POLITECNICO DI TORINO

Homework #1

Student name: *Filippo Scaramozzino 312856, Maria Antonietta Longo 308756*

Course: *Dinamiche su Network*

1. Primo esercizio

1. Consider the network in Figura 1 with link capacities

$$c_2 = c_4 = c_6 = 1, c_1 = c_3 = c_5 = 2$$

(*a*) What is the minimum aggregate capacity that needs to be removed for no feasible flow from *o* to *d* to exist?

(*b*) What is the maximum aggregate capacity that can be removed from the links without affecting the maximum throughput from *o* to *d*?

(*c*) You are given $x > 0$ extra units of capacity. How should you distribute them in order to maximize the throughput that can be sent from o to d? Plot the maximum throughput from *o* to *d* as a function of $x \geq 0$.

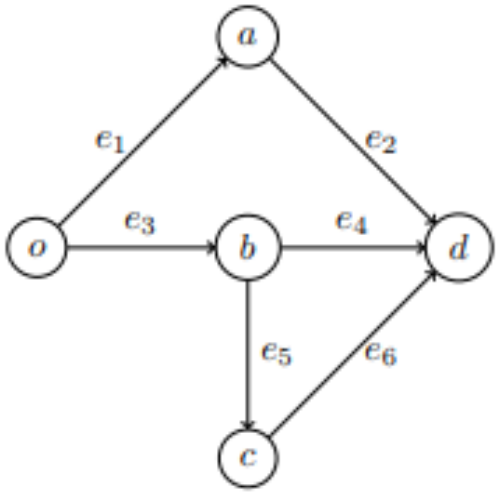


Figura 1

Soluzione.

(*a*) Per capire quale sia la quantità minima di unità di capacità da rimuovere dal grafo affinchè non ci sia *feasible flow* dal nodo *o* al nodo *d* abbiamo bisogno di vedere quale sia la capacità del minimo taglio. Infatti, il teorema secondo cui la capacità del minimo taglio coincide con la capacità di flusso massimo su di un Grafo $G = (V, E, C)$,

ci garantisce che se rimuoviamo la quantità di flusso corrispondente alla capacità del minimo taglio, allora il massimo flusso sarà uguale a zero e dunque non ci sarà *feasible flow* da o a d. Dunque, visto che il grafo della Figura 1 è semplice e con un numero ridotto di nodi posso definire tutte le partizioni possibili dell'insieme dei nodi tali che $o \subseteq U$ e $d \subseteq U^c$, con $o \neq d$ e vedere quale delle 8 partizioni abbia capacità più bassa. Un'altra alternativa è quella di applicare l'algoritmo di Ford e Fulkerson. Visto che partizionare e calcolare le capacità ci sarà utile nel punto (b), percorriamo questa strada e usiamo Ford e Fulkerson come verifica. **Attraverso questo procedimento si ricava che il minimo della capacità aggregata da rimuovere affinchè non ci sia *feasible flow* da o a d è 3**.

$$U_1 = \{o\}, U_1^c = \{a, b, c, d\}, C_{U_1} = 4$$
$$U_2 = \{o, a\}, U_2^c = \{b, c, d\}, C_{U_2} = 3$$
$$U_3 = \{o, b\}, U_3^c = \{c, d\}, C_{U_3} = 5$$
$$U_4 = \{o, c\}, U_4^c = \{a, b, d\}, C_{U_4} = 5$$
$$U_5 = \{o, a, b\}, U_5^c = \{c, d\}, C_{U_5} = 4$$
$$U_6 = \{o, a, c\}, U_6^c = \{b, d\}, C_{U_6} = 4$$
$$U_7 = \{o, b, c\}, U_7^c = \{a, d\}, C_{U_7} = 4$$
$$U_8 = \{o, a, b, c\}, U_8^c = \{d\}, C_{U_8} = 3$$

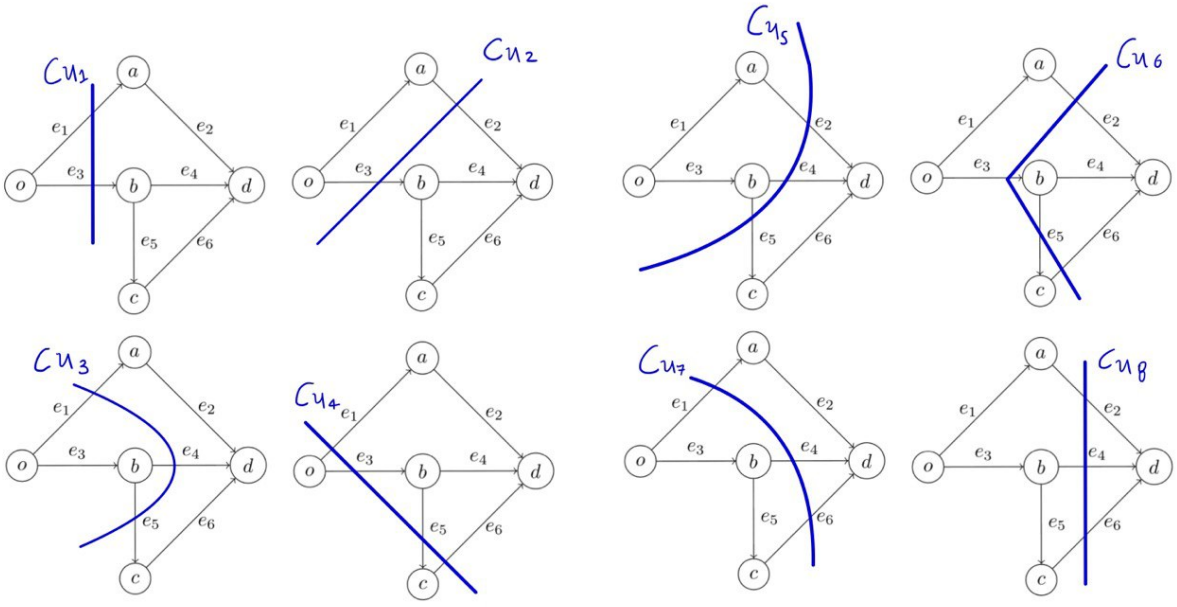
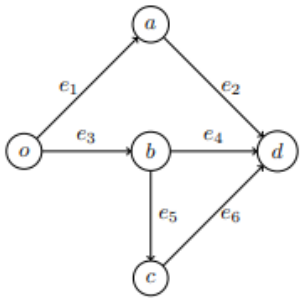
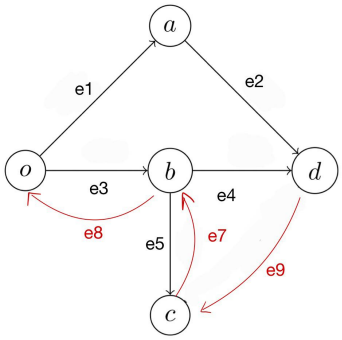


Figura 2

Troviamo conferma di quanto detto applicando l'algoritmo di Ford e Fulkerson.



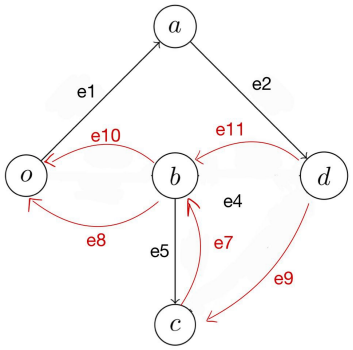
Cominciamo con $f^{(0)} = (0,0,0,0,0,0), E_0 = (e_1, e_2, e_3, e_4, e_5, e_6), c_0 = (2,1,2,1,2,1)$



scelgo $\gamma^{(0)} = (o, b, c, d)$, a questo punto

$$\epsilon = min(2,2,1) = 1$$

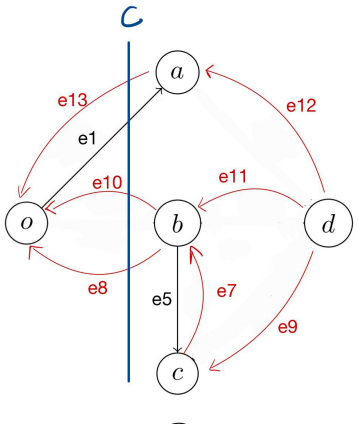
e allora $f^{(1)} = (0,0,1,0,1,1), E = (e_1, e_2, e_3, e_4, e_5, e_7, e_8, e_9), c_1 = (2,1,1,1,1,0)$



scelgo $\gamma^{(1)} = (o, b, d)$, a questo punto

$$\epsilon = min(1,1) = 1$$

e allora $f^{(2)} = (0,0,2,1,1,1), E = (e_1, e_5, e_7, e_8 e_9, e_{10}, e_{11}), c_2 = (2,1,0,0,1,0)$



scelgo $\gamma^{(2)} = (o, a, d)$, a questo punto

$$\epsilon = min(2, 1) = 1$$

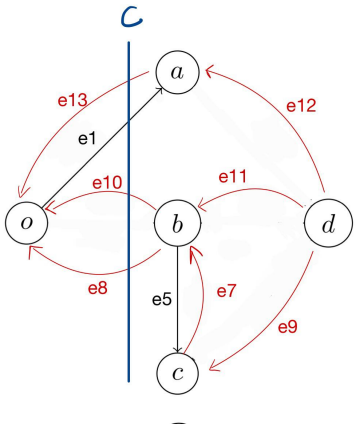
e allora $f^{(3)} = (1, 1, 2, 1, 1, 1), E = (e_1, e_5, e_7, e_8 e_9, e_{10}, e_{11}, e_{12}, e_{13}), c_3 = (1, 0, 0, 0, 1, 0)$.

Nel grafo di scarto non ci sono più cammini possibili da o a $d \rightarrow$ **STOP**.

Dal grafo di scarto, poichè $c_{e_{18}} = c_{e_{10}} = c_{e_{13}} = 1$ si evince che la capacità del minimo taglio è 3 che è proprio il risultato che ci aspettavamo.

(*b*) Per capire quante unità di capacità rimuovere dagli archi senza influenzare il max τ, possiamo decrementare di un'unità alla volta la capacità di ogni cut facendo sì che la capacità del min cut continui ad essere la minore di tutte.

Prima di procedere in questo senso, possiamo riprendere il grafo di scarto finale del punto (a):



e notare che negli archi e_5 ed e_1 resta ancora una unità di capacità ciascuno; pertanto, analizzando ogni taglio nella maniera descritta sopra, ci aspettiamo che il massimo numero di capacità che possiamo rimuovere senza influenzare τ è $c_{max} = 2$. In particolare dovremmo poter rimuovere proprio una unità dall'arco e_1 e una dall'arco e_5.

Modifiche possibili:

- Non posso rimuovere 2 unità da (o,a) altrimenti avrei $C_{U_1} = 2 < 3$
- Non posso rimuovere 2 unità da (o,b) altrimenti avrei $C_{U_2} = 1 < 3$
- Non posso rimuovere 1 unità da (o,b) altrimenti avrei $C_{U_2} = 2 < 3$
- Non posso nemmeno rimuovere 1 unità da qualcuno degli archi (a,d), (b,d) e (c,d) perchè altrimenti avrei che $C_{U_7} \leq 2 < 3$
- Rimuovo 1 unità da (b,c) e ottengo che, per gli archi coinvolti:
  - $C_{U_3} = 4$
  - $C_{U_5} = 3$

  mentre le altre capacità restano inalterate → **Posso rimuovere una unità da (b,c)**

- Rimuovo 1 unità da (o,a) e ottengo che, per gli archi coinvolti:
  - $C_{U_1} = 3$
  - $C_{U_3} = 3$
  - $C_{U_4} = 4$
  - $C_{U_7} = 3$

  mentre le altre capacità restano inalterate → **Posso rimuovere una unità da (o,a)**

Dunque, **è possibile eliminare 2 unità di capacità e il flusso massimo resta inalterato**.

(*c*) **In collaborazione con gli studenti Moubane 305716, Racca 315163, Vay 316737**

La base, su cui si fonda il nostro ragionamento, è quella di aumentare le capacità dei mincuts in modo da fargli raggiungere le capacità dei che tagli di capacità poco sopra delle capacità dei mincuts. Indicheremo da ora in poi il valore di questa capacità con C_+ e il valore della capacità dei mincuts con C_{mincut} si noti che durante il ragionamento questi valori verranno aggiornati a ogni step. Un'osservazione importante che teniamo a fare è che l'aumento di capacità di un arco e porta al medesimo aumento delle capacità dei tagli attraversati in modo uscente da e. Il ragionamento effettuato di seguito va a step, dove per ogni step avremmo un insieme di tagli che sono tutti mincut che indicheremo con C_m , l'obbiettivo è scrivere questo insieme come unione di minor numero n di sottoinsiemi di C_m tale che siano formati da mincuts attraversati da uno stesso arco. A parità di sottoinsiemi scelgo gli archi che infuenzano il maggior numero di tagli contemporaneamete.

Arco e	Capacità dei tagli dipendenti dall'arco e
e_1	C_1, C_3, C_4, C_5
e_2	C_2, C_5, C_6, C_8
e_3	C_1, C_2, C_4, C_6
e_4	C_3, C_5, C_7, C_8
e_5	C_3, C_5
e_6	C_4, C_6, C_7, C_8

Iterazioni del ragionamento.

1. $\mathcal{C}_m = \{C_2, C_8\} \rightarrow e_2$

Arco e	Capacità dei mincuts dipendenti dall'arco e
e_1	$\emptyset$
e_2	C_2, C_8
e_3	C_2
e_4	C_8
e_5	$\emptyset$
e_6	C_8

$n = 1$ quindi se $x \leq n(C_+ - C_{mincut})$ aumento di $\frac{x}{n} = x$ la capacità di e_2 che aumenta contemporaneamente i mincuts C_2 e C_8 e si raggiunge la configurazione finale di massimizzazione dei mincuts.

Invece se $x > n(C_+ - C_{mincut})$ aumentiamo la capacità dell'arco e_2 di $\frac{n(C_+ - C_{mincut})}{n} = 1$,la capacità residua $x_1 = x - n(C_+ - C_{mincut})$ verrà usata nel prossimo step con ragionamento analogo. Nel nostro caso ipotizziamo che $x \in [0, \infty)$,i nuovi valori delle capacità dei tagli in questo primo step sono i seguenti:

$$C_1 = 4, C_2 = 4, C_3 = 5, C_4 = 5, C_5 = 5, C_6 = 5, C_7 = 4, C_8 = 4$$

2. $\mathcal{C}_m = \{C_1, C_2, C_7, C_8\} = \{C_1, C_2\} \cup \{C_7, C_8\} \rightarrow e_3$ e e_4

Arco e	Capacità dei mincuts dipendenti dall'arco e
e_1	C_1
e_2	C_2, C_8
e_3	C_1, C_2
e_4	C_7, C_8
e_5	$\emptyset$
e_6	C_7, C_8

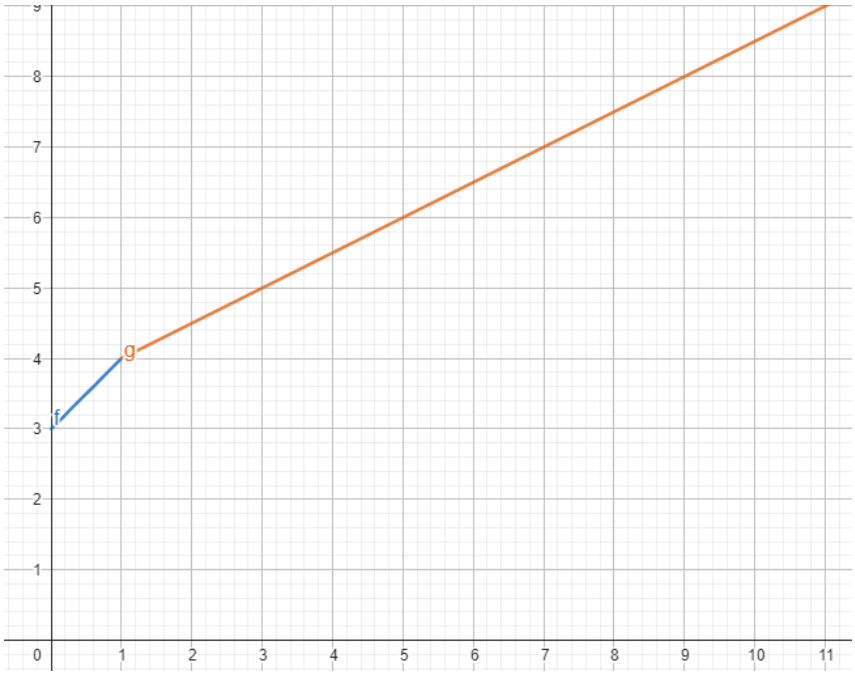
$n = 2$ se fosse $x_1 \leq n(C_+ - C_{mincut}) = 2$ aumenteremo di $\frac{x_1}{2}$ la capacità di e_3 e e_4 i quali aumenterebbero contemporaneamente i mincuts C_1,C_2,C_7 e C_8 e si raggiungerebbe la configurazione finale di massimizzazione dei mincuts. Visto che stiamo considerando

$x \in [0, infty)$ siamo nel $x_1 > 2$ quindi dobbiamo aumentare la capacità degli archi e_3 e e_4 di $(C_+ - C_{mincut}) = 1$, la capacità residua $x_2 = x_1 - n(C_+ - C_{mincut})$ verrà usata nel successivo step seguendo gli stessi criteri dei precedenti step.Ora i valori delle capacità dei tagli sono i seguenti.

$$C_1 = 5, C_2 = 5, C_3 = 6, C_4 = 6, C_5 = 6, C_6 = 6, C_7 = 5, C_8 = 5$$

3. $C_m = \{C_1, C_2, C_7, C_8\} = \{C_1, C_2\} \cup \{C_7, C_8\} \rightarrow e_3$ e e_4

Si può osservare che i mincuts sono gli stessi del punto precedente, quindi la scelta degli archi sarà la medesima e anche la capaità dei vari tagli sarà la stessa. Questo ragionamento si può iterare per un qualunque successivo step e si può dedurre che l'incremento sarà sempre $\frac{x_i}{2}$ all'$(i+1)$-esimo step. In conclusione il grafico di $\tau(x)$ è il seguente:



2. Secondo esercizio

2. Consider o-d network flows on the graph in Figura 2. The links are endowed with delay functions

$\tau_1(x) = \tau_6(x) = 3x, \tau_2(x) = \tau_3(x) = x + 1, \tau_4(x) = 2x, \tau_5(x) = 2,$
and the throughput is 2.

(*a*) Compute the social optimum flow vector, i.e., the flow vector that minimizes the average delay from o to d.

(*b*) Compute the user optimum flow vector, i.e., the Wardrop equilibrium, and the price of anarchy.

(*c*) Consider a new link e_7 with delay function $\tau_7(x) = x$. Find a head and a tail of the link e_7 such that Braess' paradox arises, and compute the price of anarchy on the new graph.

(*d*) Compute an optimal toll vector ω on the new graph, i.e., a non-negative toll vector that reduces the price of anarchy to 1. If possible, compute a full-support optimal toll vector, i.e., such that $\omega_e > 0$ for every link e. Construct an optimal toll vector with the smallest possible support.

(*e*) Consider the original graph in Figura 2, and add an additional link $e_7 = (n_1, n_4)$ with delay function

$$\tau_7(x) = \alpha x + 2, \alpha \geq 0.$$

Consider o-d network flows on the new graph with throughput $X > 0$. Find an optimal toll vector independent of X and α. Hint: focus on the optimization problem related to social optimum flows and Wardrop equilibria flows

.

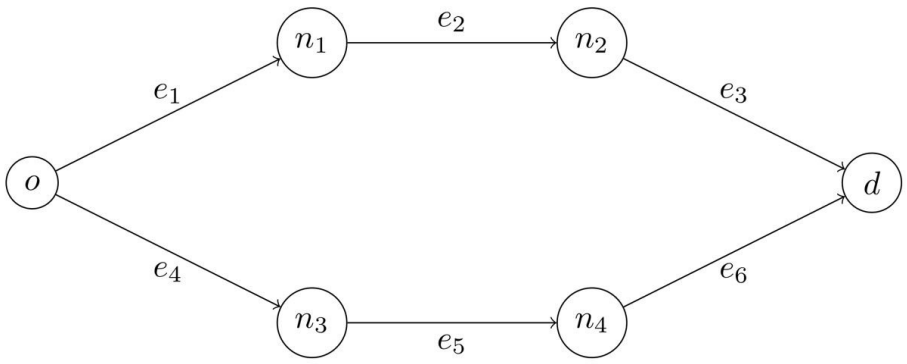


Figura 2

Soluzione. Per questo esercizio ci riferiamo al grafo della Figura 2 con la notazione riportata nella Figura 2.1:

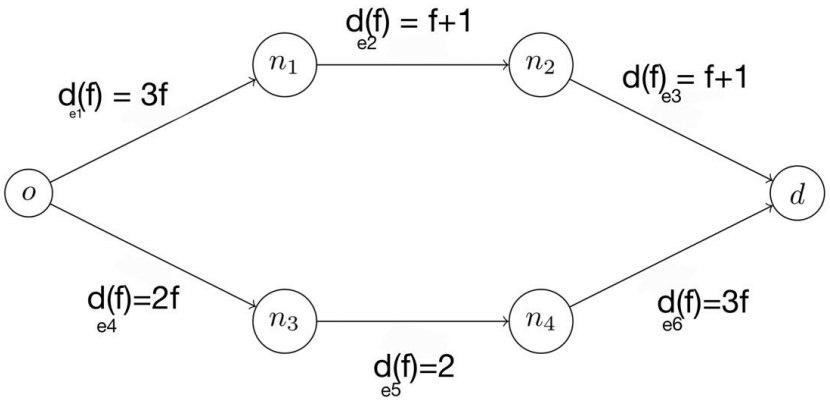


Figura 2.1

In questo grafo con $\tau = 2$ individuiamo due *o-d path*.
$\mathrm{p}^{(1)} = o \to n_1 \to n_2 \to d$
$\mathrm{p}^{(2)} = o \to n_3 \to n_4 \to d$
$z = (z_1, z_2)$

(*a*) Per avere l'ottimo sociale devo risolvere

$$\min z_1(3z_1 + z_1 + 1 + z_1 + 1) + z_2(2z_2 + 2 + 3z_2)$$
$$s.t. z_1 + z_2 = 2$$

ovvero

$$\min z_1(5z_1 + 2) + z_2(5z_2 + 2)$$
$$s.t. z_1 + z_2 = 2$$

Poichè $z_1 = 2 - z_2$, posso scrivere tutto in funzione di z_2 e ottengo che la funzione da minimizzare è:

$$(2 - z_2)(10 - 5z_2 + 2) + z_2(5z_2 + 2) =$$
$$24 - 10z_2 - 12z_2 + 5z_2^2 + 5z_2^2 + 2z_2 =$$
$$10z_2^2 - 20z_2 + 24$$

Dunque considero la funzione obiettivo $f(z_2) = 5z_2^2 - 10z_2 + 12$ e la minimizzo: verifico che è convessa: $f'(z_2) = 10z_2 - 10$ ed $f''(z_2) = 10 > 0 \Rightarrow$ convessa. Trovo un minimo quando $f'(z_2) = 0 \Leftrightarrow z_2 - 1 = 0 \Leftrightarrow z_2 = 1$

Da cui si ottiene:

$$z_1 = z_2 = 1 \to z = (z_1, z_2)^T = (1,1)^T \Rightarrow f^* = (f_1, \ldots, f_6)^T = (1,1,1,1,1,1)^T$$

Calcoliamo anche il costo totale corrispodente:

$$\sum_{e \in E} f_e^* d_e\,(f_e^*) = 14$$

(*b*) Cominciamo calcolando l'equilibio di Wardrop ovvero l'ottimo per gli utenti. Una configurazione di flussi è equilibrio di Wardrop se $\forall z_i$, $i = 1, 2$ associato ad un o-d path vale che $z_i > 0$ solo se $\Delta_i \leq \Delta_j$ per ogni altro o-d path $p^{(j)}$ con Δ_i funzione di ritardo associata a $p^{(i)}$. In questo caso:

$$\Delta_1 = 3z_1 + z_1 + 1 + z_1 + 1 = 5z_1 + 2$$

$$\Delta_2 = 2z_2 + 2 + 3z_2 = 5z_2 + 2$$

- $$z_1 < 0 \qquad \rightarrow \qquad \Delta_1 \leq \Delta_2 \qquad \Leftrightarrow \qquad 5z_1 + 2 \leq 5z_2 + 2$$

  Usando che $z_2 = 2 - z_1$ e svolgendo i calcoli otteniamo:

  $$z_1 \leq 1$$

- $$z_2 < 0 \qquad \rightarrow \qquad \Delta_2 \leq \Delta_1 \qquad \Leftrightarrow \qquad 5z_2 + 2 \leq 5z_1 + 2$$

  Usando che $z_2 = 2 - z_1$ e svolgendo i calcoli otteniamo:

  $$z_1 \geq 1$$

  Da cui si ricava che $z_1 = 1$ e $z_2 = 2 - z_1 = 1$

Per quanto detto si ottiene:

$$z = (z_1, z_2)^T = (1, 1)^T$$

$$\Rightarrow f^{(0)} = (f_1, \ldots, f_6)^T = (1, 1, 1, 1, 1, 1)^T$$

Detto cio, possiamo dobbiamo usare questa distribuzione di flussi per calcolare il Price of Anarchy:

$$PoA = \frac{\sum_{e \in E} f_e^{(0)} d_e \left(f_e^{(0)} \right)}{\sum_{e \in E} f_e^* d_e \left(f_e^* \right)} = \frac{14}{14} = 1$$

Pertanto si ottiene che ottimo degli utenti (eq. Wardrop) e ottimo sociale o di sistema coincidono.

(*c*) Posizioniamo l'arco e_7 con coda sul nodo n_1 e testa sul nodo n_4 come in Figura 2.2 in modo da influire sui due o-d paths descritti nel punto a):

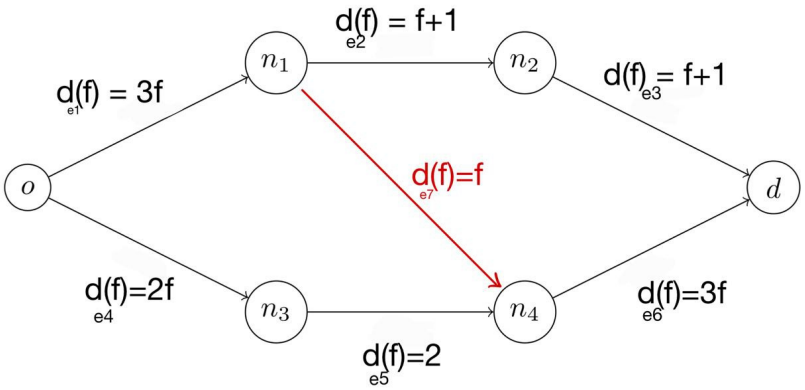


Figura 2.2

A questo punto mostriamo che si verifica il paradosso di Braess, ovvero ricalcoliamo ottimo sociale ed equilibrio di Wardrop sul nuovo grafo e verifichiamo che, con questa scelta, il Price Of Anarchy finale risulta maggiore di 1.
Dopo aver notato che c'è un nuovo o-d path $p^{(1)} = o \to n_1 \to n_2 \to d$, impostiamo il seguente sistema per calcolare l'ottimo sociale:

$$\begin{cases} z_1 + z_2 + z_3 = 2 \\ 3\left(z_1 + z_3\right)^2 + 2\left(z_1 + 1\right) z_1 + 2z_2^2 + 2z_2 + 3\left(z_2 + z_3\right)^2 = g(z_1, z_2, z_3) \end{cases}$$

$$\Rightarrow g\left(z_1,z_2\right) = 6z_1^2 + 6z_2 + 2z_1z_2 - 14z_1 - 14z_2 + 28$$

$$\begin{cases} \dfrac{\partial g}{\partial z_1} = 12z_1 + 2z_2 - 14 = 0 \\ \dfrac{\partial g}{\partial z_2} = 12z_2 + 2z_1 - 14 = 0 \end{cases} \Rightarrow z_1 = z_2$$

$$\Rightarrow 12z_2 + 2z_1 - 14 = 0 \Rightarrow z_1 = 1 \Rightarrow z_2 = 1 \wedge z_3 = 0$$

Da cui otteniamo i seguenti vettori di flusso ottimale su archi e su paths.

$$\Rightarrow z^* = \left(z_1^*, z_2^*, z_3^*\right)^T = (1,1,0)^T$$

$$\Rightarrow f^* = \left(f_1^*, \ldots, f_7^*\right)^T = (1,1,1,1,1,1,0)^T$$

Per quanto riguarda l'equilibrio di Wardrop:

$$\begin{aligned} \Delta_1 &= 5z_1 + 3z_3 + 2 \\ \Delta_2 &= 5z_1 + 3z_3 + 2 \\ \Delta_3 &= 3z_1 + 3z_2 + 7z_3 \end{aligned}$$

$$z_1 > 0 \Rightarrow \begin{cases} \Delta_1 \leq \Delta_2 \Rightarrow z_1 \leq z_2 \\ \Delta_1 \leq \Delta_3 \Rightarrow 2z_1 - 3z_2 - 4z_3 + 2 \leq 0 \end{cases}$$

$$z_2 > 0 \Rightarrow \begin{cases} \Delta_2 \leq \Delta_1 \Rightarrow z_2 \leq z_1 \\ \Delta_2 \leq \Delta_3 \Rightarrow -3z_1 + 2z_2 - 4z_3 + 2 \leq 0 \end{cases}$$

$$z_3 > 0 \Rightarrow \begin{cases} \Delta_3 \leq \Delta_1 \Rightarrow -2z_2 + 3z_2 + 4z_3 - 2 \leq 0 \\ \Delta_3 \leq \Delta_2 \Rightarrow 3z_1 - 2z_2 + 4z_3 - 2 \leq 0 \end{cases}$$

$$\begin{cases} z_1 > 0 \\ z_2 > 0 \\ z_3 > 0 \\ z_1 + z_2 + z_3 = 2 \end{cases} \Rightarrow \begin{cases} z_1 = z_2 \\ 3z_1 - 2z_2 + 4z_3 - 2 = 0 \\ 2z_1 - 3z_2 - 4z_3 + 2 = 0 \\ z_1 + z_2 + z_3 = 2 \end{cases}$$

Da cui ricaviamo i seguenti vettori dei flussi:

$$\Rightarrow z^{(0)} = (z_1, z_2, z_3)^T = \left(\frac{6}{7}, \frac{6}{7}, \frac{2}{7}\right)^T$$

$$\Rightarrow f^{(0)} = (f_1, \ldots, f_7)^T = \left(\frac{8}{7}, \frac{6}{7}, \frac{6}{7}, \frac{6}{7}, \frac{6}{7}, \frac{8}{7}, \frac{2}{7}\right)^T$$

Arrivati a questo punto abbiamo tutto ciò che ci serve per il calcolo del Price Of Anarchy:

$$PoA = \frac{14.29}{14} > 1$$

Poichè $PoA > 1$ deduciamo che si è verificato il paradosso di Braess: il tempo di permanenza medio dell'utente non migliora (nel senso che non diminuisce) se viene introdotto un nuovo arco nella rete.

(*d*) **In collaborazione con gli studenti Moubane 305716, Racca 315163, Vay 316737**
Conoscendo il vettore di flussi ottimale sugli archi e il vettore delle derivate delle funzioni di costo, posso calcolare tutte le componenti del vettore ottimale ω_e^* dei tolls applicando la seguente formula:

$$\omega_e^* = f_e^* \cdot d_e'(f_e^*)$$

Il vettore delle funzioni di costo e il vettore ottimale dei flussi sono rispettivamente:

$$d(f) = (3f, f+1, f+2, 2f, 2, 3f, f)^T$$

e

$$f^* = (f_1^*, \ldots, f_7^*)^T = (1, 1, 1, 1, 1, 1, 0)^T$$

Applicando la formula di sopra otteniamo il vettore cercato:

$$\Rightarrow \omega^* = (3, 1, 1, 2, 0, 3, 0)^T$$

Ora che abbiamo trovato ω_e^*, sommiamo ad ogni funzione di costo la medesima quantità costante $c > 0$ possiamo verificare che la scelta di un path a discapito dell'altro, per chi si immette nella rete, resta invariata .

Calcoliamo l'equilibrio di Wardop con la nuova configurazione di tolls e vediamo che arriviamo ad ottenere il vettore di flussi calcolato tramite lo studio dell'ottimo sociale.

$$\omega^+ = \omega^* + 1^T * c, c > 0$$

Aggiungiamo $c > 0$ e troviamo i seguenti tempi di percorrenza su paths:

$$\begin{aligned}\Delta_1 &= 5z_1 + 3z_3 + 2 + (5 + 3c)\\ \Delta_2 &= 5z_1 + 3z_3 + 2 + (5 + 3c)\\ \Delta_3 &= 3z_1 + 3z_2 + 7z_3 + (6 + 3c)\end{aligned}$$

Ricaviamo:

$$\Rightarrow z^{(0)} = (z_1, z_2, z_3)^T = (1, 1, 0)^T$$

$$\Rightarrow f^{(0)} = (f_1, \ldots, f_7)^T = (1, 1, 1, 1, 1, 1, 0)^T$$

Per la scelta del toll ottimale nell'arco e_7 supponiamo che sia sufficiente imporre un toll grande abbastanza da fare in modo che gli utenti immessi nella rete siano restii a scegliere l'arco e_7. Abbiamo quindi imposto il seguente sistema:

$$\begin{aligned}\Delta_1 &= 5z_1 + 3z_3 + 2\\ \Delta_2 &= 5z_1 + 3z_3 + 2\\ \Delta_3 &= 3(z_1 + z_3) + (z_3 + c) + 3(z_2 + z_3)\end{aligned}$$

$$\begin{cases}\Delta_1 \leq \Delta_2\\ \Delta_2 \leq \Delta_1\end{cases} \Rightarrow z_1 = z_2$$

Allora è sufficiente inserire un toll del valore di:

$$\begin{cases}\Delta_1 \leq \Delta_3 \Rightarrow z_1 \leq \dfrac{z_3 + 4 + K}{5}\\ z_1 = z_2\\ z_1 + z_2 + z_3 = 2\\ z_1 = 1\end{cases} \Rightarrow K \geq 1$$

(*e*) **In collaborazione con gli studenti Moubane 305716, Racca 315163, Vay 316737**
Calcoliamo il nuovo ottimo sociale del grafo imponendo alla funzione di costo anche il nuovo vincolo del traffico totale:

$$\begin{cases} z_1 + z_2 + z_3 = \chi \\ 3(z_1 + z_3)^2 + 2(z_1 + 1)z_1 + 2z_2^2 + 2z_2 + 3(z_2 + z_3)^2 + (\alpha z_3 + 2)z_3 = h(z_1, z_2, z_3) \end{cases}$$

$$h(z_1, z_2) = (5 + \alpha)\left(z_1^2 + z_2^2\right) - (2\alpha + 6)(z_1\chi + z_2\chi) + 2\alpha z_1 z_2 + 2\chi + (6 + \alpha)\chi^2$$

$$\begin{cases} \frac{\partial h}{\partial z_1} = 2(5 + \alpha)z_1 - (2\alpha + 6)\chi + 2\alpha z_2 = 0 \\ \frac{\partial h}{\partial z_2} = 2(5 + \alpha)z_2 - (2\alpha + 6)\chi + 2\alpha z_1 = 0 \end{cases}$$

$$\Rightarrow z_1 = z_2 = \frac{6 + 2\alpha}{10 + 4\alpha}x \Rightarrow z_3 < 0$$

$$z_1 = \chi - z_2$$

$$\Rightarrow \begin{cases} z_1 = \chi - z_2 \\ h(z_1, z_2) \end{cases}$$

$$\Rightarrow h(z_2) = 10z_2^2 - 10\chi z_2 + 2\chi$$

$$h'(z_2) = 20z_2 - 10\chi = 0 \Rightarrow z_2 = \frac{\chi}{2} \Rightarrow z_1 = \frac{\chi}{2}$$

$$\Rightarrow z^{(0)} = [z_1, z_2, z_3]^T = \left[\frac{\chi}{2}, \frac{\chi}{2}, 0\right]^T$$

$$\Rightarrow f^{(0)} = [f_1, \ldots, f_7]^T = \left[\frac{\chi}{2}, \frac{\chi}{2}, \frac{\chi}{2}, \frac{\chi}{2}, \frac{\chi}{2}, \frac{\chi}{2}, 0\right]^T$$

Non è necessario calcolare il flusso al Wardrop infatti, basta analizzare il grafo con i flussi distribuiti al Social Optimum e verificare che ad ogni nodo la scelta di tutti gli utenti sia il path più veloce. Rapidamente si può notare come al nodo n_1 gli utenti abbiano la scelta di spostarsi sul path z_3 o proseguire, come prima, sul path z_1: il tempo di percorrenza da n_1 alla destinazione al momento del Social Optimum è pari a $+2$ lungo il path z_1 mentre se scegliessero il path z_3 allora impiegherebbero $3\frac{\chi}{2} + 2$ che è maggiore del precedente. Questo implica che nessun utente sceglierà mai un percorso diverso da quello del social optimum e quindi l'equilibrio di Wardrop è uguale al Social Optimum. Per questo il PoA è pari a 1 per qualsiasi valore di α e quindi il vettore di toll ottimali è banalmente il vettore nullo.

3. Terzo esercizio

3. Consider the simple graph $G = (V, E)$ in Figura 3.

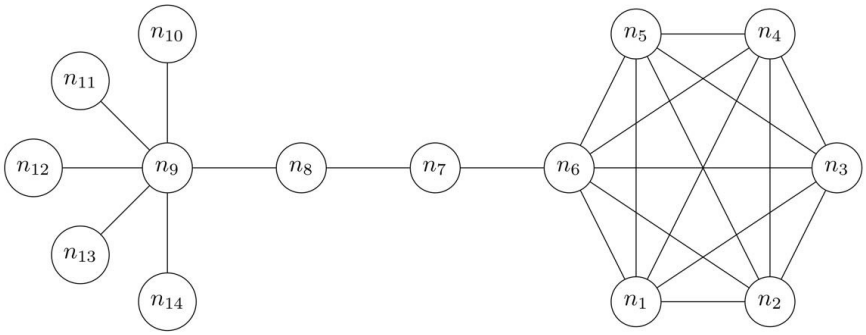


Figura 3

(*a*) Compute the degree centrality, the eigenvector centrality, the invariant distribution centrality, and comment the results. You can implement the computation in Matlab or Python.

(*b*) Write an iterative algorithm in Matlab or Python for the computation of Katz centrality, with $\beta = 0.15$ = and uniform intrinsic centrality μ.

(*c*) Write a distributed algorithm in Matlab or Python for the computation of Pagerank centrality, with $\beta = 0.15$ and uniform intrinsic centrality μ.

(*d*) Analyse the results of points (b) and (c), focusing in particular on the centrality of nodes n_6 and n_9.

Soluzione. Poichè ci servirà per tutti i punti, scriviamo prima di tutto la matrice di adiacenza W per il grafo della Figura 3:

$$W = \begin{bmatrix}
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

(*a*) Per la *degree centrality, eigenvector centrality e invariant distribution centrality* ragioniamo come segue:

∗ La *degree centrality* si definisce come: $w_i^- = \sum_{j \in V} W_{ji}$. Poichè nel nostro caso il grafo è tale che $w_i^- = w_i^+ = w_i$, la *degree centrality* dell'i-esimo possiamo direttamente calcolarla come la somma degli elementi dell'i-esima riga o colonna essendo W = W'. Riportiamo nell'immagine seguente il calcolo della *degree centrality* in MATLAB.

```
%DEGREE CENTRALITY ----> è la somma dei pesi degli archi entranti
zi = sum(W,1)';
disp("DEGREE CENTRALITY")
disp("nodi e relativa centralità")
[linspace(1,n,n)' zi]
 %è il vettore che ha in posizione i-esima la degree centrality del nodo i-esimo

```

```
Command Window
DEGREE CENTRALITY
nodi e relativa centralità

ans =

    1    5
    2    5
    3    5
    4    5
    5    5
    6    6
    7    2
    8    2
    9    6
   10    1
   11    1
   12    1
   13    1
   14    1
```

L'i-esima riga dell'output contiene nella prima colonna l'i-esimo nodo e nella seconda la *degree centrality* relativa all' i-esimo nodo.

∗ Pochè il grafo in questione è fortemente connesso, *l'eigenvector centrality* è l'unico autovettore non-negativo normalizzato di W' corrispondente al suo autovalore dominante λ_w , che e reale e non-negativo. Quindi calcoliamo gli autovalori e determiniamo W ovvero l'autovalore dominante. Dopodichè ne ricavo l'autovettore relativo e lo normalizzo. Riportiamo nell'immagine seguente il calcolo della *eigenvector centrality* in MATLAB.

```
%EIGENVECTOR CENTRALITY
lambda_vet = eig(W');
lambda_w = max(abs(eig(W')));
[autovettori, autovalori] = eig(W');

 disp("EIGENVECTOR CENTRALITY")
 lambda_vett = autovettori(:,n)/sum(autovettori(:,n));
 % -> l'ultima colonna della matrice "autovettori" è l'autovettore relativo all'autovalore 1
 [linspace(1,n,n)', lambda_vett]
```

```
Command Window

EIGENVECTOR CENTRALITY

ans =

   1.0000    0.1583
   2.0000    0.1583
   3.0000    0.1583
   4.0000    0.1583
   5.0000    0.1583
   6.0000    0.1639
   7.0000    0.0340
   8.0000    0.0071
   9.0000    0.0018
  10.0000    0.0003
  11.0000    0.0003
  12.0000    0.0003
  13.0000    0.0003
  14.0000    0.0003
```

L'i-esima riga dell'output contiene nella prima colonna l'i-esimo nodo e nella seconda la *l'eigenvector centrality* relativa all' i-esimo nodo.

∗ Il grafo G è bilanciato e poichè in un grafo bilanciato l' *invariant distribution centrality* coincide con la *degree centrality*, ma normalizzata, possiamo usare i risultati del punto a) e verificare che $\pi = P'\pi$, con $\pi = z / \sum_{i=1}^{n} z_i$, dove z è il vettore che in posizione i-esima contiene la *degree centrality* del nodo i. Riportiamo nell'immagine seguente il calcolo dell' *invariant distribution centrality* in MATLAB.

```
58    disp("INVARIANT DISTRIBUTION CENTRALITY")
59    pi = zi/sum(zi);
60    [linspace(1,n,n)' pi];
61
62    %verifica
63    D = diag(zi);
64    P = inv(D)*W;
65    [linspace(1,n,n)' pi P'*pi]
66
67
```

Command Window

```
ans =

    1.0000    0.1087    0.1087
    2.0000    0.1087    0.1087
    3.0000    0.1087    0.1087
    4.0000    0.1087    0.1087
    5.0000    0.1087    0.1087
    6.0000    0.1304    0.1304
    7.0000    0.0435    0.0435
    8.0000    0.0435    0.0435
    9.0000    0.1304    0.1304
   10.0000    0.0217    0.0217
   11.0000    0.0217    0.0217
   12.0000    0.0217    0.0217
   13.0000    0.0217    0.0217
   14.0000    0.0217    0.0217
```

L'i-esima riga dell'output contiene nella prima colonna l'i-esimo nodo, nella seconda la *l'invariant distribution centrality* relativa all' i-esimo nodo e l'ultima colonna è una verifica del fatto che vale la proprietà che $\pi_i = \sum_{j=1}^{n} P'_{ji} * \pi_i$ per un grafo bilanciato.

Guardando il grafo in Figura 3, possiamo da subito notare che i nodi n_6 ed n_9 sono quelli con più archi entranti ed uscenti, mentre, poichè $n_{10}, n_{11}, n_{12}, n_{13}, n_{14}$ sono dei *sink* collegati ad un unico nodo, ci aspettiamo che abbiano una centralità più bassa; i nodi n_8 ed n_7 ci aspettiamo abbiano una centralità intermedia, ma comunque più bassa di n_1,n_2,n_3, n_4 ed n_5 che sono *intercollegati tra loro*.

```
73    disp("Considerazioni finali")
74    disp("nodi degree centrality eigenvector centrality invariant distribution centrality")
75    [ linspace(1,n,n)' zi lambda_vett pi]
76
77
```

Command Window

```
ans =

    1.0000    5.0000    0.1583    0.1087
    2.0000    5.0000    0.1583    0.1087
    3.0000    5.0000    0.1583    0.1087
    4.0000    5.0000    0.1583    0.1087
    5.0000    5.0000    0.1583    0.1087
    6.0000    6.0000    0.1639    0.1304
    7.0000    2.0000    0.0340    0.0435
    8.0000    2.0000    0.0071    0.0435
    9.0000    6.0000    0.0018    0.1304
   10.0000    1.0000    0.0003    0.0217
   11.0000    1.0000    0.0003    0.0217
   12.0000    1.0000    0.0003    0.0217
   13.0000    1.0000    0.0003    0.0217
   14.0000    1.0000    0.0003    0.0217
```

In questa figura troviamo: nella prima colonna i nodi e poi degree centrality, eigenvector centrality e invariant distribution centrality rispettiva mente nella seconda, nella terza e nella quarta colonna

Confrontando i valori delle tre colonne di centralità tra di loro possiamo notare come nella *degree centrality* non si tenga conto di quanto siano centrali e quindi *imoportanti* i nodi a cui l'i-esimo nodo è collegato, ma alla quantità dei collegamenti, pertanto il nodo n_9 risulta come il più centrale. Poichè *l'eigenvector centrality* misura l'influenza di un determinato nodo tenendo conto anche di quanto siano a loro volta centrali i nodi a cui il nodo in questione è collegato, notiamo che n_7 ed n_8 che presentano lo stesso valore di centralità per la *degree centrality*, nell'*eigenvector centrality* hanno valori di centralità diversi: questo perchè n_8 ha legami con n_9 che è a sua volta collegato con dei *sinks* e quindi nodi poco influenti cioè con bassa centralità, mentre n_7 è legato con n_6 che è a sua volta collegato con dei nodi molto influenti. Infine per l'*invariant distribution centrality* valgono le stesse considerazioni fatte per la *degree centrality* in quanto in un grafo bilanciato queste misure di centralità risultano essere perfettamente uguali.

(*b*) Segue il codice della funzione che esegue il calcolo della Katz centrality mediante un algoritmo iterativo.

```
function [z] = katz_centrality_iterative(W,n,beta,kmax)


    u=(1/n)*ones(n,1);
    z=zeros(n,kmax);

    lambda_w=max(abs(eig(W')));
    alpha=(1-beta)/lambda_w;

    z(:,1)=u;


    i=2;
    while(i<kmax)

        z(:,i)=alpha*W'*z(:,i-1)+beta*u;

        i=i+1;

    end


    z=z(:,i-1);
```

Applicando questa funzione al nostro grafo G, con $\beta = 0.15$ e $k_{max} = 20$ ottieniamo il seguente output:

```
26    beta=0.15;
27    kmax=20;
28
29    [z_kc] = katz_centrality_iterative(W,n,beta,kmax);
30    disp("Katz Centrality")
31    [linspace(1,n,n)' z_kc]
32
33
34
```

Command Window

```
ans =

    1.0000    0.0729
    2.0000    0.0729
    3.0000    0.0729
    4.0000    0.0729
    5.0000    0.0729
    6.0000    0.0768
    7.0000    0.0270
    8.0000    0.0198
    9.0000    0.0269
   10.0000    0.0153
   11.0000    0.0153
   12.0000    0.0153
   13.0000    0.0153
   14.0000    0.0153
```

Katz centrality sul grafo G con $\beta = 0.15$

(*c*) Segue il codice della funzione che esegue il calcolo della Page-rank centrality mediante un algoritmo distribuito.

```
function [z] = page_rank_iterative(W,n,beta,kmax)


w=sum(W,1)';

D=diag(1./w);

P=(D)*W;


u=(1/n)*ones(n,1);
z=zeros(n,kmax);

for i=1:n
    z(i, 1) = u(i);
end

for k=2:kmax

for i=1:n

    for j=1:n

        z(i,k)= z(i,k) + (P(j,i)*z(j,k-1));

    end

    z(i,k)=z(i,k)*(1-beta) + beta*u(i);

end

end


z=z(:,kmax);
```

Applicando questa funzione al nostro grafo G, con $\beta = 0.15$ e $k_{max} = 20$ ottieniamo il seguente output:

```
22    beta=0.15;
23    kmax=20;
24    [z_pr] = page_rank_iterative(W,n,beta,kmax);
25    disp("Katz Centrality")
26    [linspace(1,n,n)' z_pr]
27
```

Command Window

```
Katz Centrality

ans =

    1.0000    0.0765
    2.0000    0.0765
    3.0000    0.0765
    4.0000    0.0765
    5.0000    0.0765
    6.0000    0.0969
    7.0000    0.0512
    8.0000    0.0607
    9.0000    0.2115
   10.0000    0.0395
   11.0000    0.0395
   12.0000    0.0395
   13.0000    0.0395
   14.0000    0.0395
```

Page-rank centrality sul grafo G con $\beta = 0.15$

(*d*) La Katz centrality, che si calcola come

$$z = \frac{1-\beta}{\lambda_w} W' z + \beta\mu$$

attribuisce un'alta centralità a nodi con tanti in-neighbours, al variare di β nell'intervallo $[0, 1)$ e fa sì che i nodi collegati con nodi dotati di un'alta centralità siano i più influenti. Ecco perchè nella Katz centrality il nodo n_6 è quello con la centralità maggiore; sebbene anche n_9 sia uno dei nodi con più archi, la sua Katz centrality resta minore di quella di n_6 in quanto i nodi a cui esso è collegato sono poco influenti ovvero con pochi collegamenti.

Al contrario, la Page-rank centrality, che si calcola come:

$$z = (1-\beta) P' z + \beta\mu$$

fa in modo che se un nodo ha tanti *links* possiede molta centralità se è collegato a pochi altri nodi e, viceversa, poca centralità se i nodi a cui è collegato, sono a loro volta collegati a tanti altri nodi. Ecco perchè in questo caso è proprio n_9 ad avere una centralità più alta: possiamo dire che la Page-rank centrality tiene conto non solo di quanti *links* abbia un nodo, ma anche dell'esclusività di questi collegamenti.