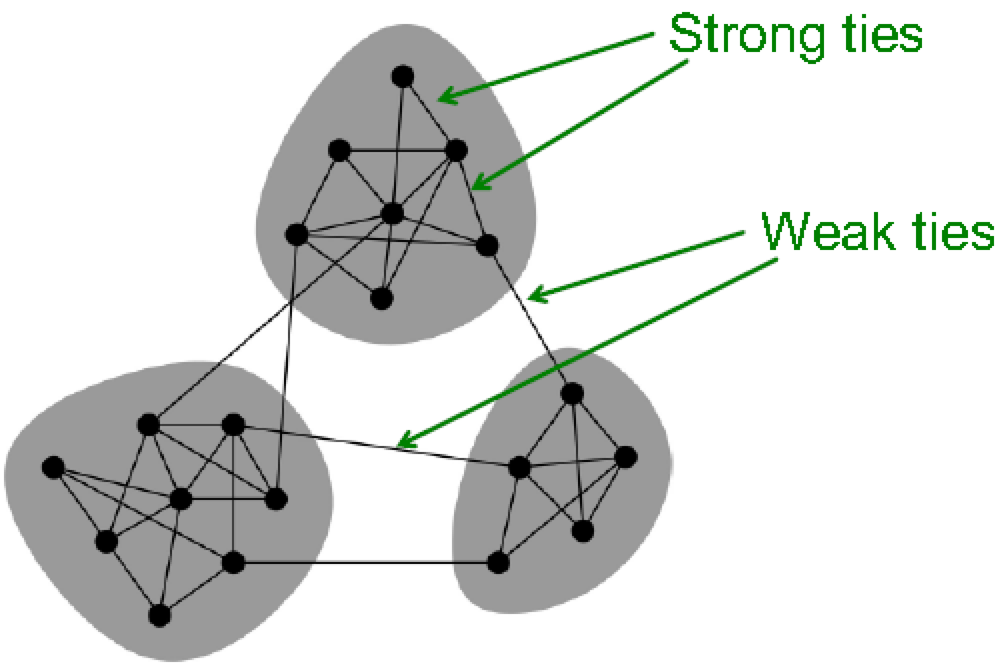
Partizionamento delle reti

Teoria di Granovetter implica la seguente visione di una rete



Ruoli di nodi / archi diversi

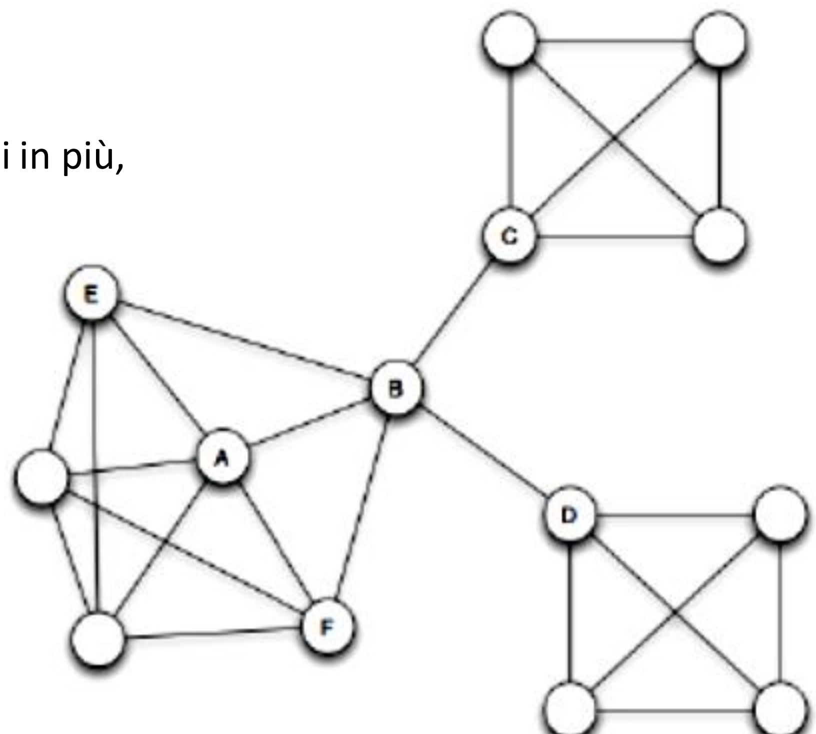
In una rete sociale

- alcuni edge sono posizionati tra due gruppi
- altri sono posizionati nel mezzo di un unico gruppo.

Analogamente

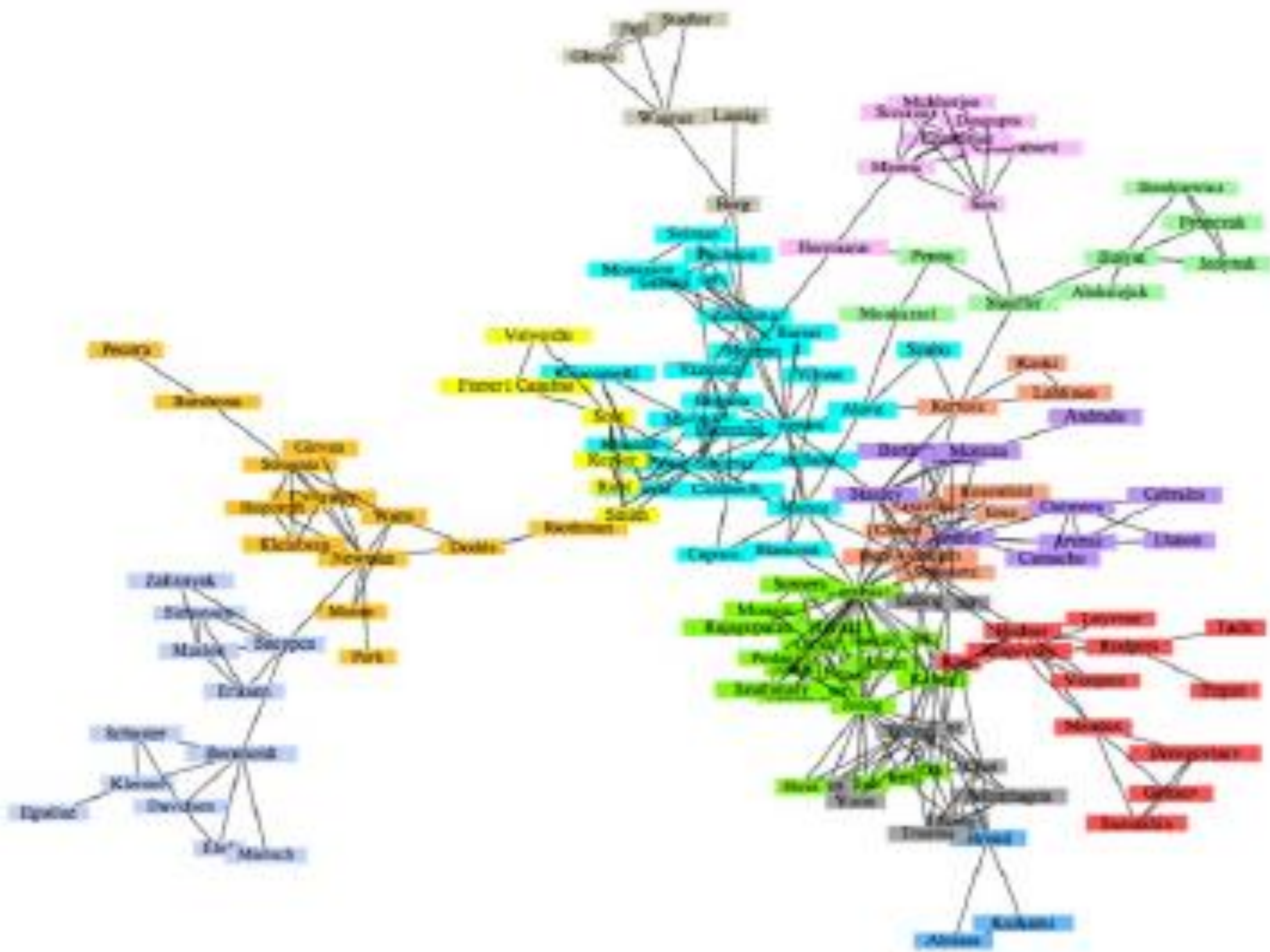
- alcuni nodi sono posizionati all'interfaccia tra vari gruppi
- altri sono posizionati nel mezzo di un unico gruppo.

- Es.
  - La posizione di B offre vantaggi rispetto ad A
  - B ha accesso anticipato alle informazioni provenienti in più, parti non interagenti della rete.



Esempio: un Co-authorship Network

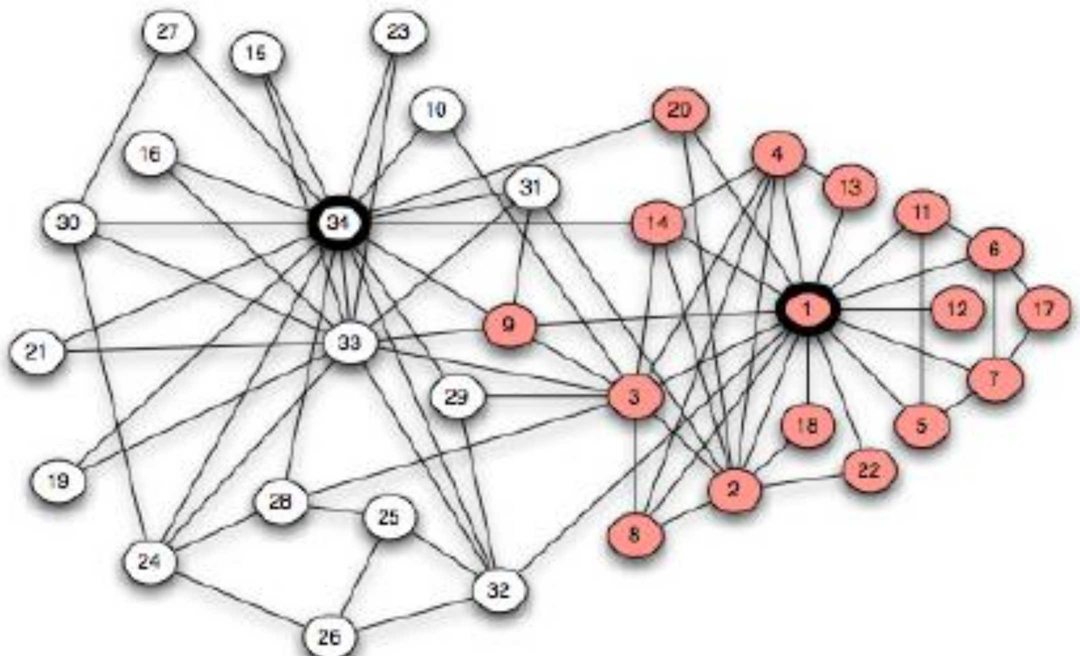
Rete delle collaborazioni tra fisici e matematici applicati che lavorano sulle reti



Partizionamento delle reti

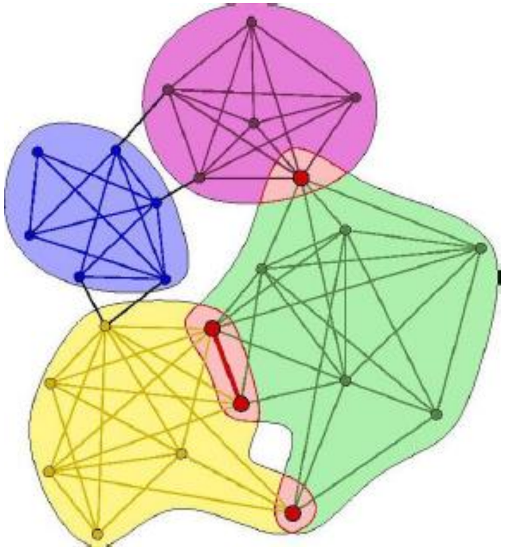
Splitting della rete Karate Club:

Potrebbero i confini dei due club essere predetti dalla struttura di rete?



Partizionamento delle reti

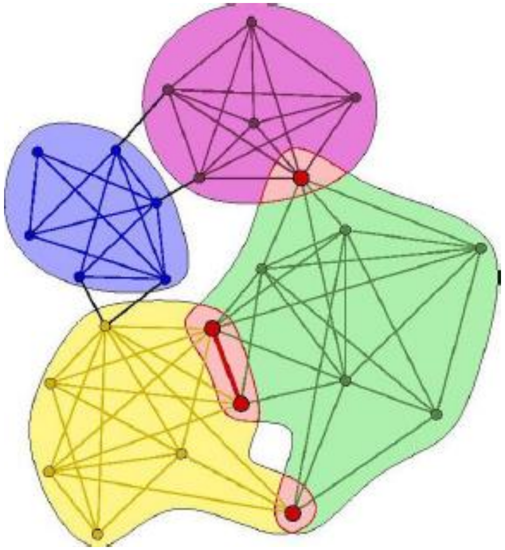
- Dalla discussione precedente ricaviamo un'idea di struttura di una rete sociale:
  - Un insieme di gruppi di nodi densamente connessi legati insieme da legami occasionali



comunità

Una **comunità** è un sottografo in cui

- i nodi membri sono molto connessi tra loro e
- la densità degli edge esistenti all'interno del gruppo è molto maggiore rispetto a quella degli edge che connettono nodi interni al gruppo con nodi esterni.

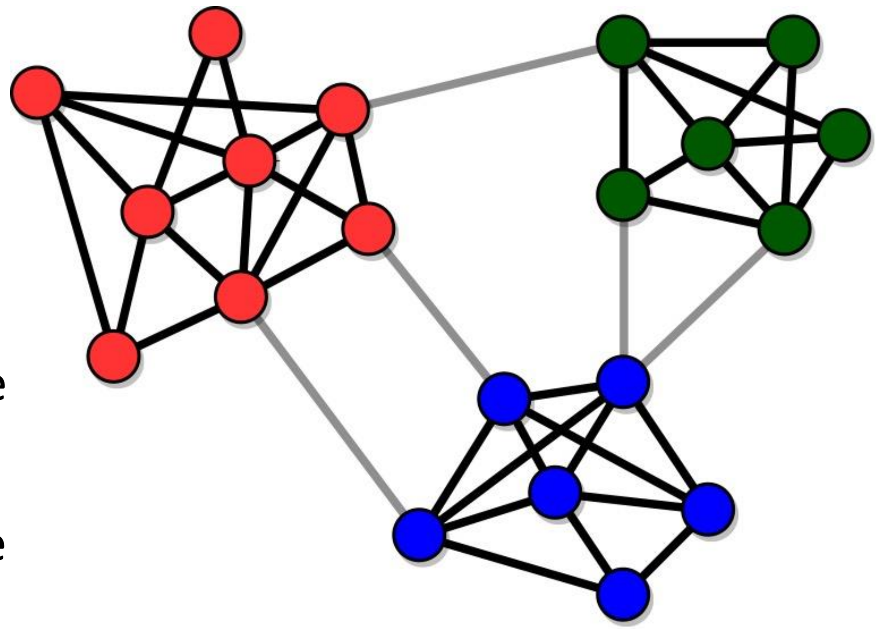


Perché studiare le comunità?

- Scoprire l'organizzazione della rete
- Identificare le caratteristiche dei nodi
- Classificare i nodi in base alla loro posizione nei cluster (centrali o di confine)

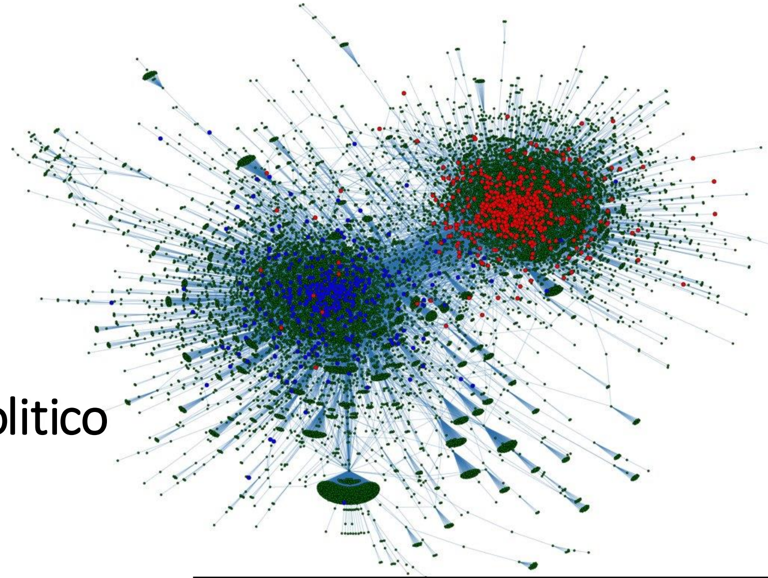
  Es. Se vogliamo bloccare epidemia/fake news è fondamentale controllare i nodi di confine

  Es. Identificare le comunità della rete è vitale per gli esperti di marketing (identificare utenti con interessi simili)



Comunità

- Gli utenti di Twitter
  con forti preferenze politiche
  tendono a seguire altri utenti
  in linea con loro e a non seguire
  gli utenti con diverso orientamento politico



Retweet di meme politici in USA prima delle elezione del 2010. Liberali(blu)-Conservatori (rosso), link= retweet

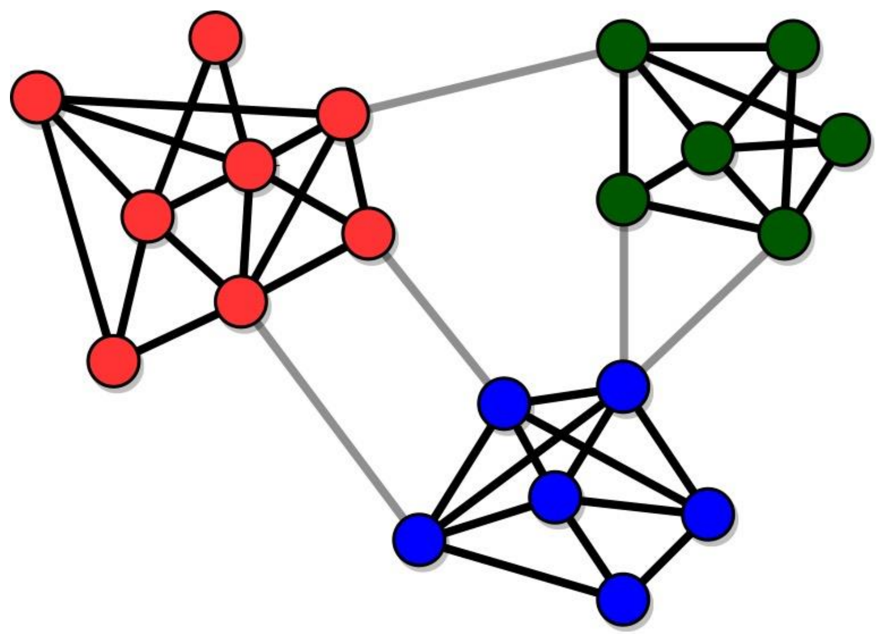
- Esempio di *echo-chamber*: contatti solo con persone con stessa idea

Comunità

Due caratteristiche principali:

Alta coesione: le comunità hanno molti collegamenti interni, quindi i loro nodi rimangono uniti

Elevata separazione: le comunità sono collegate tra loro da pochi collegamenti



Idea per partizionamento

➔ ottenere un'elevata coesione e un'elevata separazione

➔ link interni “più” dei link esterni

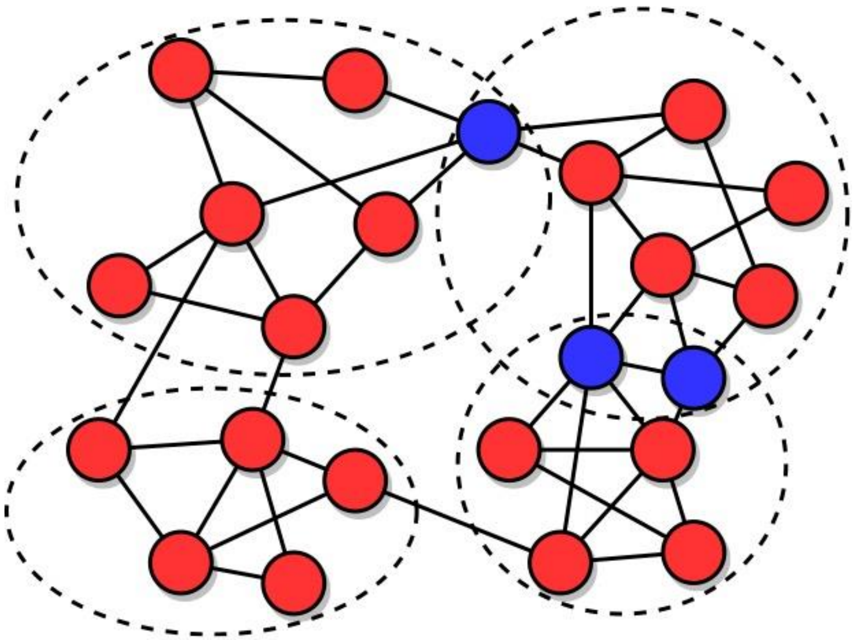
Partizionamento: Possiamo testare tutte le partizioni?

- Il numero di partizioni di n oggetti è il numero di Bell B_n
- Il numero di Bell cresce più velocemente che in modo esponenziale con n
- Conclusione: non ha senso cercare strutture comunitarie interessanti esplorando l'intero spazio delle partizioni!

n	B_n
1	1
2	2
3	5
4	15
5	52
6	203
7	877
8	4140
9	21147
10	115975
11	678570
12	4213597
13	27644437
14	190899322
15	1382958545

Osservazioni: Comunità sovrapposte

- Le comunità in molte reti reali si sovrappongono
- Una divisione di una rete in comunità sovrapposte è chiamata copertura
- Il numero di possibili coperture di una rete è di gran lunga superiore al numero di partizioni, a causa dei molti modi in cui i cluster possono sovrapporsi

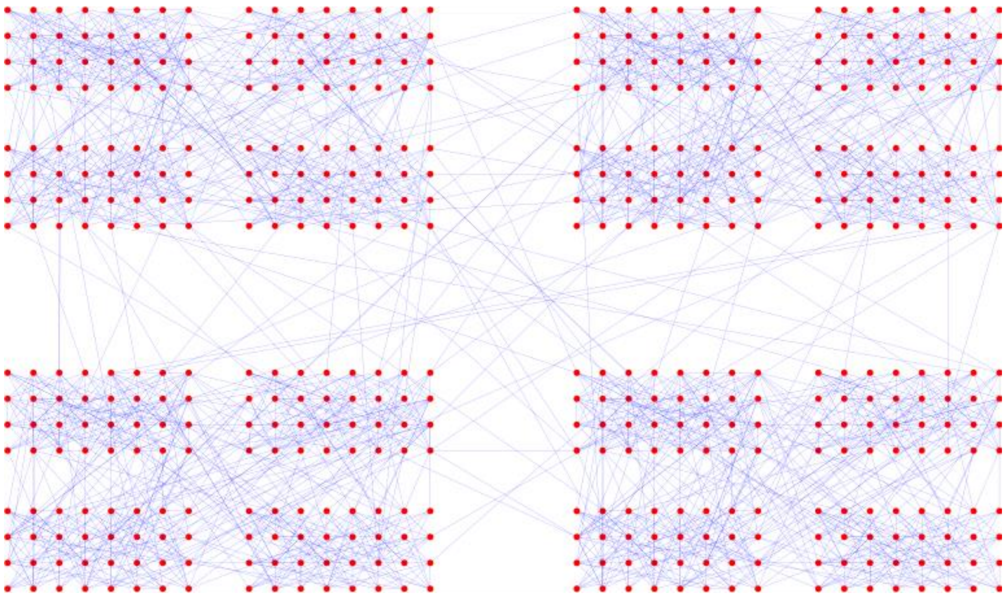


Osservazioni: Comunità gerarchiche

Se la rete ha più livelli di organizzazione,

le sue comunità potrebbero formare una gerarchia, con

comunità piccole all'interno di comunità più grandi

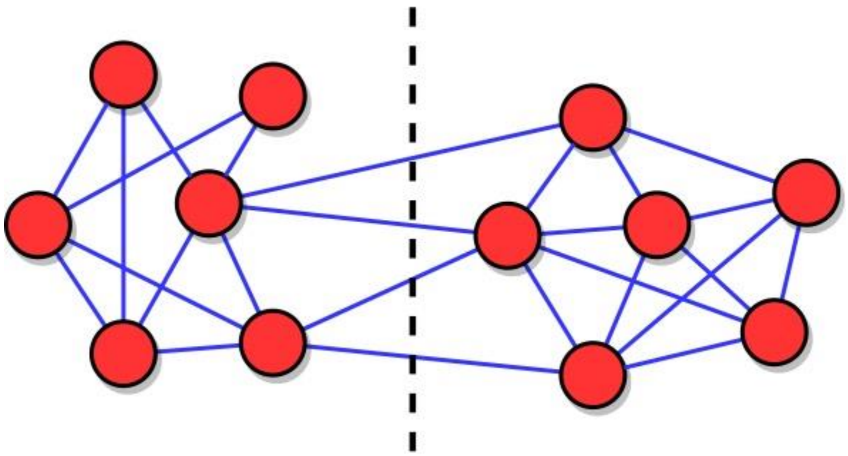


Esempio: filiali in un'azienda, a loro volta suddivise in reparti

Problema simile: Partizionamento di un grafo

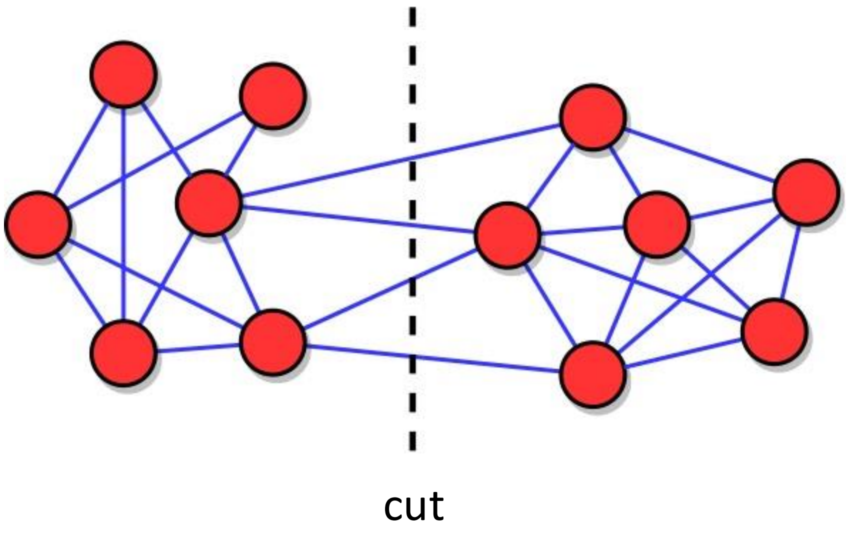
Il problema: dividere i nodi in un dato numero di

- gruppi di dimensioni predefinite, in modo tale che
- il numero di collegamenti tra i gruppi (cut/taglio) sia minimo



Problema simile: Partizionamento di un grafo

- **Bisezione:** dividere i nodi in due gruppi di uguale dimensione, in modo tale che il numero di collegamenti tra i gruppi (cut) sia minimo



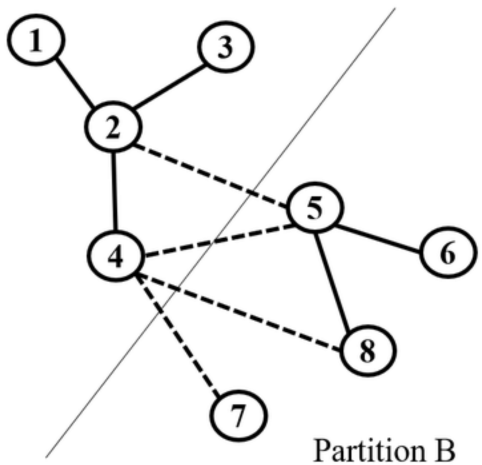
Bisezione: Algorithm di Kernighan-Lin

- Dividi in due gruppi A e B di dimensioni predefinite N_A e N_B,
  (per la bisezione $N_A = N_B$) ad es. assegnando casualmente i nodi ai gruppi

- ITERAZIONE
  1. Per ogni coppia di nodi i, j, con $i \in A$ e $j \in B$, calcola la variazione della dimensione del cut tra la partizione corrente e quella ottenuta scambiando i e j
  2. La coppia di nodi i e j che producono la massima diminuzione della dimensione del cu viene selezionata e scambiata. Questi nodi sono bloccati (non verranno più toccati)
  3. Ripetere i passi 2.1 e 2.2 fintanto che esiste uno scambio di nodi non bloccati che produce una diminuzione della dimensione del cut.

  L'iterazione produce una nuova bipartizione, che viene utilizzata come configurazione iniziale per l'iterazione successiva

- La procedura termina quando i cut delle partizioni ottenute dopo due iterazioni consecutive hanno la stessa dimensione
  (l'algoritmo non è in grado di migliorare il risultato corrente)



Partition A	E_a	I_a
1	0	1
2	1	3
3	0	1
4	3	1

Partition B	E_a	I_a
5	2	2
6	0	1
7	1	0
8	1	1

Pair	C(a,b)	G	Pair	C(a,b)	G
G_{15}	0	-1	G_{35}	0	-1
G_{16}	0	-2	G_{36}	0	-2
G_{17}	0	0	G_{37}	0	0
G_{18}	0	-1	G_{38}	0	-1
G_{25}	1	-4	G_{45}	1	0
G_{26}	0	-3	G_{46}	0	1
G_{27}	0	-1	G_{47}	1	1
G_{28}	0	-2	G_{48}	0	0

Bisezione: Algorithm di Kernighan-Lin

- **Problema**: la scelta della partizione iniziale influisce sul risultato finale. Si mostra che maggiore è la dimensione del taglio della partizione iniziale, peggiore è la soluzione finale e maggiore è il tempo per terminare

  - **Soluzione**: creare più partizioni casuali e scegliere quella con la dimensione di taglio più bassa come partizione iniziale

Bisezione: Algorithm di Kernighan-Lin

La procedura descritta è greedy
(ad ogni passaggio si cerca la partizione con cut più piccolo.

➔ l'algoritmo si blocca negli ottimali locali,
(soluzioni la cui dimensione del cut non è ottima)

L'algoritmo Kernighan-Lin è ampiamente applicato come tecnica di post-processing, per migliorare le partizioni fornite con altri metodi.
Tali partizioni possono essere utilizzate come punti di partenza per il metodo, che potrebbe restituire soluzioni con dimensioni di taglio inferiori

Partitioning: l'algoritmo può essere adattato per partizioni in più di due cluster, scambiando i nodi tra coppie di cluster

Limiti del Partitioning

- I cluster non necessariamente hanno un'elevata densità di collegamenti interni

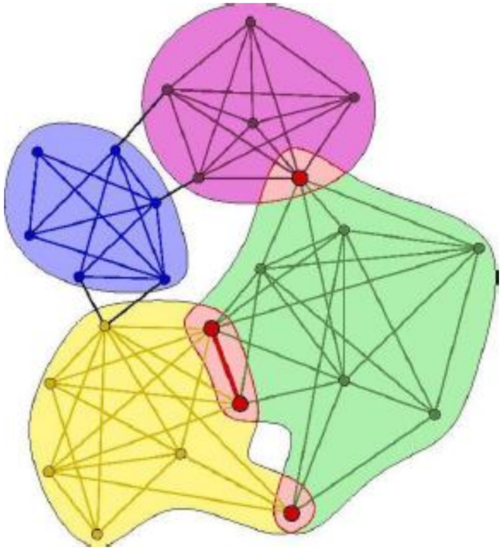
  -> i cluster trovati tramite il partizionamento del grafo non sono comunità,

- in generale Il numero di cluster deve essere fornito come input, ma di solito è sconosciuto

Community detection

- Si vuole determinare struttura naturale, e non imporre una partizione artificiale fissando il numero o la dimensione delle comunità.

- Problema algoritmico non banale

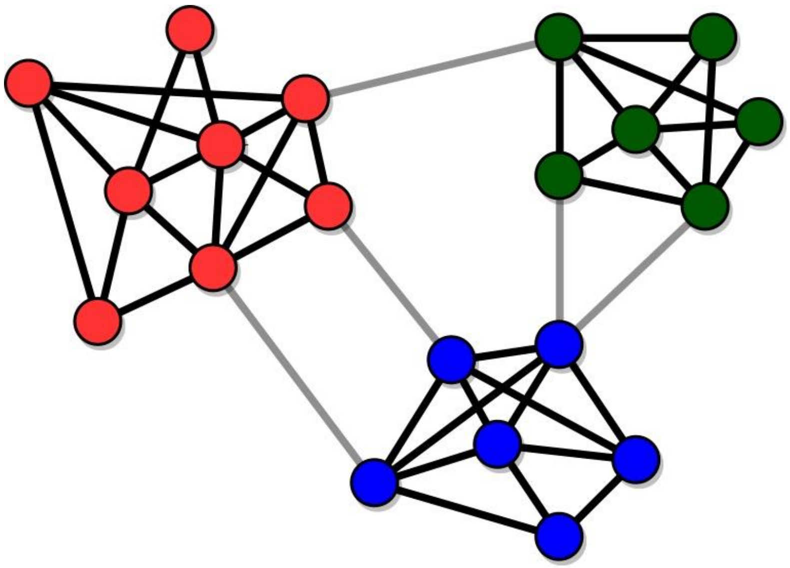


Community detection

- Proposti numerosi metodi euristici per il partizionamento delle reti. Tali metodi possono essere suddivisi in due grandi classi:

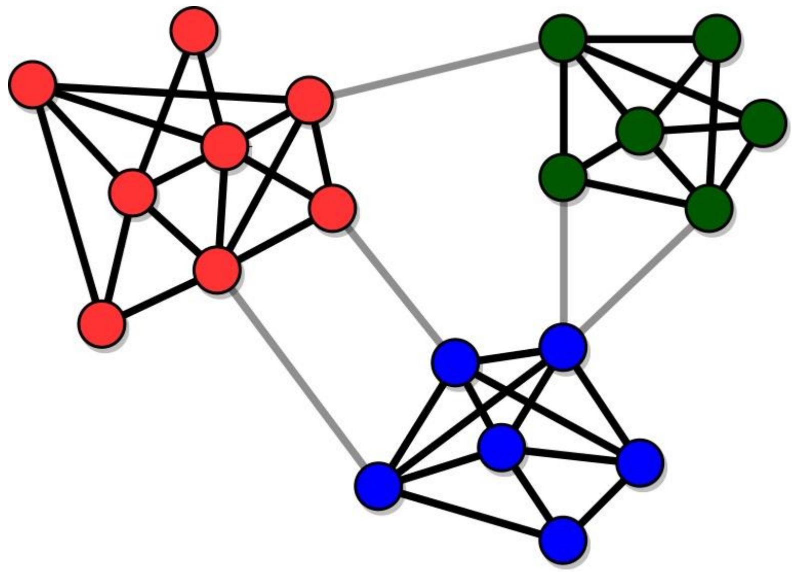
  - Metodi agglomerativi: Partendo dai singoli nodi, si cerca di aggregarli in gruppi (le comunità);

  - Metodi divisivi: partendo dalla rete nel suo complesso, si cerca di dividerla in sottoreti (le comunità) connesse in modo sparso tra loro.



Metodi di divisivi

- **Dove partizionare**?



Nodi diversi ⇔ Ruoli diversi

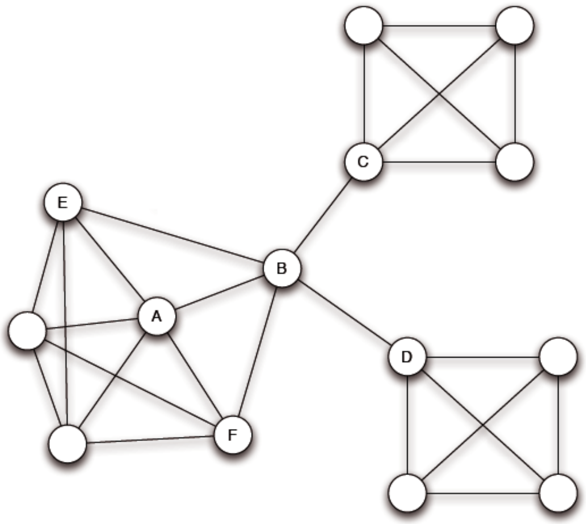
- Embeddedness di un arco AB = #(vicini comuni ad A e B)

  - È il numeratore della funzione di neighborhood overlap
    - Un local bridge ha embeddedness nulla

    - La fiducia tra i due nodi adiacenti cresce con l'embeddedness dell'arco che li unisce
    - Se un nodo è adiacente ad archi con alta embeddedness ha più facilità di relazione e maggiore fiducia nei suoi vicini

Es. A è circondato da archi con alta embeddedness

È incluso in una comunità molto coesa



Centralità di un Nodo

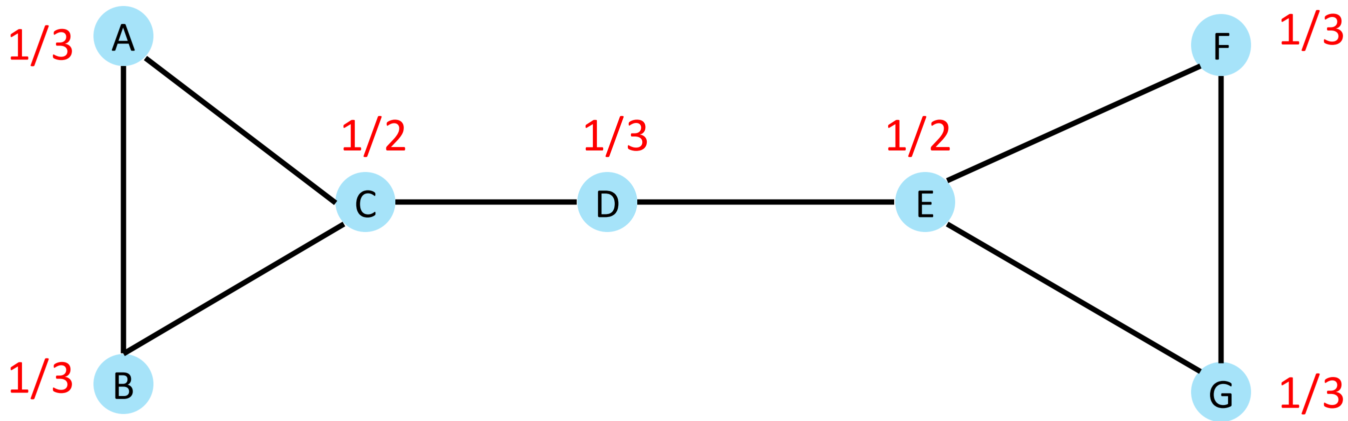
- Misura il ruolo svolto dal nodo nella rete

- Diverse misure di centralità legate a aspetti diversi della rete
  - Degree centrality
  - Closeness centrality
  - Betweenness centrality

- Definite in modo da fornire un valore in [0, 1]
  - Misurano l'"importanza" del nodo

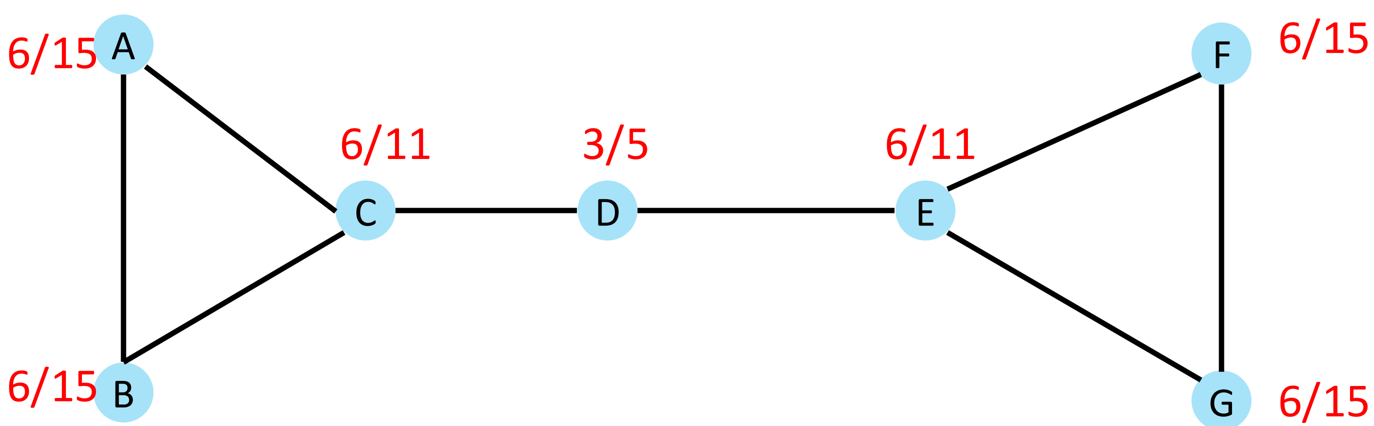
Degree Centrality

- Degree centrality di *z*
  - grado(*z*)/(*n*-1)
  - Misura l'importanza di un nodo in base al numero dei suoi vicini



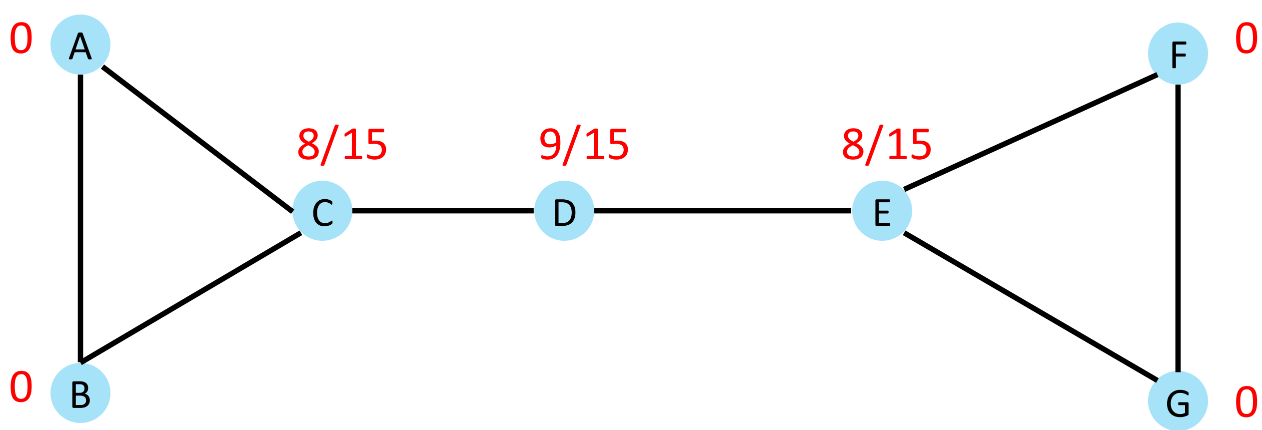
Closeness Centrality

- Closeness centrality di *z*
  - $(n\text{-}1)/\sum_{u \neq z} \mathrm{d}(u,z)$
  - Inverso della distanza media
  - Misura la velocità con cui un nodo viene raggiunto/raggiunge da tutti gli altri



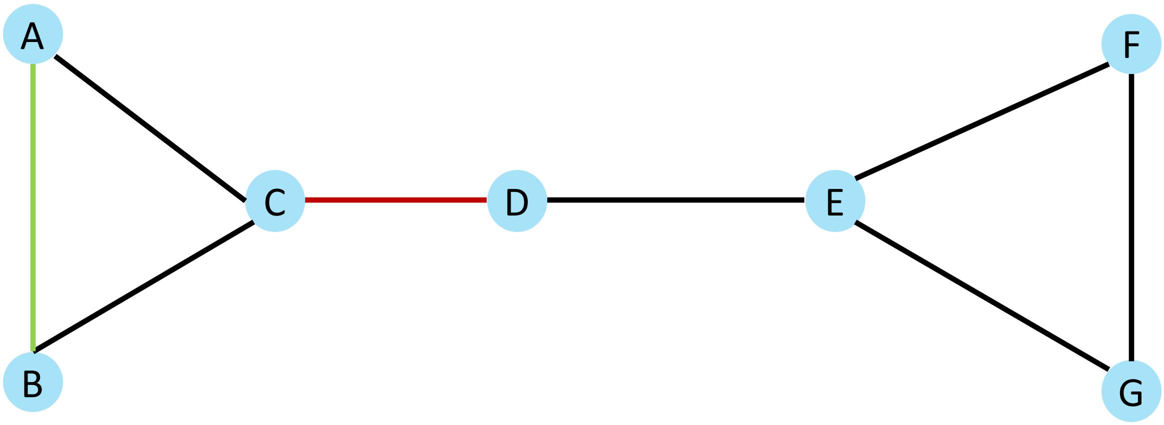
Betweeness Centrality

- Betweenness centrality di z
  - $2 / (n-1)(n-2) \sum_{u \neq v,\, z \neq u,v} P_z(u,v) / P(u,v)$
  - $P(u,v)$ = # cammini minimi tra u e v
  - $P_z(u,v)$ = # cammini minimi tra u e v passanti per z
  - **Misura quanto il nodo z è cruciale per la comunicazione tra tutte le coppie di nodi**



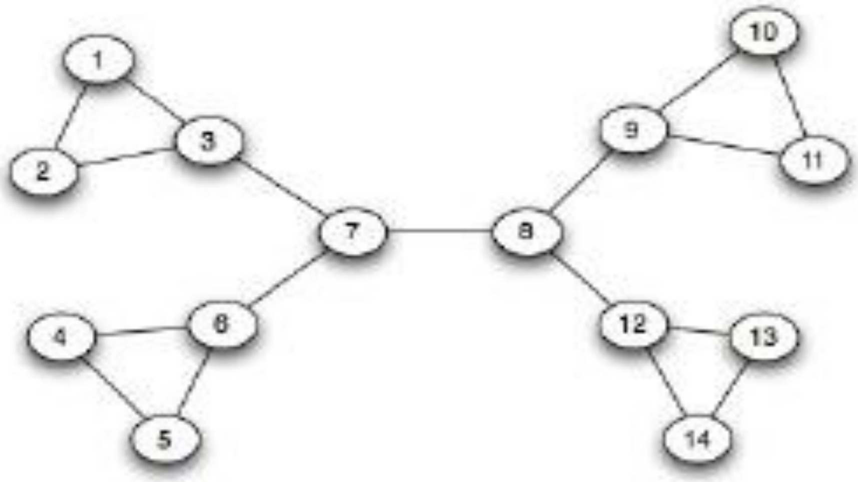
Betweeness Centrality di un edge

- Betweenness centrality di un edge *e* valuta
  - **Misura quanto l'edge *e* è cruciale per la comunicazione tra tutte le coppie di nodi**

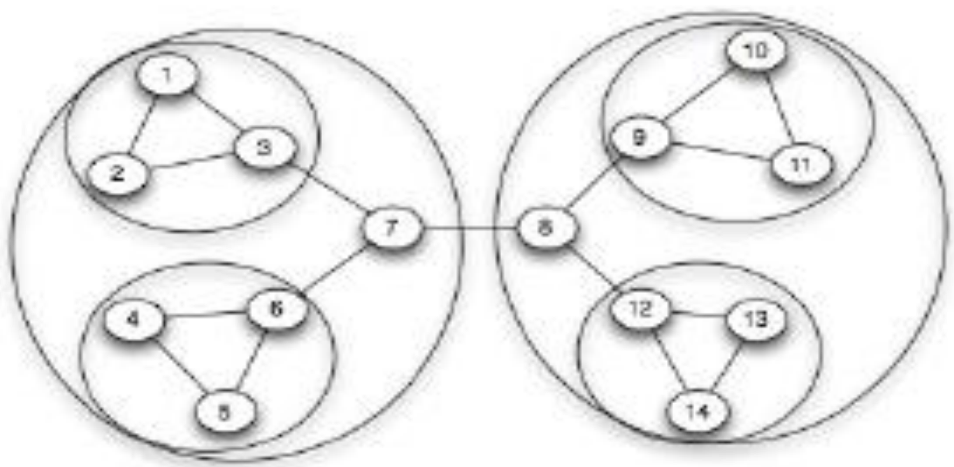


Partizionamento delle reti

- Dove partizionare?



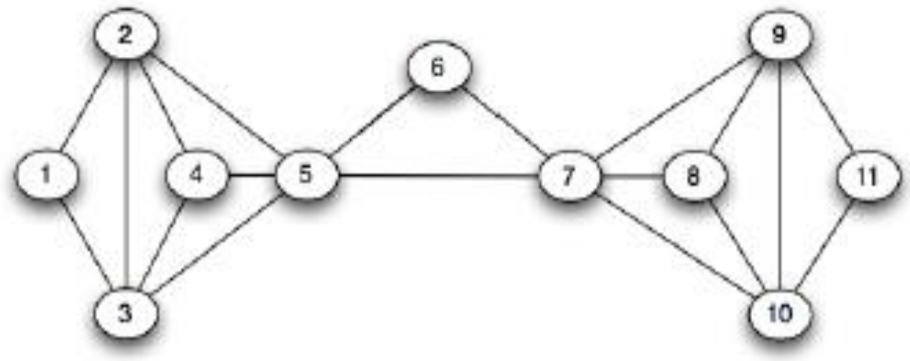
(a) *A sample network*



(b) *Tightly-knit regions and their nested structure*

Partizionamento delle reti

- strutture molto connesse sono chiaramente visibili.

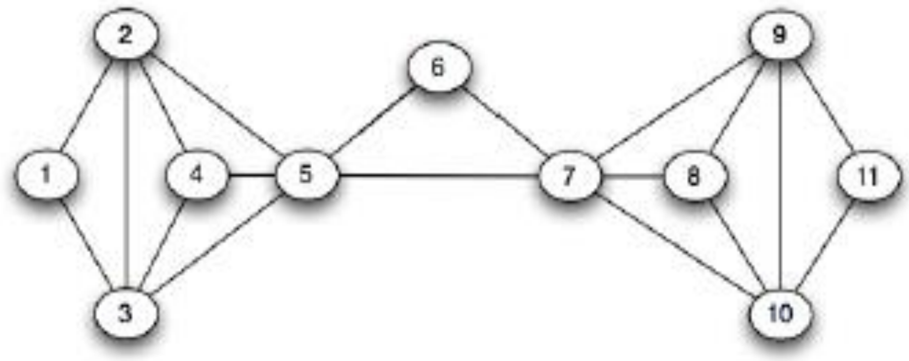


Bridge e local bridges spesso collegano parti della rete debolmente interagenti; si può provare a rimuovere prima questi bridges

- Basta individuare i bridge?
  - No

Partizionamento delle reti

- strutture molto connesse sono chiaramente visibili.



Ponti e ponti locali spesso collegano parti della rete debolmente interagenti; si può provare a rimuovere prima questi ponti

- **Idea per il partizionamento**: Definire il concetto di "traffico" sulla rete e cercare gli archi che portano la maggior parte di questo traffico.
- Ci aspettiamo che questi archi collegano nodi che si trovano in differenti regioni (densamente collegate), e quindi sono buoni candidati per la rimozione.

Traffico

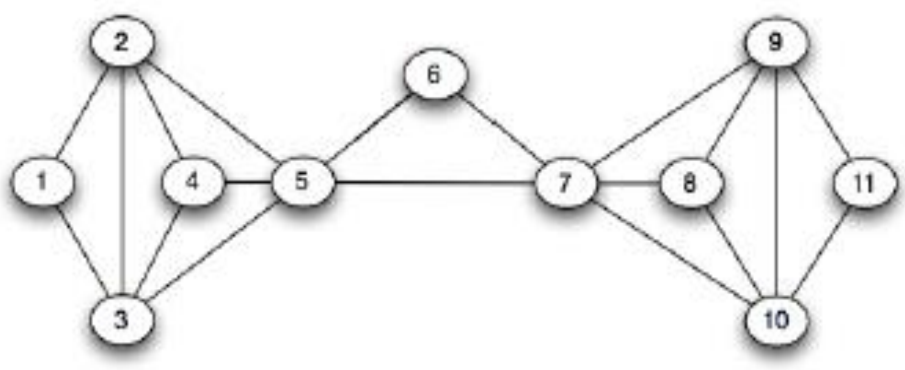
Traffico: Per ogni coppia di nodi A e B nel grafo che sono collegati da un cammino, immaginiamo vi sia un’unità di ‘’flusso" lungo gli archi da A a B.

Se A e B appartengono a diverse componenti connesse, nessun flusso scorre tra di loro.

Il flusso tra A e B si divide in modo uniforme lungo tutto il possibili percorsi più brevi da A a B:

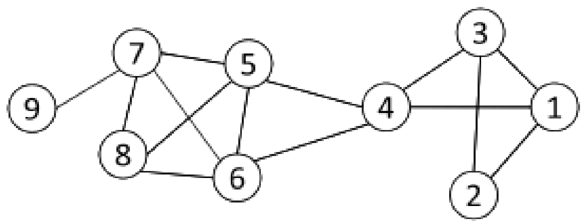
➔

se ci sono k cammini minimi da A e B, 1 / k di unità di flusso passa lungo ciascuno.



Betweeness

La **betweenness** di un arco è la quantità totale di flusso che esso porta, contando il flusso tra cammini minimi di tutte le coppie di nodi che utilizzando questo edge.



La betweenness di (1, 2) è 4 =6(1/2) + 1, infatti

- tutti I cammini minimi tra 2 e 4, 5, 6, 7, 8, 9 passano per (1, 2) o (2,3), e
- (1,2) è il cammino minimo tra 1 e2

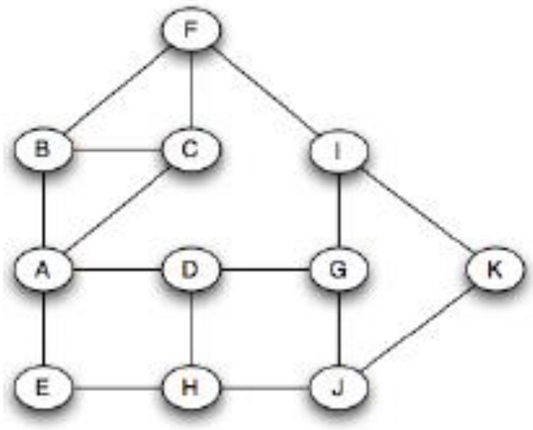
Come calcolare i valori di betweeness?

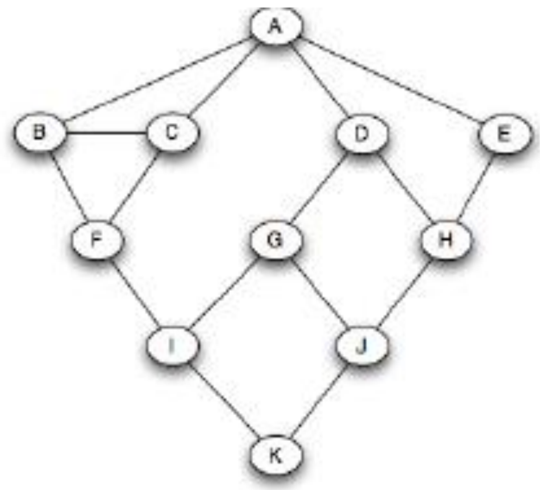
- Si può fare efficientemente?

- Idea
  - Utilizziamo una BFS da ogni nodo *u*
  - Calcoliamo come un'unità di flusso si distribuisce nella rete a partire da un nodo *u*

Calcolare i valori di betweeness

1. Eseguire una ricerca in ampiezza (BFS) del grafo



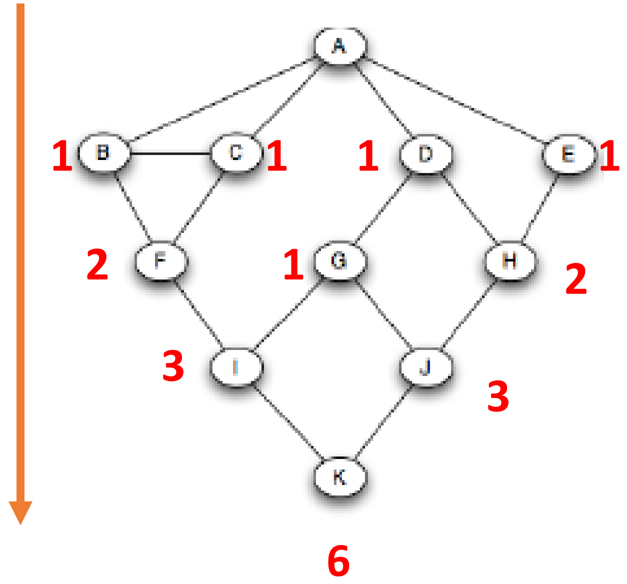


Sono visualizzati i risultati della BFS dal nodo A;
la ricerca viene viene eseguita in ampiezza da ogni nodo a turno.

Algoritmo per il Calcolo della Betweenness

- **A partire da ogni nodo u esegui una BFS e costruisci l'albero dei percorsi minimi**

- **Per ogni nodo *v*, calcola quanti cammini minimi ci sono da *u* a *v***
  - Se il nodo *v* si trova al *k*-imo livello dell'albero i cammini minimi da *u* a *v* hanno lunghezza *k* e passano per i padri di *v* nell'albero
  - Il numero dei cammini minimi per v è la somma del numero di cammini minimi per i suoi padri

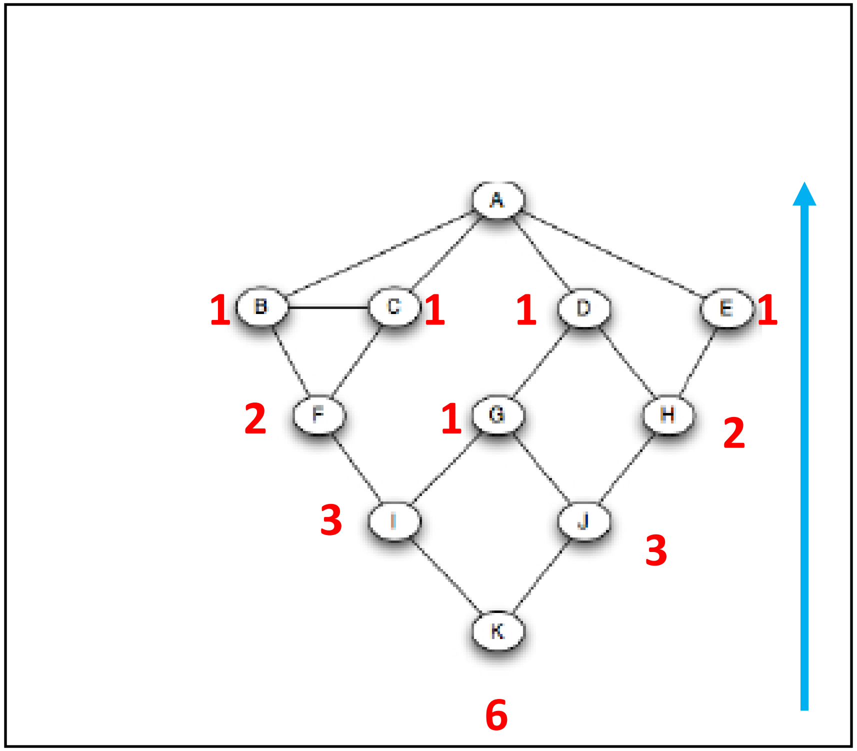


Può essere fatto sommando valori per percorsi più brevi, muovendosi verso il basso attraverso la struttura di ricerca in ampiezza.

Algoritmo per il Calcolo della Betweenness

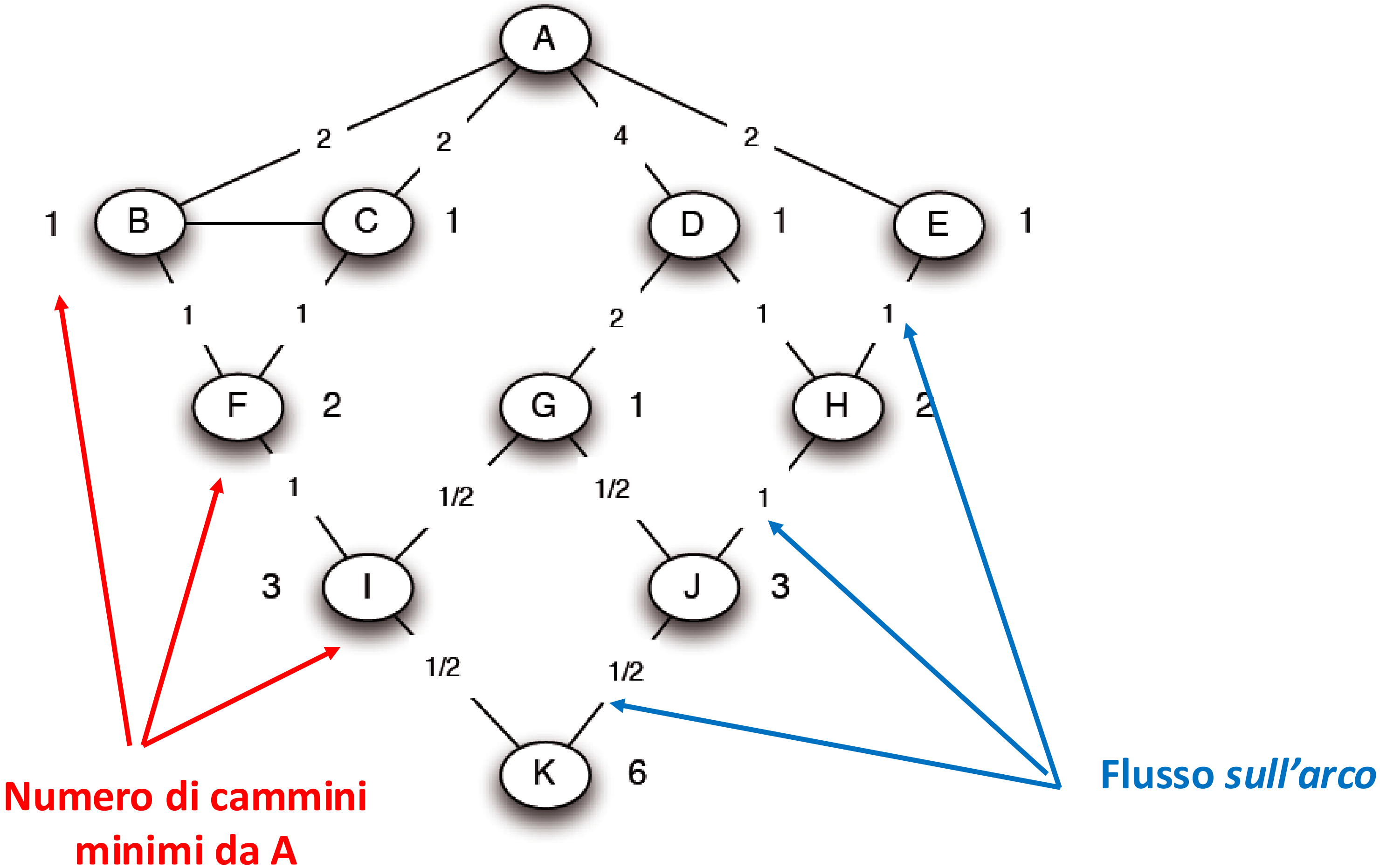
- **A partire da ogni nodo u esegui una BFS e costruisci l'albero dei percorsi minimi**

- **Per ogni nodo *v*, calcola quanti cammini minimi ci sono da *u* a *v***
  - Se il nodo *v* si trova al *k*-imo livello dell'albero i cammini minimi da *u* a *v* hanno lunghezza *k* e passano per i padri di *v* nell'albero
  - Il numero dei cammini minimi per v è la somma del numero di cammini minimi per i suoi padre



- **Determina quanto flusso attraversa ogni arco del grafo**
  - Bottom up
  - Ad ogni nodo assegna un'unità di flusso più tutto quello che gli arriva dai figli
  - Ogni nodo distribuisce equalmente il proprio flusso tra i padri

Determinare la quantità di flusso



Algoritmo per il Calcolo della Betweenness

- Algoritmo
  1. **A partire da ogni nodo u esegui una BFS e costruisci l'albero dei percorsi minimi**
  2. **Per ogni nodo *v*, calcola quanti cammini minimi ci sono da *u* a *v***
     - Se il nodo *v* si trova al *k*-imo livello dell'albero i cammini minimi da *u* a *v* hanno lunghezza *k* e passano per i padri di *v* nell'albero
     - Il numero dei cammini minimi per v è la somma del numero di cammini minimi per i suoi padri
  3. **Determina quanto flusso attraversa ogni arco del grafo**
     - Bottom up
     - Ad ogni nodo assegna un'unità di flusso più tutto quello che gli arriva dai figli
     - Ogni nodo distribuisce il proprio flusso tra i padri (in proporzione al numero di cammini minimi)

Comunità: Algoritmo di Girvan-Newman

1. **Calcolare** la betweenness di ogni arco
   Trovare l'arco di massima betweenness (o gli archi se c'è un pareggio) e rimuovere questi archi dal grafo.
   / * Questo può disconnettere il grafo. Se è così, questo è il primo livello di regioni del partizionamento del grafo. * /

2. **Iterare:** Procedere in questo modo finché rimangono archi nel grafo, in ogni fase ricalcolando tutte le betweenness e rimuovendo l'arco o gli archi di betweness massima
   / * Questo può decomporre alcune componenti esistenti in componenti più piccole; in tal caso, queste sono le regioni annidate all'interno delle regioni più grandi. * /

Algoritmo di Girvan-Newman

1. **Trovare** l'arco di massima betweenness (o gli archi se c'è un pareggio) e rimuovere questi archi dal grafo.
   1. / * Questo può disconnettere il grafo. Se è così, questo è il primo livello di regioni del partizionamento del grafo. * /

2. **Iterare:** ricalcolare le betweenness, e togliere l'arco o archi di massima betweenness.
   1. Procedere in questo modo finché rimangono archi nel grafo,

Complessità: L'algoritmo richiede di ricalcolare ad ogni passo la betweenness di tutti gli m archi

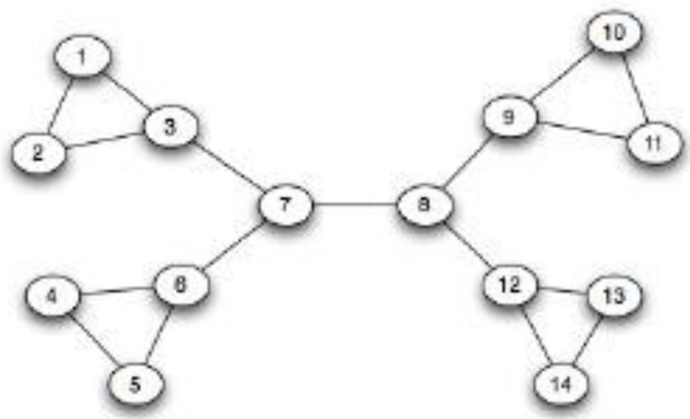
Quindi si hanno O(m) passi.

Ad ogni passo occorre un tempo O(m) per computare la BFS e su di essa la betweeness, per ognuno degli N nodi → In totale $O(m^2N)$.

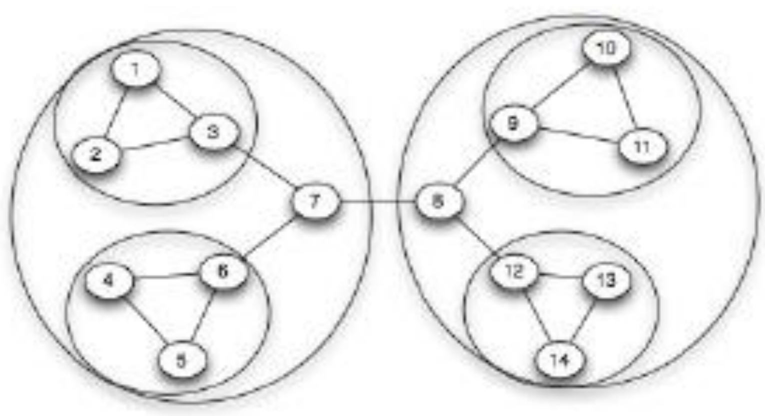
Approcci alternativi: Approssimazione della betweenness

Eliminazione archi : Algoritmo di Girvan-Newman
Esempio

- Dove dividiamo?

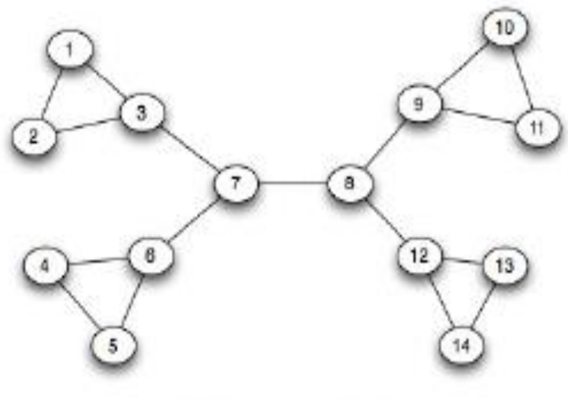


(a) *A sample network*



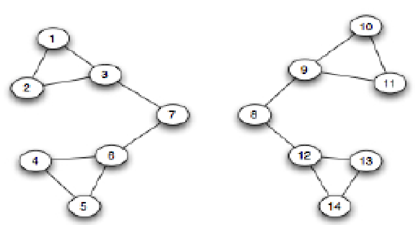
(b) *Tightly-knit regions and their nested structure*

Algoritmo di Girvan-Newman: Esempio



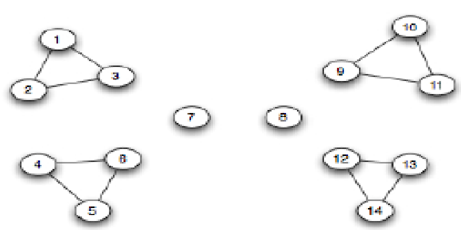
- Betweenness(1-3) = 1X12=12
- Betweenness(7-8)= 7x7 = 49
- Betweenness(3-7)=betweenness(67) =betweenness(8-9) =betweenness(8-2)= 3X11=33

suddivisioni successive



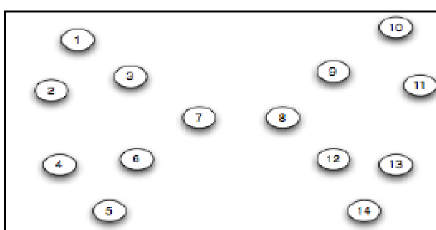
(a) *Step 1*

- Betweenness(1-3) = 1X5=5
- Betweenness(3-7)=betweenness(67) =betweenness(8-9) = betweenness(8-12)= 3X4=12



(b) *Step 2*

- Betweenness di ogni edge = 1



(c) *Step 3*

Algoritmo di Girvan-Newman

1. **Trovare** l'arco di massima betweenness (o gli archi se c'è un pareggio) e rimuovere questi archi dal grafo.
   1. / * Questo può disconnettere il grafo. Se è così, questo è il primo livello di regioni del partizionamento del grafo. * /

2. **Iterare:** ricalcolare le betweenness, e togliere l'arco o archi di massima betweenness**.**
   1. Procedere in questo modo finché rimangono archi nel grafo,

Complessità: L'algoritmo richiede di ricalcolare ad ogni passo la betweenness di tutti gli m archi

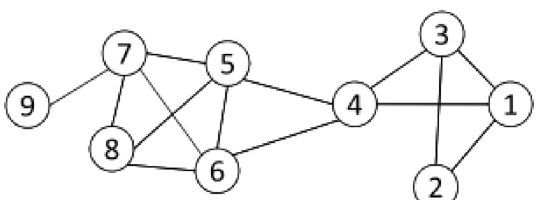
Quindi si hanno O(m) passi.

Ad ogni passo occorre un tempo O(m) per computare la BFS e su di essa la betweeness, per ognuno degli N nodi → In totale $O(m^2N)$.

Eliminazione archi : Algoritmo di Girvan-Newman

1. **Trovare** l'arco di massima betweenness (o gli archi se c'è un pareggio) e rimuovere questi archi dal grafo.
   / * Questa operazione può disconnettere il grafo. Se è così, questo è il primo livello di regioni del partizionamento del grafo. * /

2. **Iterare:** Procedere in questo modo finché rimangono archi nel grafo, in ogni fase ricalcolando tutte le betweenness e rimuovendo l'arco o gli archi di betweness massima



	1	2	3	4	5	6	7	8	9
1	0	4	1	9	0	0	0	0	0
2	4	0	4	0	0	0	0	0	0
3	1	4	0	9	0	0	0	0	0
4	9	0	9	0	10	10	0	0	0
5	0	0	0	10	0	1	6	3	0
6	0	0	0	10	1	0	6	3	0
7	0	0	0	0	6	6	0	2	8
8	0	0	0	0	3	3	2	0	0
9	0	0	0	0	0	0	8	0	0

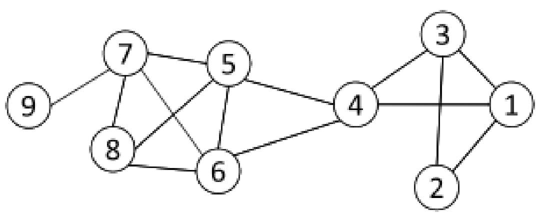
Rimuoviamo (4,5) e (4, 6),

Valori iniziali di betweenness

Eliminazione archi : Algoritmo di Girvan-Newman

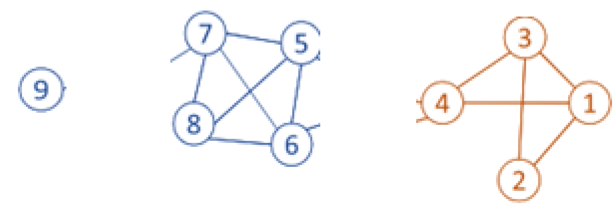
1. **Trovare** l'arco di massima betweenness (o gli archi se c'è un pareggio) e rimuovere questi archi dal grafo.
   / * Questa operazione può disconnettere il grafo. Se è così, questo è il primo livello di regioni del partizionamento del grafo. * /

2. **Iterare:** Procedere in questo modo finché rimangono archi nel grafo, in ogni fase ricalcolando tutte le betweenness e rimuovendo l'arco o gli archi di betweness massima



Ricalcolando I valori, risulta che l'edge (7,9) ha il valore max 4, e deve essere levato.

tre componenti:

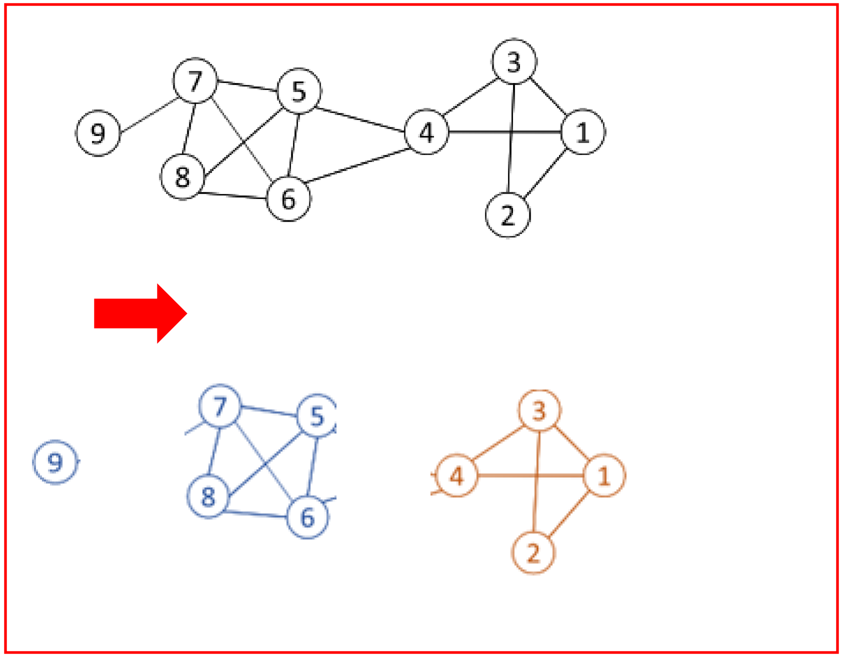


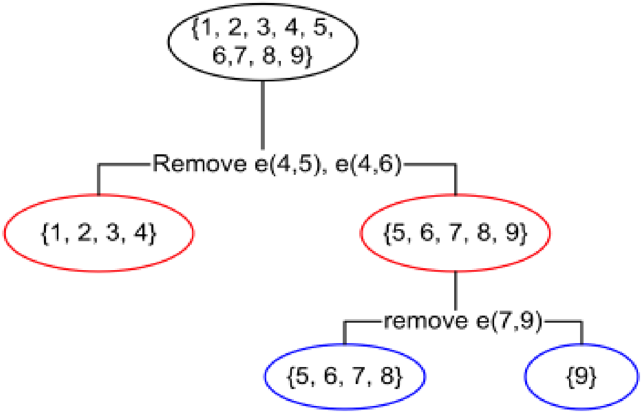
Eliminazione archi : Algoritmo di Girvan-Newman

1. **Trovare** l'arco di massima betweenness (o gli archi se c'è un pareggio) e rimuovere questi archi dal grafo.

   / * Questa operazione può disconnettere il grafo. Se è così, questo è il primo livello di regioni del partizionamento del grafo. * /

2. **Iterare:** Procedere in questo modo finché rimangono archi nel grafo, in ogni fase ricalcolando tutte le betweenness e rimuovendo l'arco o gli archi di betweness massima





Algoritmo di Girvan-Newman

1. **Trovare** l'arco di massima betweenness (o gli archi se c'è un pareggio) e rimuovere questi archi dal grafo.
    1. / * Questo può disconnettere il grafo. Se è così, questo è il primo livello di regioni del partizionamento del grafo. * /

2. **Iterare:** ricalcolare le betweenness, e togliere l'arco o archi di massima betweenness**.**
    1. Procedere in questo modo finché rimangono archi nel grafo,

    Complessità: L'algoritmo richiede di ricalcolare ad ogni passo la betweenness di tutti gli m archi

    Quindi si hanno O(m) passi.

    Ad ogni passo occorre un tempo O(m) per computare la BFS e su di essa la betweeness, per ognuno degli N nodi ➔ In totale $O(m^2N)$.

3. **Determina la partizione *"migliore"* tra tutte quelle trovate alle varie iterazioni**

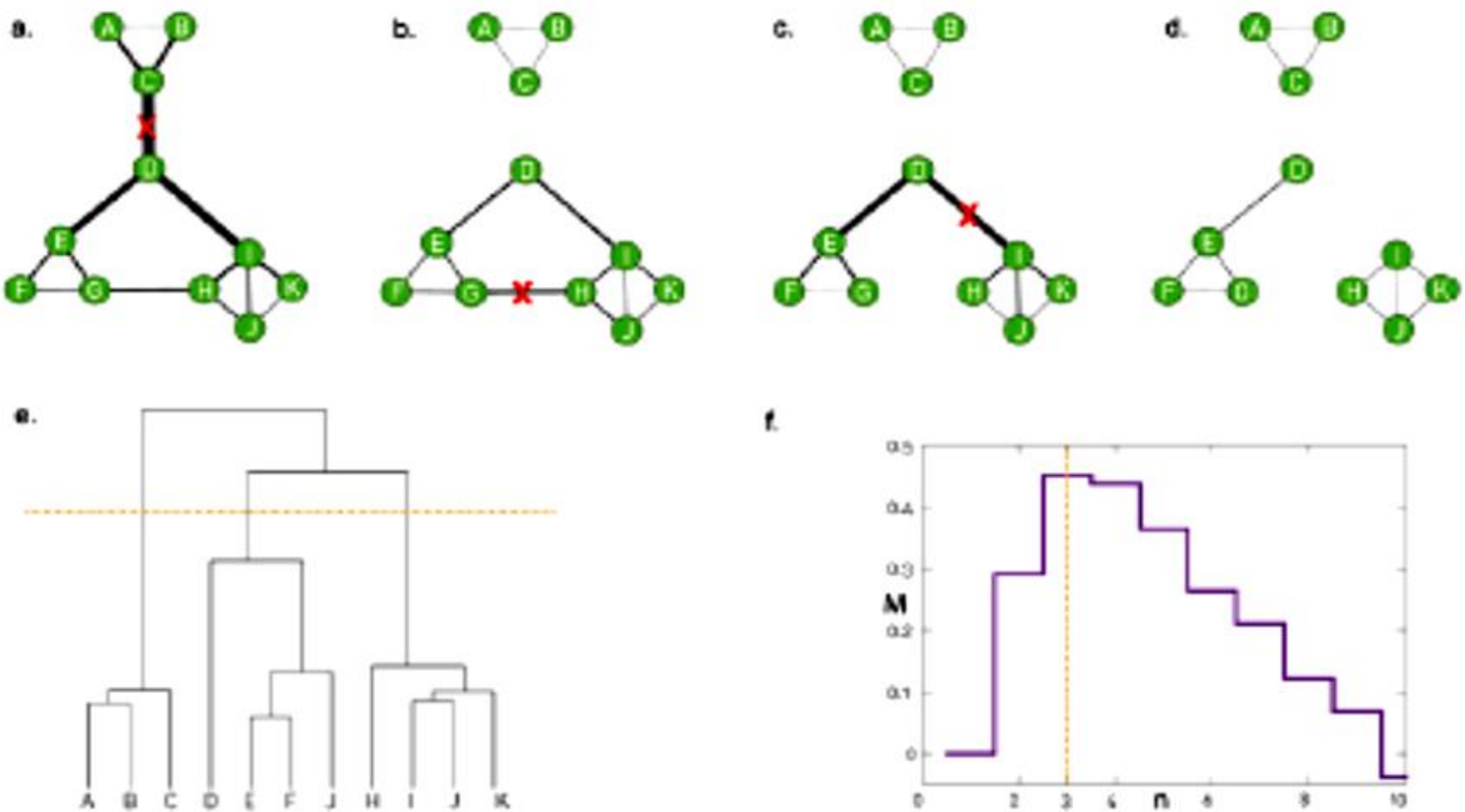
Quale partizione scegliere?

Poiché **l**'algoritmo lavora per partizionamenti successivi, in generale è necessario un criterio per stabilire quale livello di partizione è quello giusto.

Una possibilità è quella di valutare i vari livelli di partizionamento utilizzando una *misura di bontà delle partizioni*.

Quale partizione scegliere?

Es. Applicando l'algoritmo di Girvan-Newman, si ottengono le seguenti partizioni successive. La modularità M (che vedremo in seguito) ci dice che la partizione migliore è al livello 3 (l'ultima riportata in figura).



Modularità

quantità scalare che misura la densità di archi all'interno delle comunità individuate

Data una partizione dei vertici $\mathcal{C} = \{C_1, \ldots, C_t\}$ in cluster/comunità

la modularità Q ($\mathcal{C}$) di $\mathcal{C}$ è definita come

$$Q(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left[\frac{|E(C)|}{m} - \left(\frac{\sum_{v \in C} \deg(v)}{2m} \right)^2 \right]$$

Nota: C varia su tutti i cluster ➔ tutti edges sono contati due volte in totale

dove E (C) indica l'insieme di edges tra vertici del cluster C

Modularità

$$Q(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left[\frac{|E(C)|}{m} - \left(\frac{\sum_{v \in C} \deg(v)}{2m} \right)^2 \right]$$

IDEA

- per massimizzare il primo termine, dovrebbero essere contenuti molti edge nei cluster

- la minimizzazione del secondo termine si ottiene dividendo il grafo
  in molti cluster di grado totale ''piccolo''.

Modularità

$$Q(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left[\frac{|E(C)|}{m} - \left(\frac{\sum_{v \in C} \deg(v)}{2m} \right)^2 \right]$$

Modularità prende valori tra -1/2 e 1

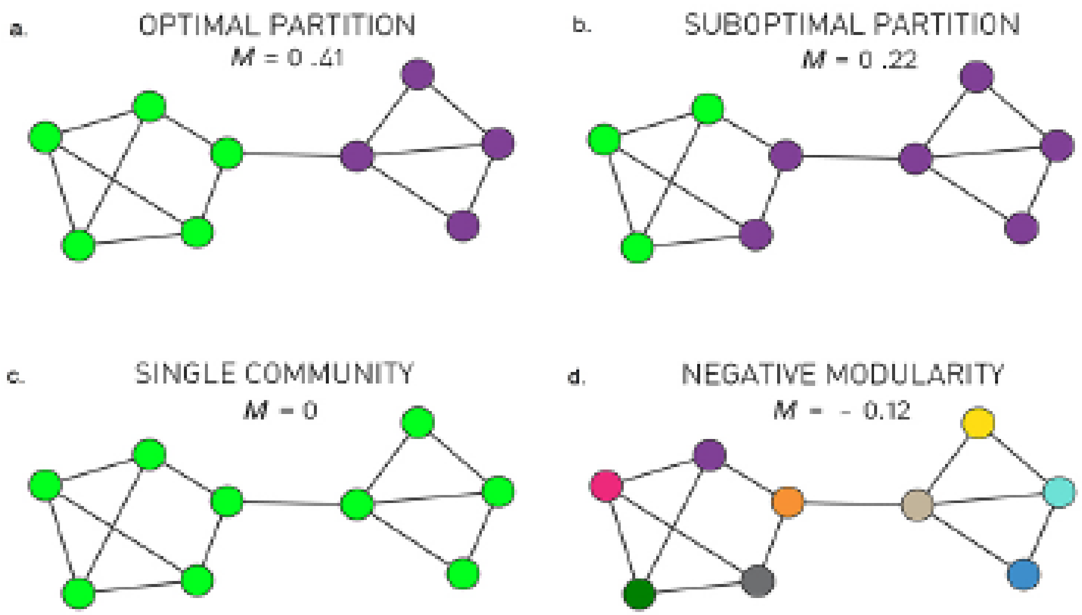
- Q(C) = - 1/2 quando tutti gli edge sono tra cluster
  - grafo bipartito con V=X U Y con 2 cluster corrispondenti a X e Y

- Quando non vi sono edges tra cluster
  -es. tutti cluster di 2 nodi connessi da 1 edge → Q(C)=1-1/m)

  Q(C) → 1 al crescere del numero dei nodi

Modularity

$$Q(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left[\frac{|E(C)|}{m} - \left(\frac{\sum_{v \in C} \deg(v)}{2m} \right)^2 \right]$$



Variazione della modularità al variare della partizione, in un grafo che ammette una chiara partizione in due comunità.

Modularità

$$Q(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left[\frac{|E(C)|}{m} - \left(\frac{\sum_{v \in C} \deg(v)}{2m} \right)^2 \right]$$

L'esperienza mostra che

- valori >0.5 indicano la presenza effettiva di una struttura in comunità,
- valori prossimi allo zero indicano che la distribuzione degli archi tra intra- e inter-comunità non si discosta dalla casualità (in grafi ER il valore atteso è 0)