia (ang. via Mandrolisai);
- 869: Cornalias (ang. via Perugino);
- 921: Cornalias (chiesa).

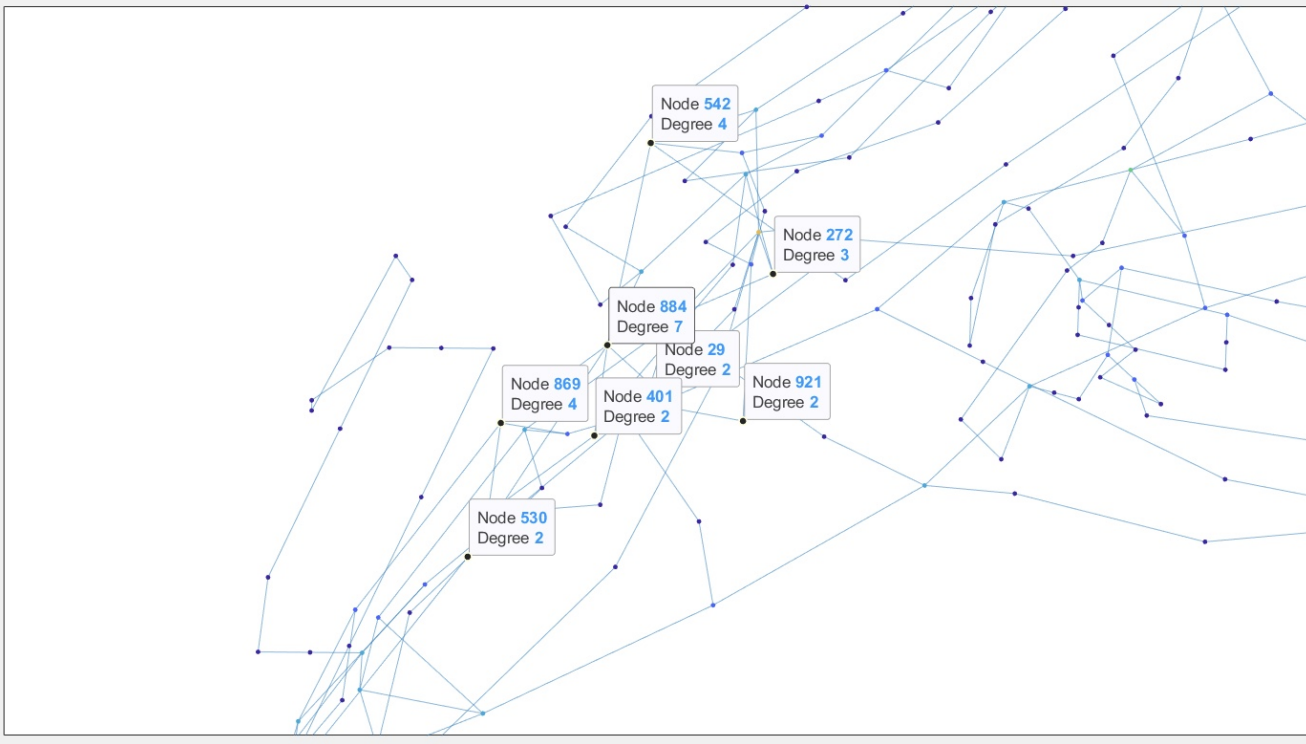
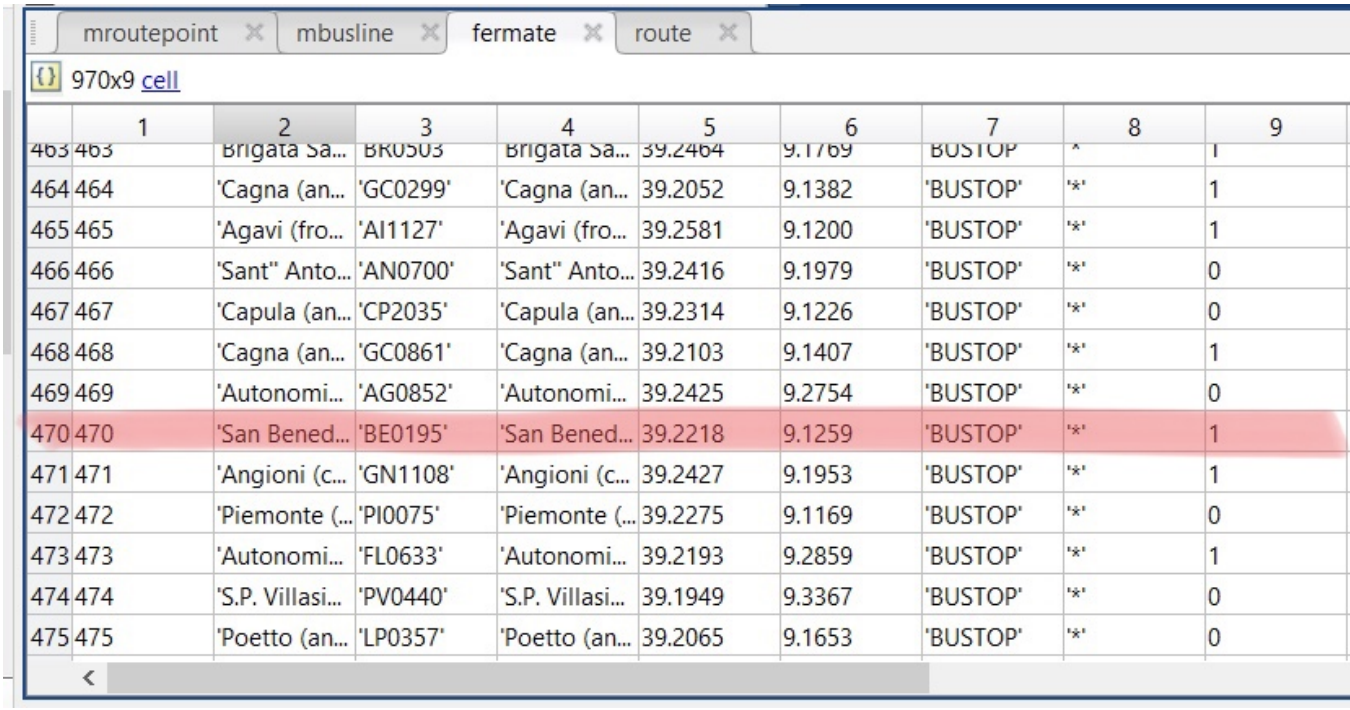


Figura 5.2: *Zoom del grafo che si ottiene su MATLAB che mette in evidenza i collegamenti diretti tra il nodo 884 e gli altri nodi.*



mroutepoint | mbusline | fermate | route

970x9 cell

	1	2	3	4	5	6	7	8	9
463	463	Brigata Sa...	BR0503	Brigata Sa...	39.2464	9.1769	BUSTOP	*	1
464	464	'Cagna (an...	'GC0299'	'Cagna (an...	39.2052	9.1382	'BUSTOP'	'*'	1
465	465	'Agavi (fro...	'AI1127'	'Agavi (fro...	39.2581	9.1200	'BUSTOP'	'*'	1
466	466	'Sant" Anto...	'AN0700'	'Sant" Anto...	39.2416	9.1979	'BUSTOP'	'*'	0
467	467	'Capula (an...	'CP2035'	'Capula (an...	39.2314	9.1226	'BUSTOP'	'*'	0
468	468	'Cagna (an...	'GC0861'	'Cagna (an...	39.2103	9.1407	'BUSTOP'	'*'	1
469	469	'Autonomi...	'AG0852'	'Autonomi...	39.2425	9.2754	'BUSTOP'	'*'	0
470	470	'San Bened...	'BE0195'	'San Bened...	39.2218	9.1259	'BUSTOP'	'*'	1
471	471	'Angioni (c...	'GN1108'	'Angioni (c...	39.2427	9.1953	'BUSTOP'	'*'	1
472	472	'Piemonte (...	'PI0075'	'Piemonte (...	39.2275	9.1169	'BUSTOP'	'*'	0
473	473	'Autonomi...	'FL0633'	'Autonomi...	39.2193	9.2859	'BUSTOP'	'*'	1
474	474	'S.P. Villasi...	'PV0440'	'S.P. Villasi...	39.1949	9.3367	'BUSTOP'	'*'	0
475	475	'Poetto (an...	'LP0357'	'Poetto (an...	39.2065	9.1653	'BUSTOP'	'*'	0

- **closeness**: il nodo con closeness centrality maggiore ha indice 470 e corrisponde alla fermata *San Benedetto (Buon Pastore)*;
  La **closeness centrality** invece misurava quanto un nodo è *vicino* agli altri nodi della rete. Abbiamo determinato che nel caso della rete CTM quello con misura maggiore è la fermata San Benedetto (Buon Pastore), con $closeness = 6,5x10^{-5}$, infatti dalla figura 5.3 e utilizzando la legenda delle linee in figura 5.4 si vede che questa fermata è toccata direttamente dalle linee 1, 29, 30, 31 e inoltre è vicina alle fermate delle linee 6 e 3.

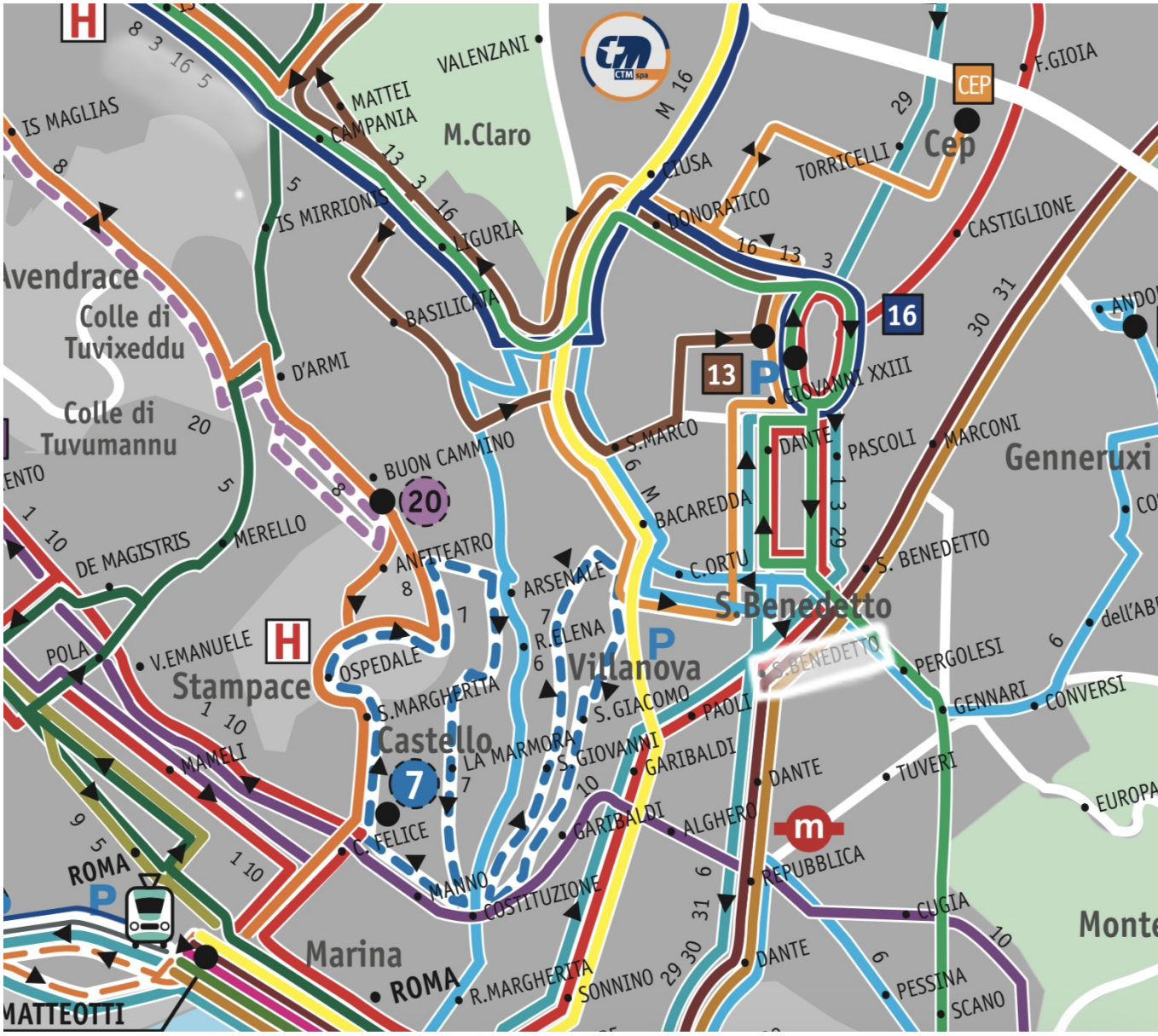


Figura 5.3: *La fermata San benedetto è nel riquadro bianco.*

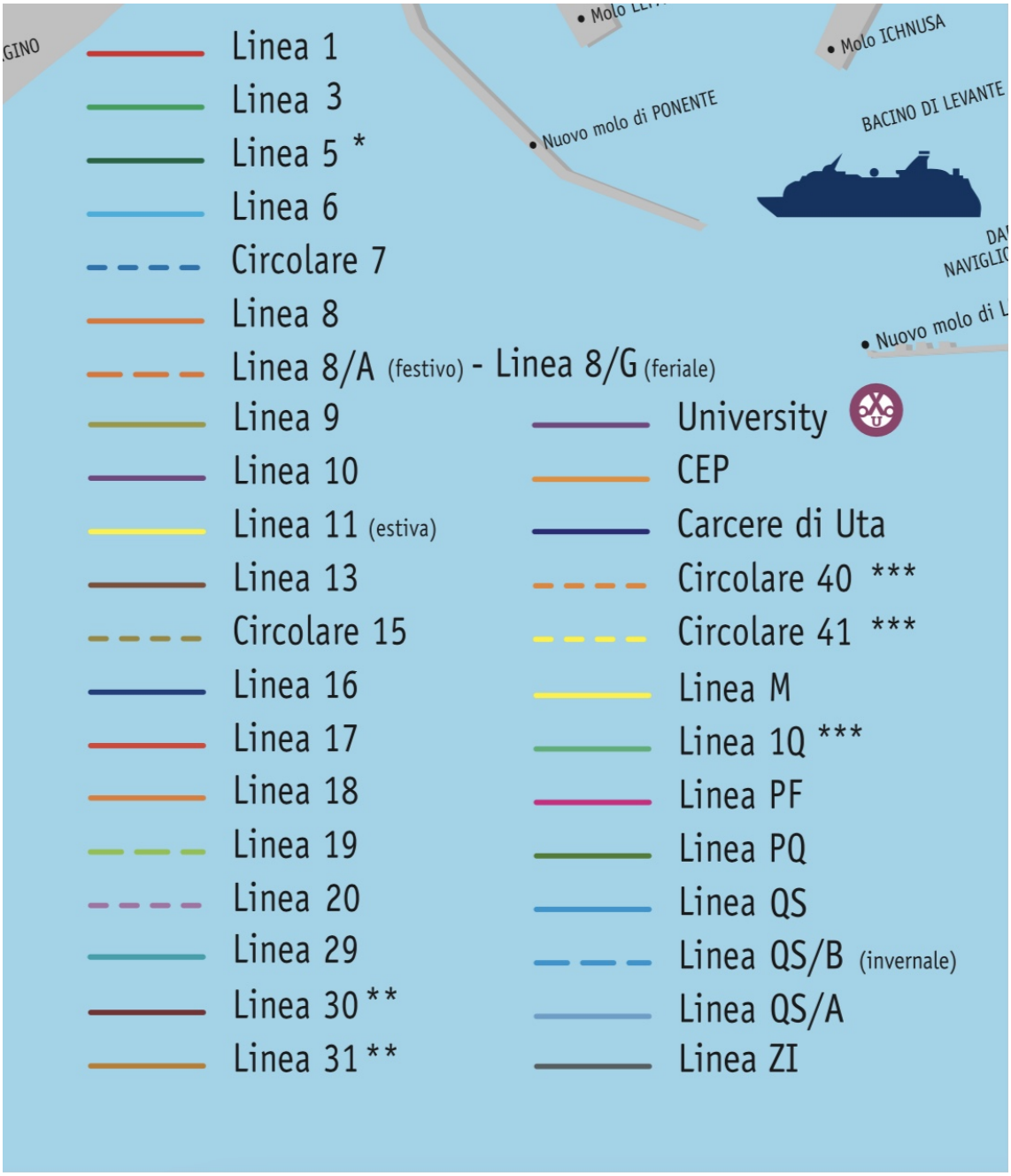
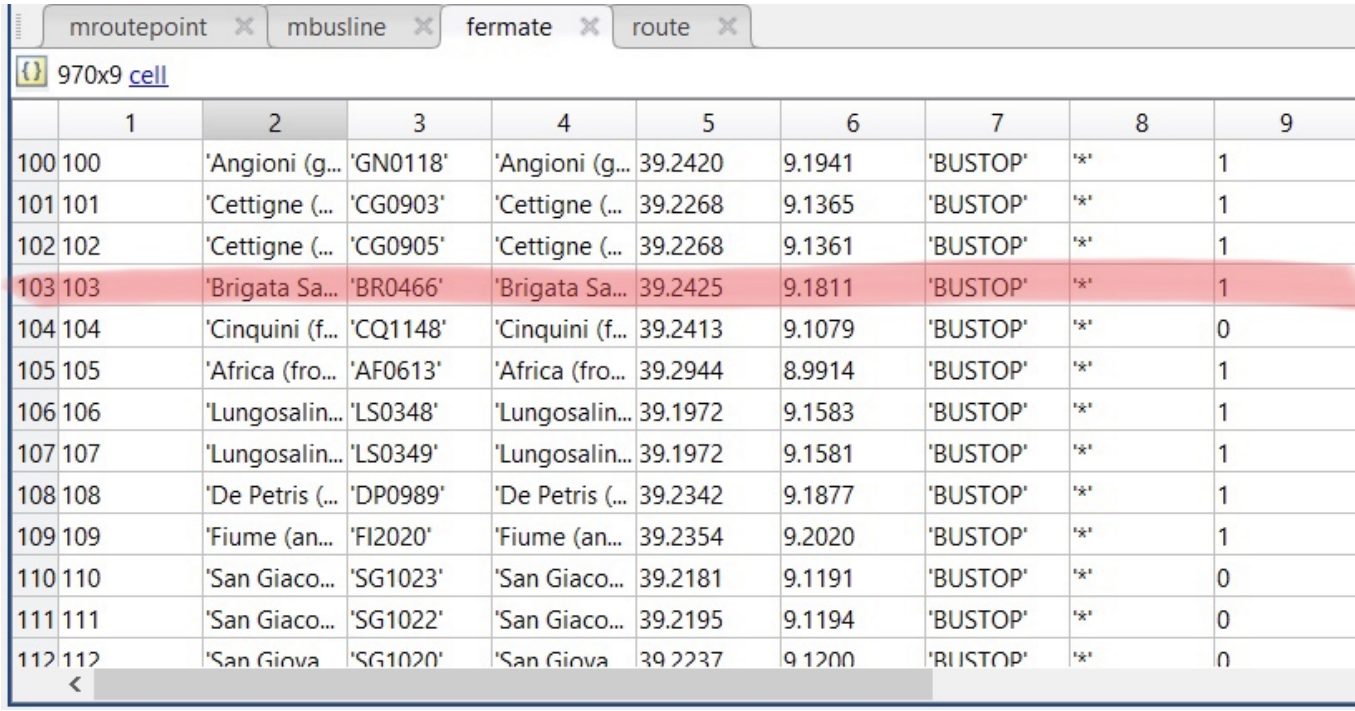


Figura 5.4: *Legenda delle varie lineee CTM.*

- **betweenness**: il nodo con centralità maggiore è quello con indice 103, che corrisponde alla fermata *Brigata Sassari (CTM Point)*, con *betweenness* = $9,27x10^4$.



mroutepoint | mbusline | fermate | route

970x9 cell

	1	2	3	4	5	6	7	8	9
100	100	'Angioni (g...	'GN0118'	'Angioni (g...	39.2420	9.1941	'BUSTOP'	'*'	1
101	101	'Cettigne (...	'CG0903'	'Cettigne (...	39.2268	9.1365	'BUSTOP'	'*'	1
102	102	'Cettigne (...	'CG0905'	'Cettigne (...	39.2268	9.1361	'BUSTOP'	'*'	1
103	103	'Brigata Sa...	'BR0466'	'Brigata Sa...	39.2425	9.1811	'BUSTOP'	'*'	1
104	104	'Cinquini (f...	'CQ1148'	'Cinquini (f...	39.2413	9.1079	'BUSTOP'	'*'	0
105	105	'Africa (fro...	'AF0613'	'Africa (fro...	39.2944	8.9914	'BUSTOP'	'*'	1
106	106	'Lungosalin...	'LS0348'	'Lungosalin...	39.1972	9.1583	'BUSTOP'	'*'	1
107	107	'Lungosalin...	'LS0349'	'Lungosalin...	39.1972	9.1581	'BUSTOP'	'*'	1
108	108	'De Petris (...	'DP0989'	'De Petris (...	39.2342	9.1877	'BUSTOP'	'*'	1
109	109	'Fiume (an...	'FI2020'	'Fiume (an...	39.2354	9.2020	'BUSTOP'	'*'	1
110	110	'San Giaco...	'SG1023'	'San Giaco...	39.2181	9.1191	'BUSTOP'	'*'	0
111	111	'San Giaco...	'SG1022'	'San Giaco...	39.2195	9.1194	'BUSTOP'	'*'	0
112	112	'San Giova...	'SG1020'	'San Giova...	39.2237	9.1200	'BUSTOP'	'*'	0

La **betweenness centrality** misura il *ruolo di mediatore* di un nodo nella rete e abbiamo determinato che per la rete CTM quello con misura maggiore corrisponde alla fermata Brigata Sassari (CTM Point). Dall'immagine 5.5 si nota infatti che per questa fermata passano le linee 30, 31, 1Q, 40, 41, QSA, QSB, 19. L'elevato valore di betweenness è dovuto, appunto, al fatto che attraverso questa fermata è possibile mettere in comunicazione diverse parti della rete, ovvero che equivale al nodo attraveso cui è obbligatorio passare per accedere a diverse porzioni della rete (in telecomunicazioni equivarrebbe al ruolo del *router gateway*). Infatti, se analizziamo le città attraversate da ciascuna delle linee elencate sopra, vediamo che siamo in grado di raggiungere più o meno tutto l'hinterland di Cagliari:

- 30,31: Quartu, Quartucciu, Selargius, Cagliari;
- 1Q: Quartu (fino a zona Capitana);
- 40,41: Quartu (fino a zona Flumini);
- QSA: Quartu, Quartucciu, Monserrato, Sestu;

– QSB: Quartu, Quartucciu, Monserrato, Cagliari;

– 19: Quartu, Quartucciu, Monserrato, Cagliari, Elmas, Assemini.

Figura 5.5: *La fermata Brigata Sassari è nel riquadro bianco ed equivale a quella segnata dal simbolo CTM.*

- **position**: il nodo con posizione più centrale nella rete ha indice 470, ovvero sempre *San Benedetto*, in linea con la definizione data di position centrality, ovvero di nodo che riesce a comunicare nel minor tempo possibile con tutti gli altri e questo è ovvio se è anche il nodo più vicino agli altri della rete, ovvero che ha valore di closeness centrality maggiore.

Volendo fare un'analisi complessiva, calcoliamo di queste principali fermate le restanti misure di centralità, utilizzando i comandi $degree(indice–nodo)$, $closeness(indice–nodo)$, $betweenness(indice-nodo)$ e riassumiamo tutti i dati in forma tabellare:

fermata (indice)	degree	closeness	betweenness
Abruzzi(884)	7	$5.4x10^{-5}$	$3.2x10^{4}$
San Benedetto (470)	5	$6.5x10^{-5}$	$5.5x10^{4}$
Brigata Sassari (103)	5	$5.2x10^{-5}$	$9.3x10^{4}$

possiamo quindi dedurre che, nel complesso, il nodo più importante della rete CTM (nel caso di *rete non orientata*) corrisponde alla fermata **Brigata Sassari (CTM Point)**, in quanto abbiamo un valore leggermente minore di degree rispetto a quello massimo, che viene però compensato da una buona closeness ed una ancora migliore betweenness.

Se invece consideriamo la rete reale **orientata** (quindi il comando per il grafo sarà G=*digraph*(A)), possiamo calcolare le varie misure di centralità, tenendo però in considerazione la differenza tra arco in ingresso (bus che si ferma al RITORNO della linea bus) e arco in uscita dal nodo (bus che si ferma in ANDATA della linea bus):

- **in-degree**: si calcola in MATLAB utilizzando sempre il comando *centrality* a cui si passa oltre alla rete G, la specifica *indegree.* Il nodo con numero massimo di archi in ingresso ha indice 341, che corrisponde a *Roma (Molo Sanità)*, con valore $indeg = 5$. Se andiamo a calcolare in-degree per il nodo che aveva degree massimo nel caso di rete non orientata (indice 884), troviamo che questo ha indeg=3, ovvero perde il suo primato.

  Dalla figura 5.6 si nota che il valore massimo è $indeg = 5$, infatti la fermata Roma (Molo Sanità) è connessa direttamente ai nodi:

  - 81: Roma (Molo Dogana);
  - 317: Matteotti (capolinea 30, 31);
  - 332: Matteotti (capolinea M);
  - 431: Roma (stazione FS);
  - 817: Carlo Felice (Palazzo Civico).

- **out-degree**: si calcola specificando *outdegree.* Il nodo con numero massimo di archi in uscita ha indice 88, che corrisponde a *Roma (fronte Molo Sanità)*, con valore $out - deg = 5$. Se come per in-degree calcoliamo il valore di out-degree per il nodo che nel caso di rete non orientata aveva valore massimo

mroutepoint | mbusline | fermate | route

970x9 cell

	1	2	3	4	5	6	7	8	9
332	332	'Matteotti (...	'PM0029'	'Matteotti (...	39.2150	9.1094	'BUSTOP'	'*'	1
333	333	'Vico (ang. ...	'GV0997'	'Vico (ang. ...	39.2344	9.1854	'BUSTOP'	'*'	0
334	334	'Diaz (parc...	'AD0917'	'Diaz (parc...	39.2057	9.1275	'BUSTOP'	'*'	1
335	335	'Dante (Tri...	'DA0312'	'Dante (Tri...	39.2153	9.1250	'BUSTOP'	'*'	1
336	336	'Vico (ang. ...	'GV0992'	'Vico (ang. ...	39.2344	9.1857	'BUSTOP'	'*'	0
337	337	'Trieste (chi...	'TS0041'	'Trieste (chi...	39.2174	9.1083	'BUSTOP'	'*'	0
338	338	'Is Mirrionis...	'IM0171'	'Is Mirrionis...	39.2293	9.1111	'BUSTOP'	'*'	0
339	339	'Merello (a...	'ME0164'	'Merello (a...	39.2234	9.1086	'BUSTOP'	'*'	0
340	340	'San Valeri...	'VR0656'	'San Valeri...	39.2569	9.1459	'BUSTOP'	'*'	1
341	341	'Roma (Mo...	'RM0024'	'Roma (Mo...	39.2140	9.1116	'BUSTOP'	'*'	0
342	342	'Roma (fro...	'RM0026'	'Roma (fro...	39.2130	9.1136	'BUSTOP'	'*'	1
343	343	'Merello (fr...	'ME0167'	'Merello (fr...	39.2257	9.1104	'BUSTOP'	'*'	0
344	344	'Merello (fr...	'ME0168'	'Merello (fr...	39.2277	9.1103	'BUSTOP'	'*'	0

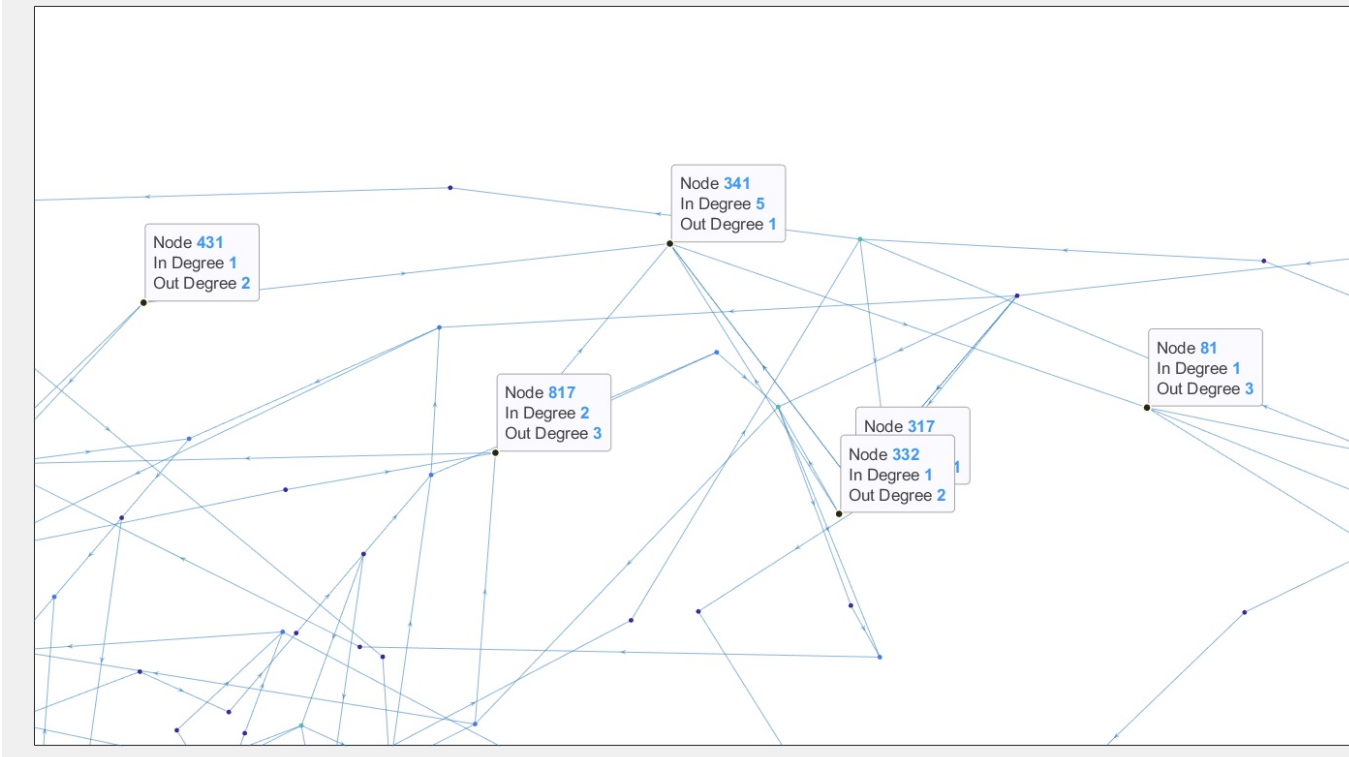


Figura 5.6: *Zoom del grafo su MATLAB che mostra i collegamenti diretti con arco orientato in ingresso verso il nodo 341.*

di degree (nodo 884), abbiamo che out-degree=4: nel complesso, quindi, in-degree+out-degree=7, che è il valore calcolato appunto nel caso ideale non orientato, mentre queste fermate messe in evidenza nel caso di rete orientata non venivano evidenziate in precedenza in quanto il loro degree totale è pari a 6, e quindi minore di quello massimo.

mroutepoint | mbusline | fermate | route

970x9 cell

	1	2	3	4	5	6	7	8	9
85	85	'Parini (ang...	'PR0978'	'Parini (ang...	39.2375	9.1796	'BUSTOP'	'*'	0
86	86	'Elmas (an...	'EL0362'	'Elmas (an...	39.2485	9.0862	'BUSTOP'	'*'	1
87	87	'Elmas (an...	'EL0363'	'Elmas (an...	39.2479	9.0865	'BUSTOP'	'*'	1
88	88	'Roma (fro...	'RM0739'	'Roma (fro...	39.2143	9.1113	'BUSTOP'	'*'	1
89	89	'Dessy (fro...	'DE0920'	'Dessy (fro...	39.2407	9.1857	'BUSTOP'	'*'	1
90	90	'San Fulgen...	'SF2010'	'San Fulgen...	39.2601	9.1332	'BUSTOP'	'*'	1
91	91	'Lungo Mar...	'LP1156'	'Lungo Mar...	39.2149	9.1765	'BUSTOP'	'*'	0
92	92	'Vesalio (sc...	'VS0684'	'Vesalio (sc...	39.2386	9.1315	'BUSTOP'	'*'	1
93	93	'Pergolesi (...	'PE0190'	'Pergolesi (...	39.2214	9.1294	'BUSTOP'	'*'	2
94	94	'Pergolesi (...	'PE0192'	'Pergolesi (...	39.2224	9.1280	'BUSTOP'	'*'	2
95	95	'Pergolesi (...	'PE0191'	'Pergolesi (...	39.2211	9.1296	'BUSTOP'	'*'	2
96	96	'Cinquini (p...	'CQ1140'	'Cinquini (p...	39.2421	9.1108	'BUSTOP'	'*'	1

Dalla figura 5.7 si nota che anche in questo caso il valore massimo è $outdeg = 5$ e la fermata Roma (fronte Molo Sanità) comunica direttamente con:

- 317: Matteotti (capolinea 30, 31);
- 332: Matteotti (capolinea M);
- 577: Matteotti (fronte ARST);
- 695: Regina Elena (fronte Bastione);
- 819: Carlo Felice (Rinascente).

- **in-closeness**: si calcola specificando *incloseness*. Il nodo con misura maggiore ha indice 342 che corrisponde alla fermata *Roma (fronte Molo Dogana)*, con valore $in-closeness = 4.5x10^{-5}$. Anche qui calcoliamo il valore di in-closeness

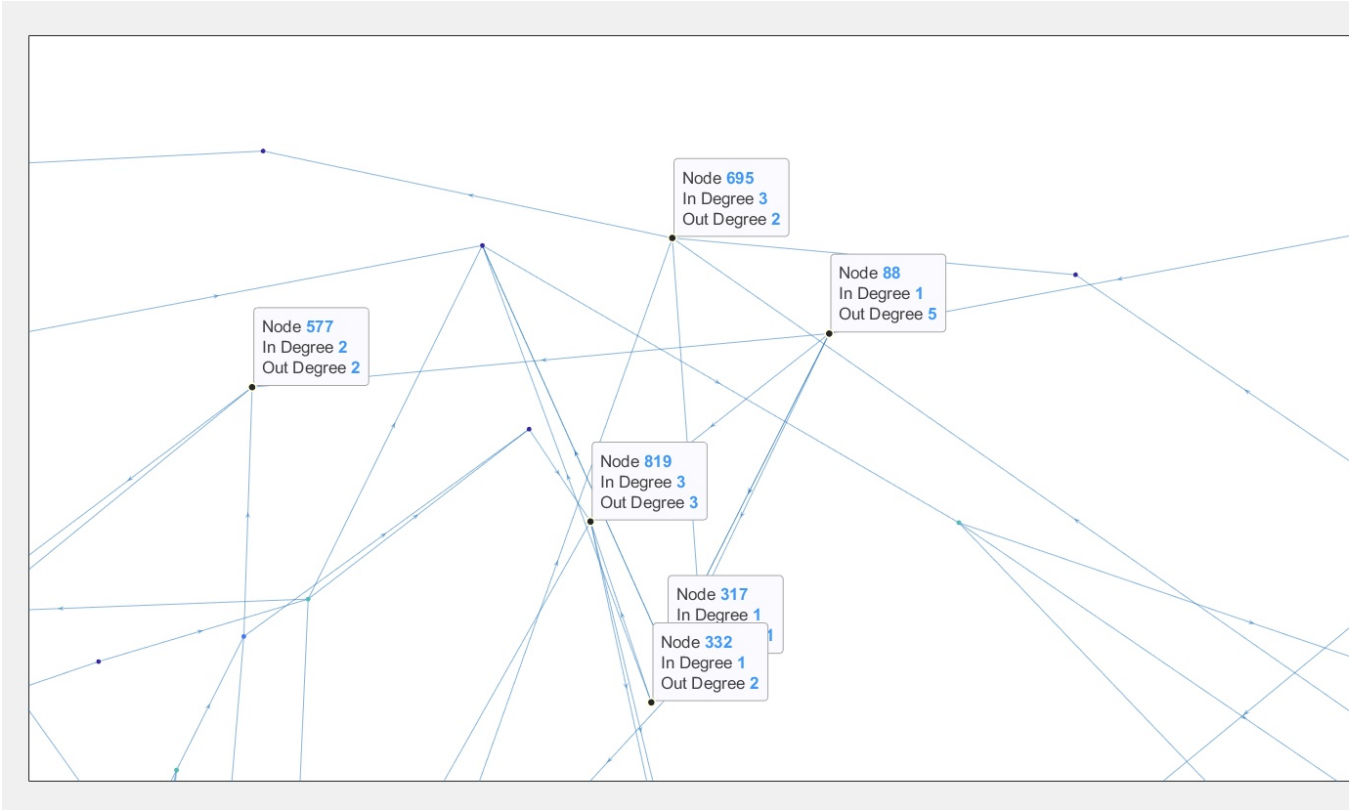


Figura 5.7: *Zoom del grafo su MATLAB che mette in evidenza i collegamenti diretti con arco orientato in uscita dal nodo 88.*

per il nodo che nel caso di rete non orientata aveva valore massimo (nodo 470) e abbiamo in-closeness=$4.35x10^{-5}$, che è appunto minore di quello massimo calcolato invece nel caso di rete orientata.

mroutepoint | mbusline | fermate | route

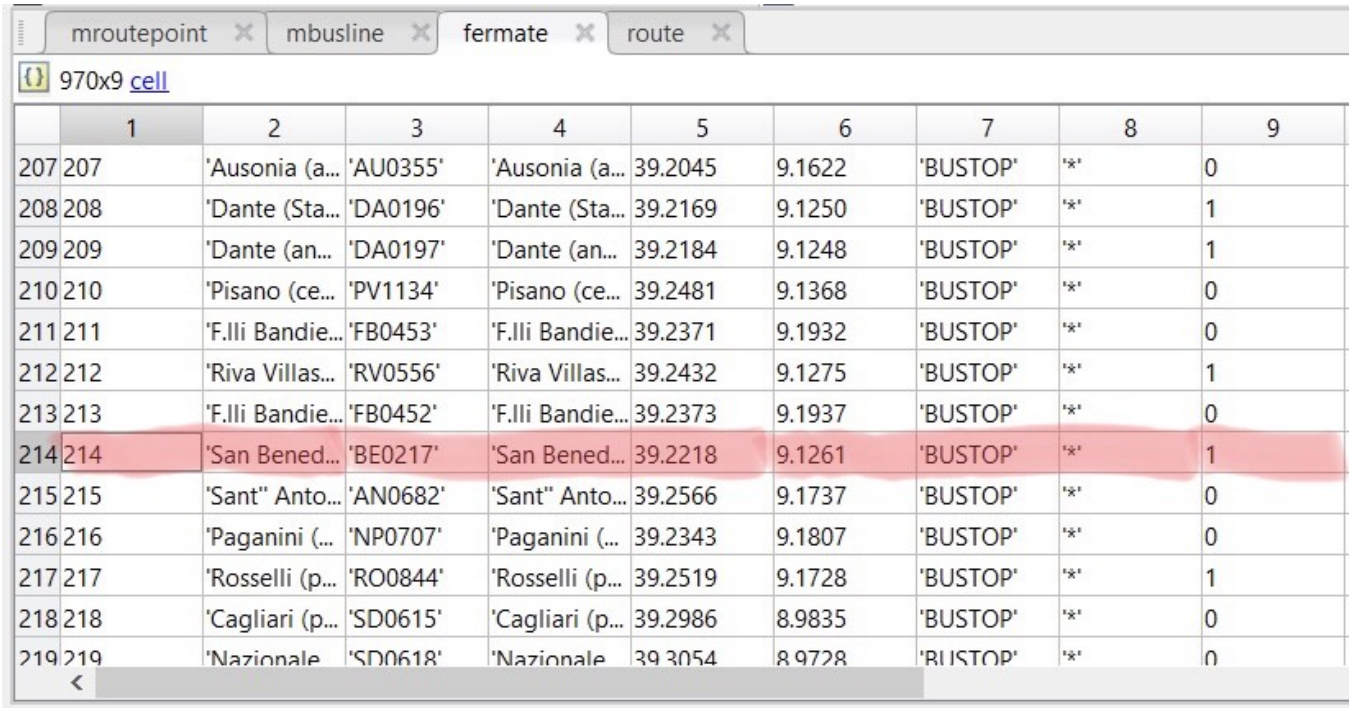
970x9 cell

	1	2	3	4	5	6	7	8	9
340	340	'San Valeri...	'VR0656'	'San Valeri...	39.2569	9.1459	'BUSTOP'	'*'	1
341	341	'Roma (Mo...	'RM0024'	'Roma (Mo...	39.2140	9.1116	'BUSTOP'	'*'	0
342	342	'Roma (fro...	'RM0026'	'Roma (fro...	39.2130	9.1136	'BUSTOP'	'*'	1
343	343	'Merello (fr...	'ME0167'	'Merello (fr...	39.2257	9.1104	'BUSTOP'	'*'	0
344	344	'Merello (fr...	'ME0168'	'Merello (fr...	39.2277	9.1103	'BUSTOP'	'*'	0
345	345	'Merello (a...	'ME0165'	'Merello (a...	39.2252	9.1103	'BUSTOP'	'*'	0
346	346	'Merello (a...	'ME0166'	'Merello (a...	39.2236	9.1092	'BUSTOP'	'*'	0
347	347	'Palau (ang...	'PA0817'	'Palau (ang...	39.2617	9.1386	'BUSTOP'	'*'	0
348	348	'Schiavazzi ...	'SV0907'	'Schiavazzi ...	39.1967	9.1387	'BUSTOP'	'*'	1
349	349	'Ospedale ...	'HB2058'	'Ospedale ...	39.2504	9.1089	'BUSTOP'	'*'	0
350	350	'Gherardo ...	'GH2016'	'Gherardo ...	39.2533	9.0935	'BUSTOP'	'*'	1
351	351	'Gherardo ...	'GH2017'	'Gherardo ...	39.2534	9.0938	'BUSTOP'	'*'	1

- **out-closeness**: si calcola specificando *outcloseness*. Il nodo con misura maggiore ha indice 214 e corrisponde alla fermata *San Benedetto (Buon Pastore)*, con valore $out - closeness = 4.6x10^{-5}$. Anche qui calcoliamo il valore per il nodo con valore di closeness maggiore nel caso di rete non orientata e abbiamo out-closeness=$4x10^{-5}$, quindi anche in questo caso la fermata San Benedetto (Buon Pastore) perde la sua importanza a fronte della closeness centrality se si analizza il caso reale di rete orientata.

- **betweenness**: non si ha un comando distinto tra reti orientate e non-orientate e il nodo con betweenness centrality maggiore corrisponde sempre a quello con indice 103, come per la rete non orientata, ma in questo caso il valore massimo è $1,49x10^5$.

Volendo fare, come per la rete non orientata, una valutazione complessiva, calcoliamo le varie misure di centralità in e out per le fermate messe in evidenza nel caso appunto di *rete orientata* e riassumiamo i valori in forma tabellare:



mroutepoint | mbusline | fermate | route

970x9 cell

	1	2	3	4	5	6	7	8	9
207	207	'Ausonia (a...	'AU0355'	'Ausonia (a...	39.2045	9.1622	'BUSTOP'	'*'	0
208	208	'Dante (Sta...	'DA0196'	'Dante (Sta...	39.2169	9.1250	'BUSTOP'	'*'	1
209	209	'Dante (an...	'DA0197'	'Dante (an...	39.2184	9.1248	'BUSTOP'	'*'	1
210	210	'Pisano (ce...	'PV1134'	'Pisano (ce...	39.2481	9.1368	'BUSTOP'	'*'	0
211	211	'F.lli Bandie...	'FB0453'	'F.lli Bandie...	39.2371	9.1932	'BUSTOP'	'*'	0
212	212	'Riva Villas...	'RV0556'	'Riva Villas...	39.2432	9.1275	'BUSTOP'	'*'	1
213	213	'F.lli Bandie...	'FB0452'	'F.lli Bandie...	39.2373	9.1937	'BUSTOP'	'*'	0
214	214	'San Bened...	'BE0217'	'San Bened...	39.2218	9.1261	'BUSTOP'	'*'	1
215	215	'Sant'' Anto...	'AN0682'	'Sant'' Anto...	39.2566	9.1737	'BUSTOP'	'*'	0
216	216	'Paganini (...	'NP0707'	'Paganini (...	39.2343	9.1807	'BUSTOP'	'*'	0
217	217	'Rosselli (p...	'RO0844'	'Rosselli (p...	39.2519	9.1728	'BUSTOP'	'*'	1
218	218	'Cagliari (p...	'SD0615'	'Cagliari (p...	39.2986	8.9835	'BUSTOP'	'*'	0
219	219	'Nazionale	'SD0618'	'Nazionale	39.3054	8.9728	'BUSTOP'	'*'	0

fermata (indice)	in-deg	out-deg	in-clos	out-clos	between
Roma (Molo S.) (341)	5	1	$4.4x10^{-5}$	$4x10^{-5}$	$1.4x10^{5}$
Roma (fronte Molo S.) (88)	1	5	$4.3x10^{-5}$	$4.3x10^{-5}$	$1.3x10^{5}$
Roma (fronte Molo D.) (342)	4	1	$4.5x10^{-5}$	$4.1x10^{-5}$	$1.3x10^{5}$
San Benedetto (214)	2	2	$3.6x10^{-5}$	$4.6x10^{-5}$	10^{5}

Dalla tabella si può quindi notare che il nodo complessivamente più importante nel caso di *rete orientata* corrisponde a **Roma (Molo Sanità)** che anche se ha un basso valore di out-degree, ovvero di linee che passano durante l'andata, compensa con un alto valore di in-degree, ovvero di linee che passano per il ritorno. Inoltre è caratterizzato da valore di in-closeness e betweenness comunque vicini a quello massimo. Altro nodo molto importante, solo un gradino più in basso è Roma (fronte Molo Sanità) che ha valori di in-degree e out-degree opposti alla fermata analizzata sopra, ma dei valori di in-closeness e out-closeness più equilibrati.

Bibliografia

[1] Directed and unidirected graphs. https://it.mathworks.com/help/matlab/math/directed-and-undirected-graphs.html, 2021.

[2] Graph plotting and customization. https://it.mathworks.com/help/matlab/math/graph-plotting-and-customization.html, 2021.

[3] Mathworks centrality. https://it.mathworks.com/help/matlab/ref/graph.centrality.html, 2021.

[4] Sparse matrices. https://it.mathworks.com/help/matlab/sparse-matrices.html, 2021.

[5] Wikipedia centrality. https://en.wikipedia.org/wiki/Centrality, 2021.

[6] A. Concas, , L. Reichel, G. Rodriguez, and Y. Zhang. Chained graphs and some applications. Submitted, 2020.

[7] A. Concas, C. Fenu, and G. Rodriguez. PQser: a Matlab package for spectral seriation. *Numer. Algorithms*, 80(3):879–902, 2019. DOI: 10.1007/s11075-018-0510-6.

[8] E. Estrada and F. Knight. *A First Course in Network Theory.* Oxford University Press, Oxford, 2015.

[9] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez. Block Gauss and anti-Gauss quadrature with application to networks. *SIAM J. Matrix Anal. Appl.*, 34(4):1655–1684, 2013. DOI: 10.1137/120886261.

[10] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez. Network analysis via partial spectral factorization and Gauss quadrature. *SIAM J. Sci. Comput.*, 35(4):A2046–A2068, 2013. DOI: 10.1137/130911123.

[11] G. Rodriguez and S. Seatzu. *Introduzione alla Matematica Applicata e Computazionale.* Pitagora Editrice, Bologna, 2010.