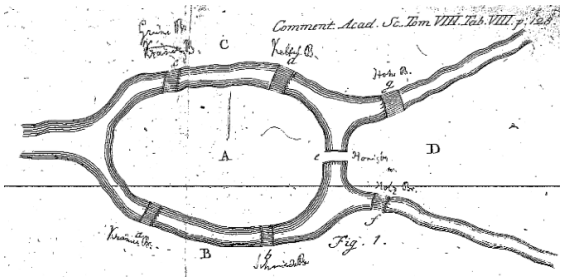
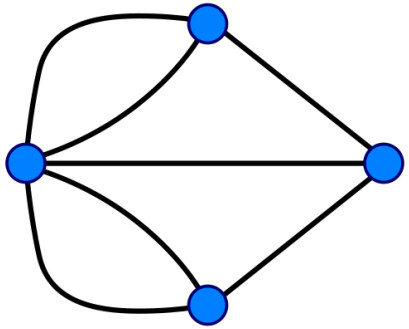




Algoritmica grafurilor

XII. Puncte de articulatie, punti, componente biconectate si masuri de calitate

Mihai Suciu

Facultatea de Matematică și Informatică (UBB)
Departamentul de Informatică

Mai, 23, 2018

Continut

Puncte de articulație și punți

Definiție 12.1

fie $G = (V, E)$ un graf neorientat și $u \in V$ un vârf oarecare din graf. Vârful u este ***punct de articulație*** *al grafului G dacă există cel puțin două vârfuri $x, y \in V, x \neq y, x \neq u,$ și $y \neq u$, astfel încât orice lanț $x \rightsquigarrow y$ trece prin u.*

Definiție 12.2

fie $G = (V, E)$ un gaf neorientat și $(u, v) \in E$ o muchie oarecare din graf. Muchia (u, v) este ***punte*** *în graful G dacă există cel puțin două vârfuri $x, y \in V, x \neq y$, astfel încât orice lanț $x \rightsquigarrow y$ în G conține muchia (u, v).*

Puncte de articulație

Teorema 12.1

Fie $G = (V, E)$ un graf neorientat si $u \in V$ un vârf oarecare din G. Vârful u este ***punct de articulație*** *în G dacă și numai dacă în urma $DFS(G)$ una din proprietățile de mai jos este satisfăcută:*

a. *$u.\pi = null$ și u domină cel puțin doi subarbori*

b. *$u.\pi \neq null$ și există un vârf v descendent al lui u în $ARB(u)$ astfel încât pentru orice vârf x din $ARB(v)$ si orice arc (x, z) parcurs de DFS avem $z.d \geq u.d$.*

- $ARB(u)$ este arborele ce are rădăcina u
- $u.d$ timpul în care a fost descoperit vârful u

Puncte de articulație (II)

Algoritmul pentru determinarea unui punct de articulație urmărește teorema 12.1. În afară de proprietățile necesare *DFS*, unui vârf $u \in V$ i se atașează proprietățile:

1. $u.b = \min\{v.d | v$ descoperit pornind din u în cursul *DFS* si $v.color \neq$ alb$\}$;
2. $subarb(u)$ este numarul subarborilor dominați de u.

Puncte de articulație (III)

Există mai multe momente importante în cursul *DFS* în care $u.b$ este modificat sau vârful u este testat pentru a fi marcat ca vârf de articulație:

- în momentul descoperirii lui u, $u.b = u.d$;
- în momentul în care din u se ajunge la un succesor v al lui u și $v.color = alb$, $u.b = \min\{u.b, v.d\}$;
- în momentul în care dintr-un succesor v al lui u se revine în u, $u.b = \min\{u.b, v.b\}$, dacă $u.b \geq u.d$ și $u.\pi \neq null$ atunci u este un vârf de articulație - cazul (b) din teorema 12.1;
- în momentul în care din u se revine în ciclul principal al *DFS*: dacă u domină doi subarbori, atunci u este punct de articulație - cazul (a) din teorema 12.1.

Puncte de articulație (IV)

- Se marchează cu $u.articlulatie = 1$ vârfurile de articulație din graf.
- Pentru a verifica ușor numărul de subarbori *DFS* al unui vârf se introduce proprietatea $u.subarb$, $\forall v \in V$, inițial $u.subarb = 0$. Acest atribut crește pentru fiecare succesor alb al lui u.

Puncte de articulație - algoritm

ARTICULATII(G)

```
1:  timp = 0
2:  for u ∈ V do
3:      u.color = alb
4:      u.π = NIL
5:      u.subarb = 0
6:      u.articulatie = 0
7:  for u ∈ V do
8:      if u.color = alb then
9:          EXPLORARE(u)
10:         if u.subarb > 1 then
11:             u.articulatie = 1
```

Puncte de articulație - algoritm (II)

EXPLORARE(u)

1: $u.d = u.b = timp++$
2: $u.color = gri$
3: **for** $v \in succs(u)$ **do**
4: **if** $u.c = alb$ **then**
5: $v.\pi = u$
6: $u.subarb++$
7: $EXPLORARE(v)$
8: $u.b = \min\{u.b, v.b\}$
9: **if** $u.\pi \neq NIL \wedge v.b \geq u.d$ **then**
10: $u.articulatie = 1$
11: **else**
12: $u.b = \min\{u.b, v.d\}$

Punti

Teorema 12.2

fie $G = (V, E)$ un graf neorientat si $(u, v) \in E$ o muchie oarecare din graf. Muchia (u, v) este punte in G dacă și numai dacă în urma $DFS(G)$ una din proprietățile de mai jos este satisfăcută:

1. *v este descendentul direct al lui u în $ARB(u)$ și nu există nici un descendent $DFS(G)$ al lui v care să formeze arce inverse cu vreun vârf $z, z.d \leq u.d$.*
2. *u este descendent direct al lui v în $ARB(v)$ și nu există nici un descendent $DFS(G)$ al lui u care să formeze arce inverse cu vreun vârf $z, z.d \leq u.d$.*

Punti (II)

Algoritmul pentru detectarea muchiilor punți străbate în adâncime graful și verifică următoarele proprietăți simetrice impuse unei muchii (u, v) pentru a fi punte:

1. $v.\pi = u$ și parcurgerea în adâncime a grafului pornind din v - străbătând muchii diferite de (u, v) - nu descoperă vârful u sau vârfuri explorate inanintea lui u, $u.d < v.b$. În acest caz, pentru a ajunge de la u la v sau la orice alt vârf descoperit din v, nu există alte lanțuri decât cele care trec prin (u, v).
2. $u.\pi = v$ și parcurgerea în adâncime a grafului pornind din u - străbătând muchii diferite de (u, v) - nu descoperă vârful v sau vârfuri explorate inanintea lui v, $v.d < u.b$.

Punti (III)

- În cursul $DFS(G)$ parcurgerea muchiilor grafului G trebuie efectuată într-un sens;
- dacă v este un vârf adiacent lui u și culoarea lui v este ablă și muchia (u, v) este străbătută de algoritm în sensul $u \rightarrow v$ atunci parcurgerea în sens invers trebuie blocată;
- altfel dacă arcul este străbătut ulterior in sensul $v \rightarrow u$ vârful u este descoperit ca vârf de culoare gri (muchia (v, u) este arc invers) iar valoarea $v.b$ este actualizată la $\min\{u.b, v.b\} \rightarrow$ va satisface $v.b = u.d$ chiar dacă pentru orice alt vârf x la care se ajunge din v avem $x.b > u.d$;
- în acest caz muchia (u, v) nu este recunoscută ca punte deși este punte.

Punti (IV)

- Pentru a bloca parcurgerea în sens invers a muchiei (u, v) algoritmul se folosește de $v.\pi$;
- la prima descoperire a vârfului v din u pe muchia (u, v) se stabilește $v.\pi = u$;
- la avansul ulterior din v se evită orice muchie (v, x) pentru care $v.\pi = x$;
- complexitatea algoritmului este aceeași ca și pentru *DFS* sau *ARTICULATII* : $\Theta(V + E)$

Punti - algoritm

- Fiecărui vârf din graf i se atașează atributul *punte* astfel $u.punte = 1$ înseamnă că muchia $(u, u.\pi)$ este punte în G.

PUNTI(G)

1: **for** $u \in V$ **do**
2: $\quad u.color = alb$
3: $\quad u.\pi = NIL$
4: $\quad u.punte = 0$
5: $timp = 0$
6: **for** $u \in V$ **do**
7: $\quad$ **if** $u.color = alb$ **then**
8: $\quad\quad$ *EXPLORARE_PUNTI(u)*

Punti - algoritm (II)

EXPLORARE_PUNTI(u)

1: $u.d = u.b = timp++$
2: $u.color = gri$
3: **for** $v \in succs(u)$ **do**
4: **if** $u.color = alb$ **then**
5: $v.\pi = alb$
6: $EXPLORARE\_PUNTI(v)$
7: $u.b = \min\{u.b, v.b\}$
8: **if** $v.b > u.d$ **then**
9: $v.punte = 1$
10: **else**
11: **if** $u.\pi \neq v$ **then**
12: $u.b = \min\{u.b, v.d\}$

Componente biconectate

Definiție 12.3

fie $G = (V, E)$ un graf neorientat. O ***componentă biconectată*** *(sau bicomponentă) a lui G este un subgraf maximal $G_b = (V_b, E_b)$ cu $V_b \subseteq V$ și $E_b \subseteq E$ care nu conține puncte de articulație.*

sau

Definiție 12.4

fie $G = (V, E)$ un graf neorientat. O ***componentă biconectată*** *(sau bicomponentă) a lui G este un subgraf maximal $G_b = (V_b, E_b)$ cu $V_b \subseteq V$ și $E_b \subseteq E$ astfel încât pentru orice muchii α și β din E_b există un ciclu simplu care contine muchiile α și β.*

Componente biconectate (II)

Teorema 12.3

fie $G = (V, E)$ un graf neorientat și u un vârf nesingular din G ($succs(u) \neq \emptyset$). Vârful u este vârf de start al unei bicomponente a lui G dacă și numai dacă în urma $DFS(G)$ există cel puțin un subarbore $ARB(v)$ dominat de u astfel încât pentru orice muchie (x, z) - cu x în $ARB(v)$ - descoperit în cursul $DFS(G)$ avem $u.d \leq z.d$.

Componente biconectate - algoritm

BICOMPONENTE(G)

1: **for** $u \in V$ **do**
2: $u.color = alb$
3: $timp = 0$
4: $componente = \emptyset$
5: **for** $u \in V$ **do**
6: **if** $u.color = alb$ **then**
7: **if** $sucs(u) \neq \emptyset$ **then**
8: $componente = componente \cup$ EXPLORARE_BICOMP(u)
9: **else**
10: $u.color = negru$
11: $componente = componente \cup \{u\}$
12: **return** componente

Componente biconectate - algoritm (II)

EXPLORARE_BICOMP(u)

1: $u.d = u.b = timp++$
2: $u.color = gri$
3: $componente_u = \emptyset$
4: **for** $v \in succs(u)$ **do**
5: **if** $v.color = alb$ **then**
6: $componente_u = componente_u \cup$ EXPLORARE_BICOMP(v)
7: $u.b = \min\{u.b, v.b\}$
8: **if** $u.d \leq v.b$ **then**
9: $componente_u = componente_u \cup \{COLECTARE(u, v)\}$
10: **else**
11: $u.b = \min\{u.b, v.d\}$
12: **return** $componente_u$

Componente biconectate - algoritm (III)

COLECTARE(start, vecin)

1: $start.color = negru$
2: $componenta = PARCURGERE \cup \{start\}$
3: $start.color = gri$
4: **return** *componenta*

PARCURGERE(varf)

1: $varf.color = negru$
2: $componenta = \{varf\}$
3: **for** $v \in succs(varf)$ **do**
4: **if** $v.color = gri$ **then**
5: $componenta = componenta \cup PARCURGERE(v)$
6: **return** *componenta*

Măsuri în grafuri

- O statistică a unui graf este o valoare numerică care caracterizează un graf.
- Exemple de astfel de valori: ordinul, dimensiunea unui graf dar și măsuri mai complexe, cum ar fi diametrul și coeficientul de grupare (*clustering coefficient*).
- Aceste statistici permit caracterizarea și analiza unui graf. Ele pot fi utilizate pentru a compara, clasifica grafuri, pentru a detecta anomalii în graf, etc.
- Statisticile pot fi utilizate pentru a mapa un graf într-un spațiu numeric simplu, în care pot fi aplicate mai multe metode statistice standard.

Măsuri în grafuri

Ca și măsuri în grafuri putem defini:

1. ordinul, dimensiunea
2. gradul minim, mediu, maxim
3. reciprocitatea (*reciprocity*)
4. incărcarea (*fill*)
5. negativitatea (*negativity*)
6. LLC
7. numărul de lanțuri de lungime 2 (*wedge count*), grafelor ghiară, K_3, grafelor pătrat, *4-tour*,
8. coeficientul *power law*, *gini*
9. distribuția relativă a gradului unui vârf
10. coeficientul de grupare (*clustering coefficient*)
11. diametrul
12. *Preferential attachment*

Diametrul unui graf

Putem defini excentricitatea unui vârf într-un graf ca și lungimea maximă a drumului minim

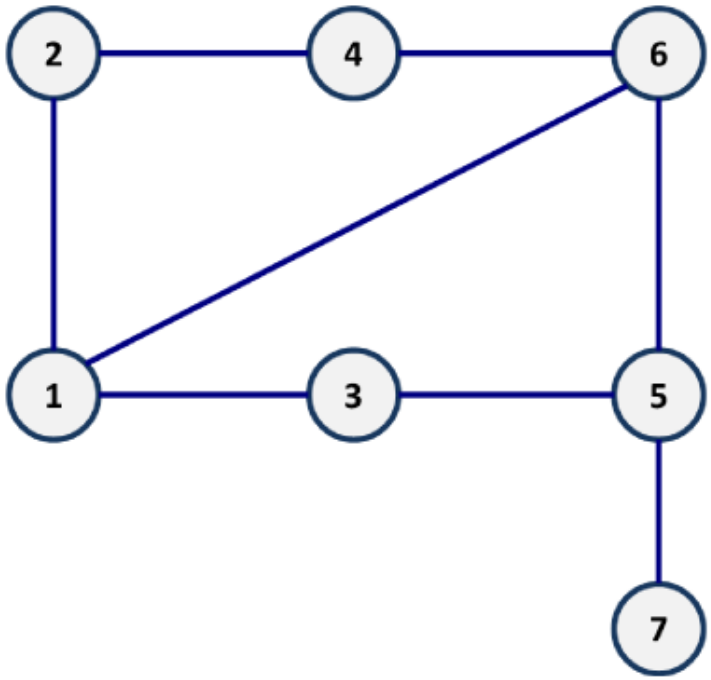
$$\epsilon(u) = \max_{v \in V} \delta(u, v)$$

unde δ este drumul minim între u și v.
Diametrul unui graf se poate defini:

$$d = \max_{u \in V} \epsilon(u) = \max_{u,v \in V} \delta(u, v)$$

Diametrul unui graf - exemplu

Care este diametrul acestui graf?



Diametrul unui graf - exemplu (II)

v	1	2	3	4	5	6	7
1	0	1	1	2	2	1	3
2	1	0	2	1	3	2	4
3	1	2	0	3	1	2	2
4	2	1	3	0	2	1	3
5	2	3	1	2	0	1	1
6	1	2	2	1	1	0	2
7	3	4	2	3	1	2	0

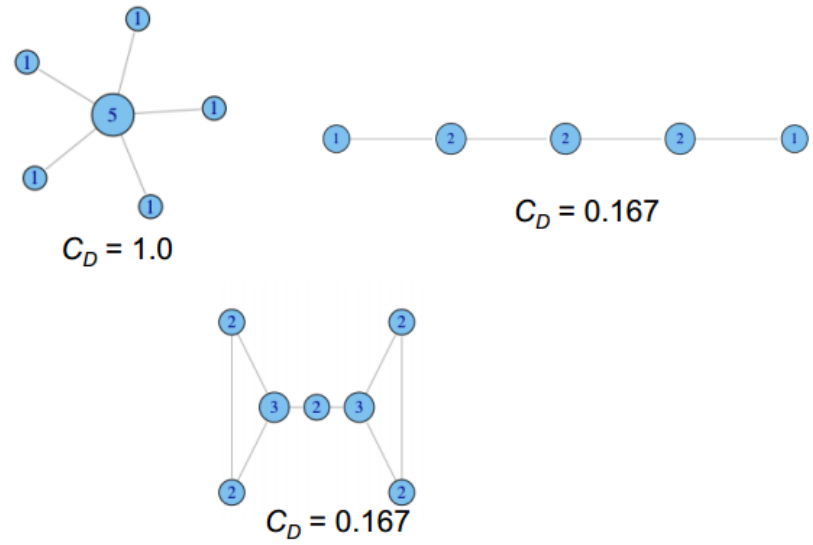
Coeficientul de centralitate - Freeman

Măsură a importanței pe baza gradurilor vârfurilor din graf.

Freeman

$$C_D = \frac{\sum_{i=1,N}[C_D(n^*) - C_D(i)]}{(N-1)(N-2)},$$

unde $C_D(n^*)$ este gradul cel mai mare din graf.



Betweenness centrality

Cât de central este un vârf.

Betweenness centrality

$$C_B(i) = \sum_{j<k} g_{jk}(i)/g_{jk},$$

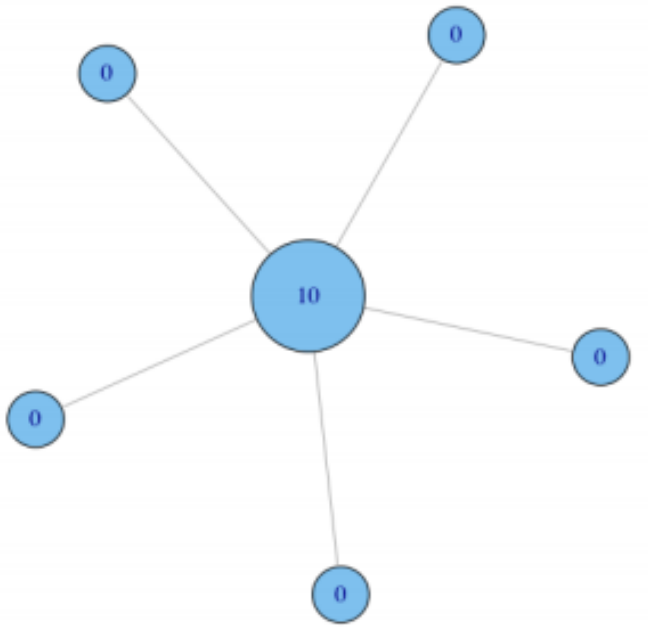
unde g_{jk} este numărul drumurilor cele mai scurte care leagă vârfurile j și k, $g_{jk}(i)$ este numărul drumurilor cele mai scurte care leagă vârfurile j și k și conțin vârful i.

Normalizare

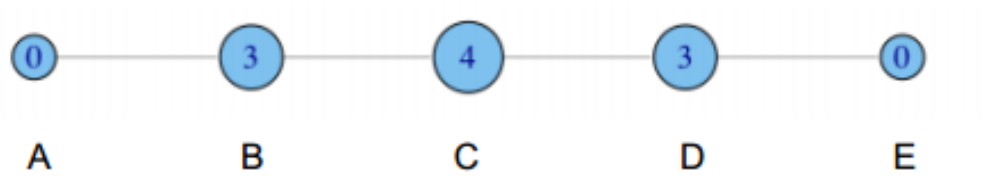
$$C'_B(i) = C_B(i)/[(N-1)(N-2)/2].$$

- normalizarea se face împărțind la numărul tuturor drumurilor posibile dacă se scoate vârful i

Betweenness centrality - exemplu

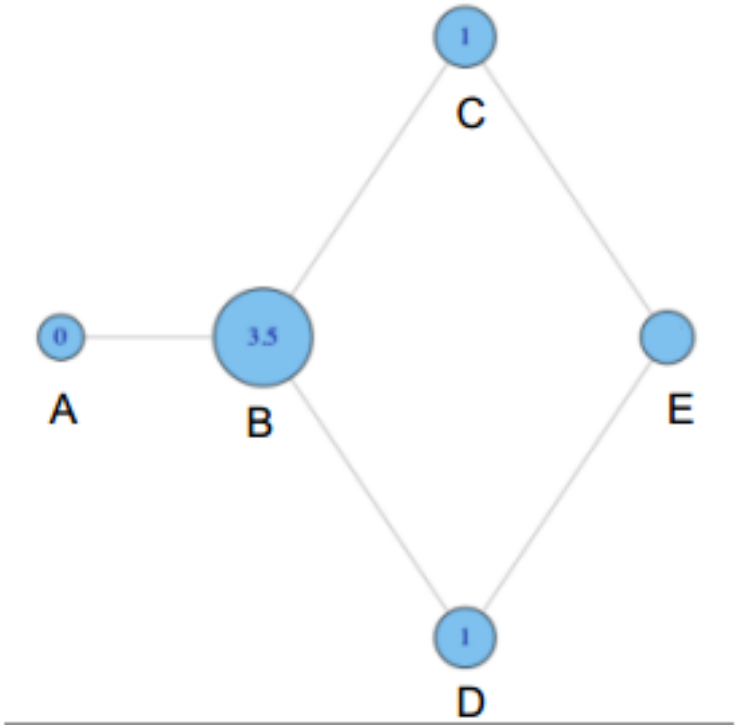


Betweenness centrality - exemplu (II)

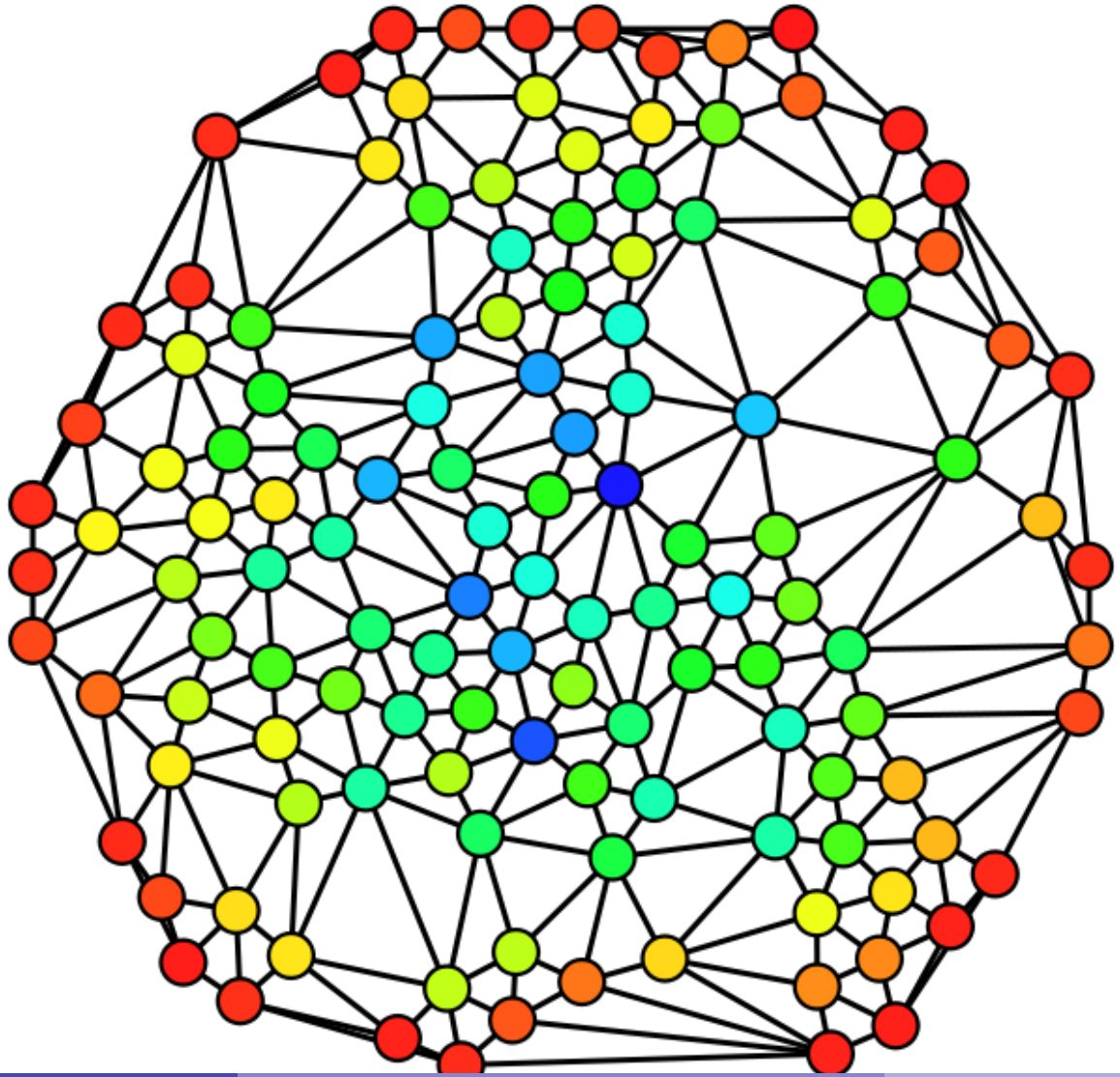


$B : (A, C), (A, D), (A, E)$
$C : (A, D), (A, E), (B, D), (B, E)$

Betweenness centrality - exemplu (III)



Betweenness centrality - exemplu (IV)



Closeness centrality

"Distanța" unui vârf față de celelalte vârfuri.

Closeness centrality

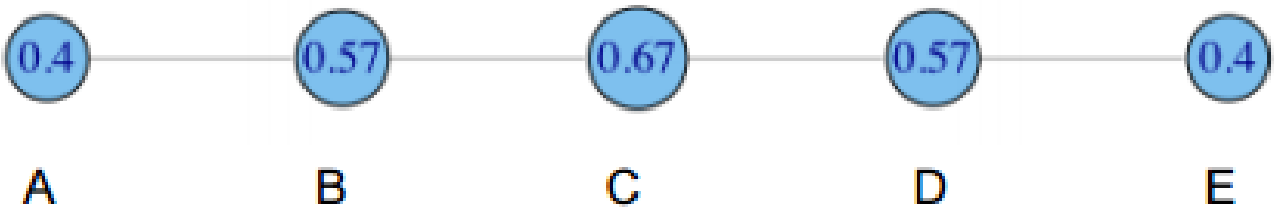
$$C_c(i) = [\sum_{j=1,N} d(i,j)]^{-1}$$

unde $d(i,j)$ este distanța între vârfurile i si j.

Normalizare

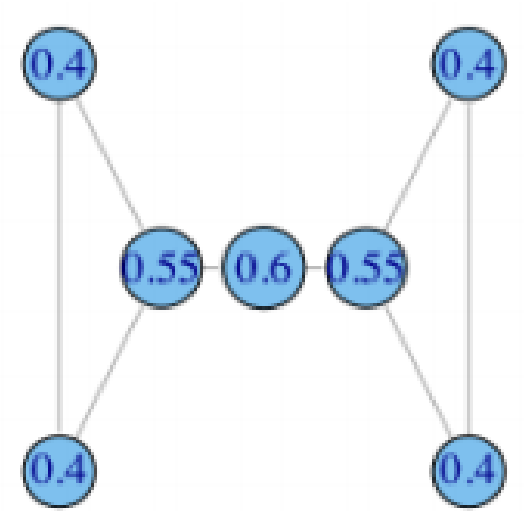
$$C'_c(i) = [\frac{\sum_{i=1,N} d(i,j)}{N-1}]^{-1}$$

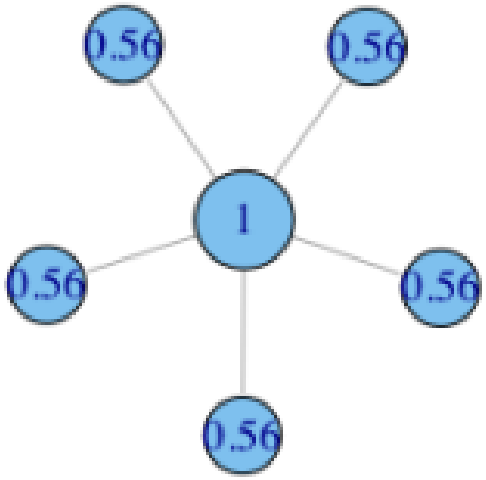
Closeness centrality - exemplu



$$C_c'(A) = \left[\frac{\sum_{j=1}^{N} d(A,j)}{N-1} \right]^{-1} = \left[\frac{1+2+3+4}{4} \right]^{-1} = \left[\frac{10}{4} \right]^{-1} = 0.4$$

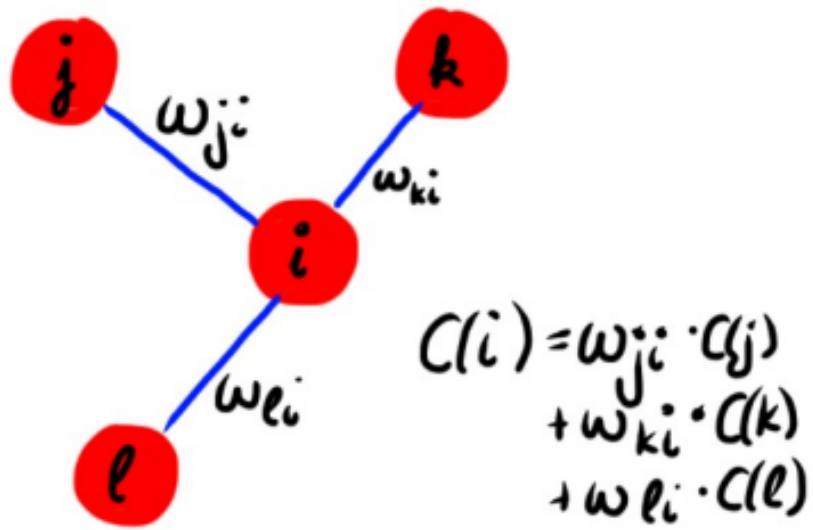
Closeness centrality - exemplu (II)





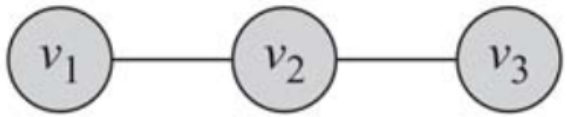
Eigencentrality (Eigenvector centrality)

O măsură a influenței unui vârf în graf.



O generalizare a măsurii de centralitate în care se ține seaman și de vecini.

Eigencentrality - exemplu



Eigencentrality - exemplu (II)

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

$$\lambda \mathbf{C}_e = A\mathbf{C}_e$$

$$(A - \lambda I)\mathbf{C}_e = 0$$

$$\mathbf{C}_e = [u_1\ u_2\ u_3]^T,$$

$$\begin{bmatrix} 0-\lambda & 1 & 0 \\ 1 & 0-\lambda & 1 \\ 0 & 1 & 0-\lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

$$\mathbf{C}_e \neq [0\ 0\ 0]^T,$$

$$det(A - \lambda I) = \begin{vmatrix} 0-\lambda & 1 & 0 \\ 1 & 0-\lambda & 1 \\ 0 & 1 & 0-\lambda \end{vmatrix} = 0,$$

Eigencentrality - exemplu (III)

$$(-\lambda)(\lambda^2 - 1) - 1(-\lambda) = 2\lambda - \lambda^3 = \lambda(2 - \lambda^2) = 0.$$

$$(-\sqrt{2}, 0, +\sqrt{2}).$$

$$\begin{bmatrix} 0-\sqrt{2} & 1 & 0 \\ 1 & 0-\sqrt{2} & 1 \\ 0 & 1 & 0-\sqrt{2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Eigencentrality - exemplu (IV)

$$\mathbf{C}_e = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1/2 \\ \sqrt{2}/2 \\ 1/2 \end{bmatrix}$$

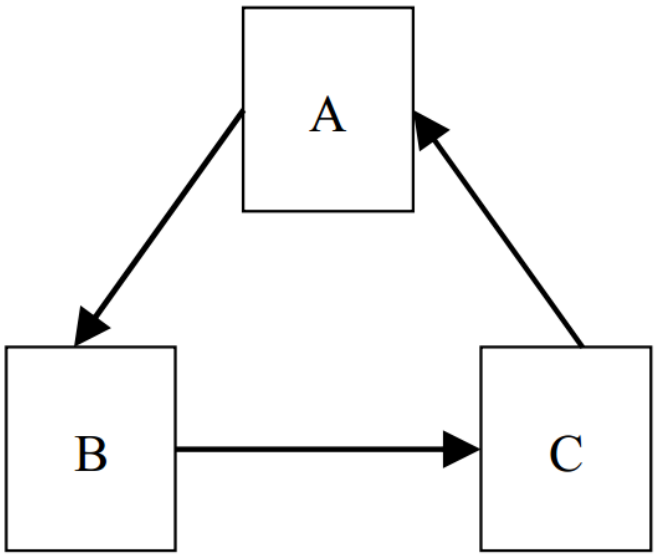
vârful C este cel mai central (important).

Page rank

$$PR(v_i) = \frac{1-d}{N} + d \sum_{v_j \in M(v_i)} \frac{PR(v_j)}{L(v_j)},$$

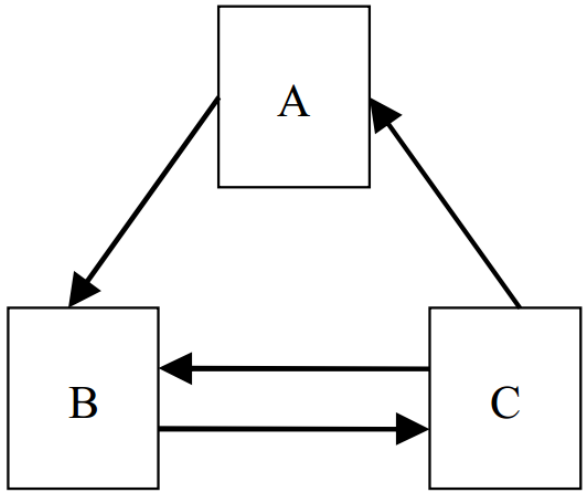
unde $M(v_j)$ este vecinătateaa vârfului v_i (arcele spre interior), $L(v_j)$ este gradul spre exterior pentru vârful v_j, d este un parametru.

Page rank - exemplu



$$PR(A) = (1-d) \times (\ 1\ /\ N\ ) + d \times (\ PR(C)\ /\ 1\ )$$
$$PR(B) = (1-d) \times (\ 1\ /\ N\ ) + d \times (\ PR(A)\ /\ 1\ )$$
$$PR(C) = (1-d) \times (\ 1\ /\ N\ ) + d \times (\ PR(B)\ /\ 1\ )$$

Page rank - exemplu (II)



$$PR(A) = (1-d) \times (1/N) + d \times (PR(C)/2)$$
$$PR(B) = (1-d) \times (1/N) + d \times (PR(A)/1 + PR(C)/2)$$
$$PR(C) = (1-d) \times (1/N) + d \times (PR(B)/1)$$