Grafos y Redes Sociales

Organización de Datos 75.06
Abril 2014

El presente apunte es una recopilación de información, observaciones y algoritmos para el manejo de grafos masivos. Varios de los temas presentados en este apunte están aun en proceso de investigación por lo que muchas secciones son simplemente una recopilación de datos interesantes que luego pueden usarse para el desarrollo de técnicas y algoritmos.

PARTE I: GRAFOS EN GENERAL

Redes y Grafos en el mundo actual

El simple concepto de un grafo formado por nodos y aristas puede llevarse al contexto de muchas áreas de apliación el a actualidad. Algunos ejemplos son:

La Web: Donde cada página es un nodo y cada link una arista
Facebook: Donde cada usuario es un nodo y cada relación "friend" es una arista
Emails: Donde cada usuario es un nodo y cada email es una arista
Google+: Donde cada usuario es un nodo y cada relacion una arista
Papers: Donde cada paper es un nodo y cada citación es una arista
etc.

Como podemos ver algunos de estos grafos son dirigidos y otros no. Un grafo es dirigido cuando sus aristas tienen "dirección". La web es un grafo dirigido pues cada link tiene una página origen y una página destino. Facebook es un grafo no-dirigido pues la relación "friends" no tiene dirección.

Entender la estructura que tiene un grafo ayuda a desarrollar algoritmos que nos den información útil sobre el mismo. Vamos a comenzar con un análisis muy global de la Web como grafo.

Estructura de Internet

La pregunta es muy simple: ¿Que aspecto tiene la Web si la representamos con un grafo?
Para encontrar una solución hay que procesar miles de millones de páginas y sus links y ver que resultado obtenemos. Antes de presentar los resultados de este estudio vamos a repasar algunas definiciones sobre grafos:

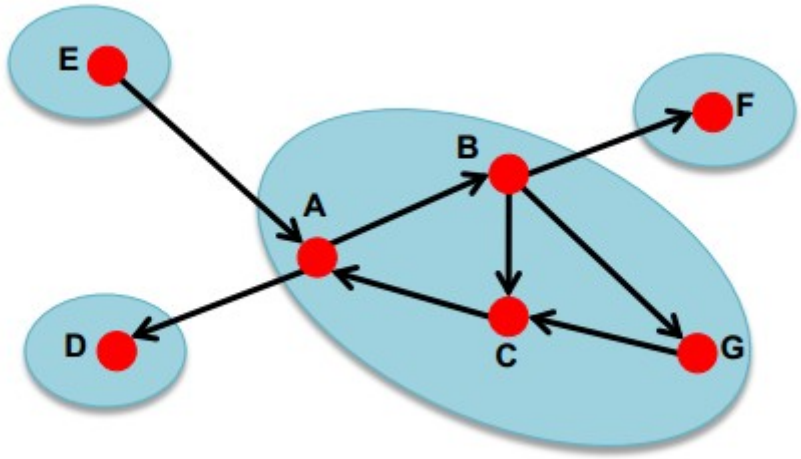
Grafo fuertemente conexo: Un grafo es fuertemente conexo si para cualquier par de vertices a,b existe un camino desde a hacia b.

Componente fuertemente conexo (CFC): Un componente fuertemente conexo es un subgrafo fuertemente conexo. Es decir un conjunto de vertices S pertenecientes a un grafo G tal que desde cualquier nodo dentro de S se puede llegar a cualquier otro nodo de S y ademas no hay otros nodos en G que puedan pertenecer a S.

Grafo dirigido aciclico (GDA): Es un grafo dirigido en el cual no hay ciclos, si u puede llevar a v entonces v no puede llevar a u.

Cualquier grafo es una combinación de estos dos tipos. Es decir que un grafo dirigido cualquiera puede representarse como un grafo dirigido aciclico entre componentes fuertemente conexos.

Ejemplo:



En este grafo hay 4 componentes fuertemente conexos:
{A,B,C,G} {E} {D} y {F}
El grafo puede expresarse como un grafo dirigido aciclico entre estos 4 componentes (como si cada componente fuera un nodo del GDA)

Lema: Los componentes fuertemente conexos particionan los nodos del grafo
(Explicación: Cada nodo solo puede pertenecer a un componente fuertemente conexo)
Demostración: Si un nodo x pertenece a S y S' entonces desde x se puede llegar a cualquier elemento de S y S' y desde cualquier elemento de S y S' se puede llegar a x
entonces S U S' sería fuertemente conexo y eso es una contradicción.

Lema: El grafo formado por los componentes fuertemente conexos es acíclico.
Demostración: Si existe un ciclo entre los CFCs de un grafo entonces cualquier elemento dentro del ciclo es alcanzable desde cualquier otro.
Pero entonces el grafo no es un conjunto de CFCs porque los CFCs son máximos.

En conclusión todo grafo es un GDA de sus componentes fuertemente conexos.

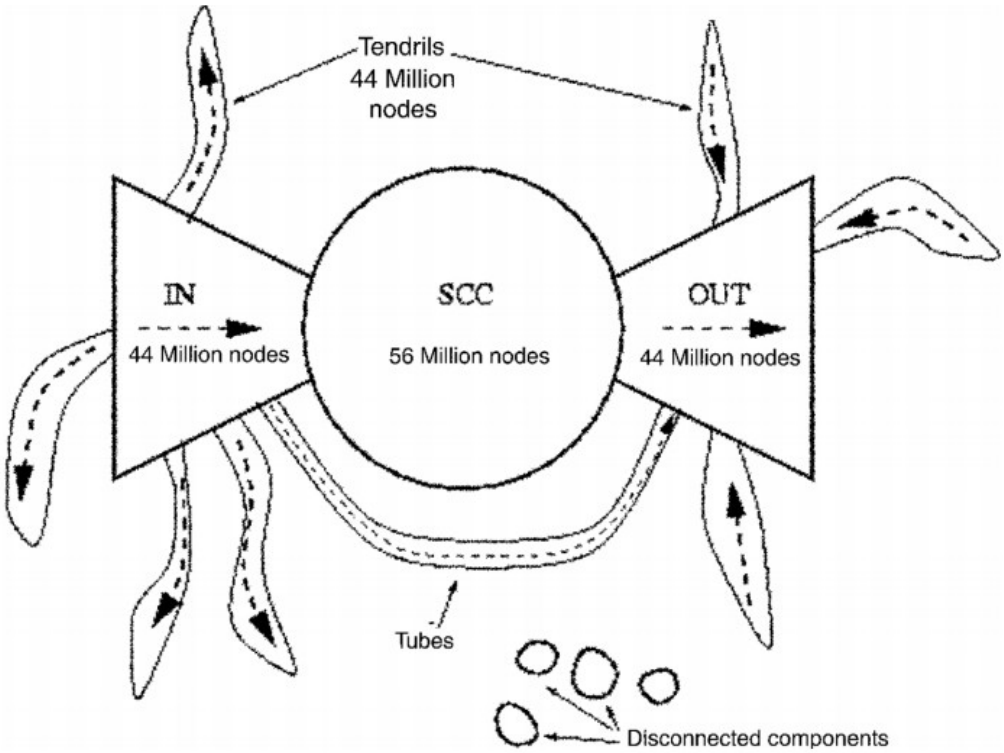
Componentes Fuertemente Conexos y la estructura de la Web

Debemos analizar ahora, que tipo de estructura tendrá la web representada por páginas y sus links. Esta claro que la forma en que las páginas linkean a otras y vice-versa tiende a formar componentes fuertemente conexos. Se puede intuir que en la Web debería existir un componente fuertemente conexo gigante que reuna a todas las páginas tales que desde cualquiera de ellas se puede llegar a cualquier otra.

La intuición para esto es que si existieran dos componentes fuertemente conexos gigantes entonces bastaria con agregar links desde dos páginas cualesquiera de estos componentes para unirlos.

En 2000 un estudio publicado por Broder procesó millones de páginas web y sus links y de ese

estudio se puede concluir que la Web tiene forma de... moño!



El CFC (SCC en inglés) reune aproximadamente al 28% de las páginas de la web. El conjunto "IN" son páginas que linkean a alguna página del CFC gigante pero no hay link del CFC a estas.
El conjunto "OUT" son páginas linkeadas desde el CFC gigante pero no hay link desde las páginas al mismo.

Además de estos tres grupos existen otros conjunto menores como pequeños componentes fuertemente conexos menores separados. Tentáculos de links desde los conjuntos "IN" y "out" y algunos "túneles" entre páginas de "IN" y "OUT".

El camino mínimo entre dos páginas cualesquiera dentro del CFC gigante es de en promedio 16 links según Broder.

Propiedades Básicas de Nodos y Grafos Aleatorios

Grado de un vértice y Grado promedio:

El grado de un vértice es la cantidad de aristas que inciden sobre el mismo.
En un grafo dirigido podemos distinguir el grado de entrada (aristas que llegan) dell grado de salida.

Grado promedio de un vértice (K')

En un grafo no-dirigido es K' = 2E / N (E = cantidad de aristas, N = nodos)
En un grafo dirigido es K' = E/N

Grafo Completo

Un grafo completo es un grafo que tiene todas las aristas posibles.
Emax = (N (N-1)) / 2

En un grafo promedio K' es N-1

Grafo Disperso

En un grafo disperso la cantidad de aristas es mucho menor al máximo es decir:

E <<<<< Emax
o lo que es lo mismo
K' <<<<< N-1

Casi todas las redes sociales y grafos que estudiamos son dispersos.

Ejemplos:

Una fracción de la web: N=319.000 K'=9.65
Linkedln: N=6.946.668 K'=8.87

Caracterizacion de un Grafo

Vamos a estudiar ahora algunas propiedades que sirven para caracterizar un grafo, es decir entender sus propiedades. Esto sirve por ejemplo para saber que un algoritmo que funciona bien en un cierto tipo de grafo funcionará bien en grafos de características similares.

Vamos a caracterizar un grafo por tres propiedades:

- Distribución del grado
- Diámetro del grafo
- Longitud promedio de un camino
- Coeficiente de clustering

Distribución del Grado P(k)

P(k) mide la probabilidad de que un nodo tomado al azar tenga grado k.
Si Nk es el número de nodos con grado k entonces P(k) = Nk / N

Diámetro del Grafo

El diámetro de un grafo se define como el camino mínimo de longitud máxima entre dos nodos cualesquiera del grafo.

Los caminos mínimos a partir de un nodo se pueden calcular, por ejemplo, con el algoritmo de Dijsktra.
Todos los caminos mínimos en un grafo se pueden calcular con el algoritmo de Floyd-Warshall.

Estos algoritmos no son tema de este apunte.

Longitud Promedio de un Camino h'

Es la longitud promedio del camino entre 2 nodos cualesquiera del grafo.

h' = 1/2Emax Sum(hij)

Donde hij es la distancia desde i hasta j (ignorar distancias infinitas es decir entre nodos no conectados)

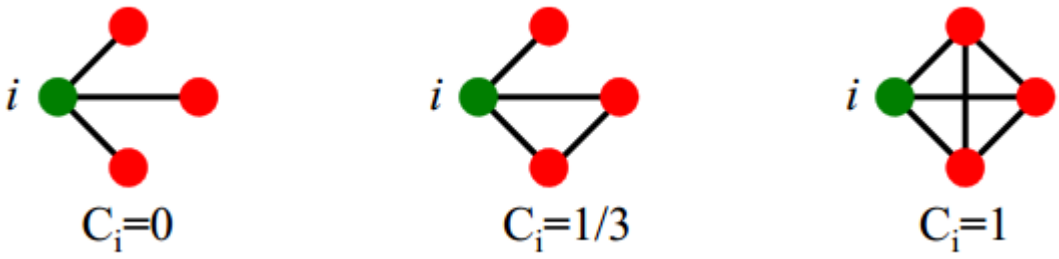
Coeficiente de Clustering

El coeficiente de clustering de un nodo mide que porcentaje de los vecinos de un cierto nodo están conectados entre sí.

Ci es un número entre 0 y 1.

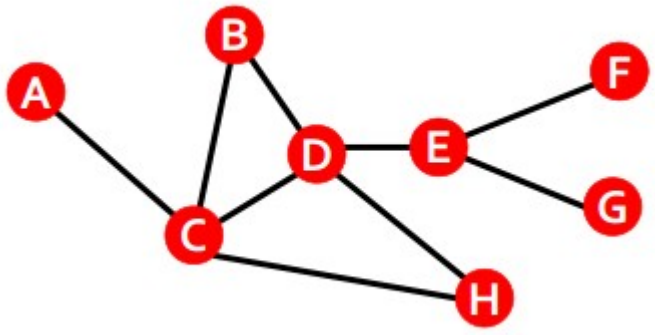
Ci = (2 ei) / (ki (ki -1))

ei = número de aristas entre los vecinos del nodo i
ki = grado del nodo i



El coeficiente de clustering de un grafo es el promedio entre el coeficiente de clustering de sus nodos: Sum(Ci) / N

Ejemplo:



¿Cuál es el coeficiente de clustering de este grafo?

Ejemplo: Para "D"
Cd = 2 ed / kd (kd -1)

kd = 4
ed = 2 (C-H y C-B)

Cd = 4 / 12 = 1/3

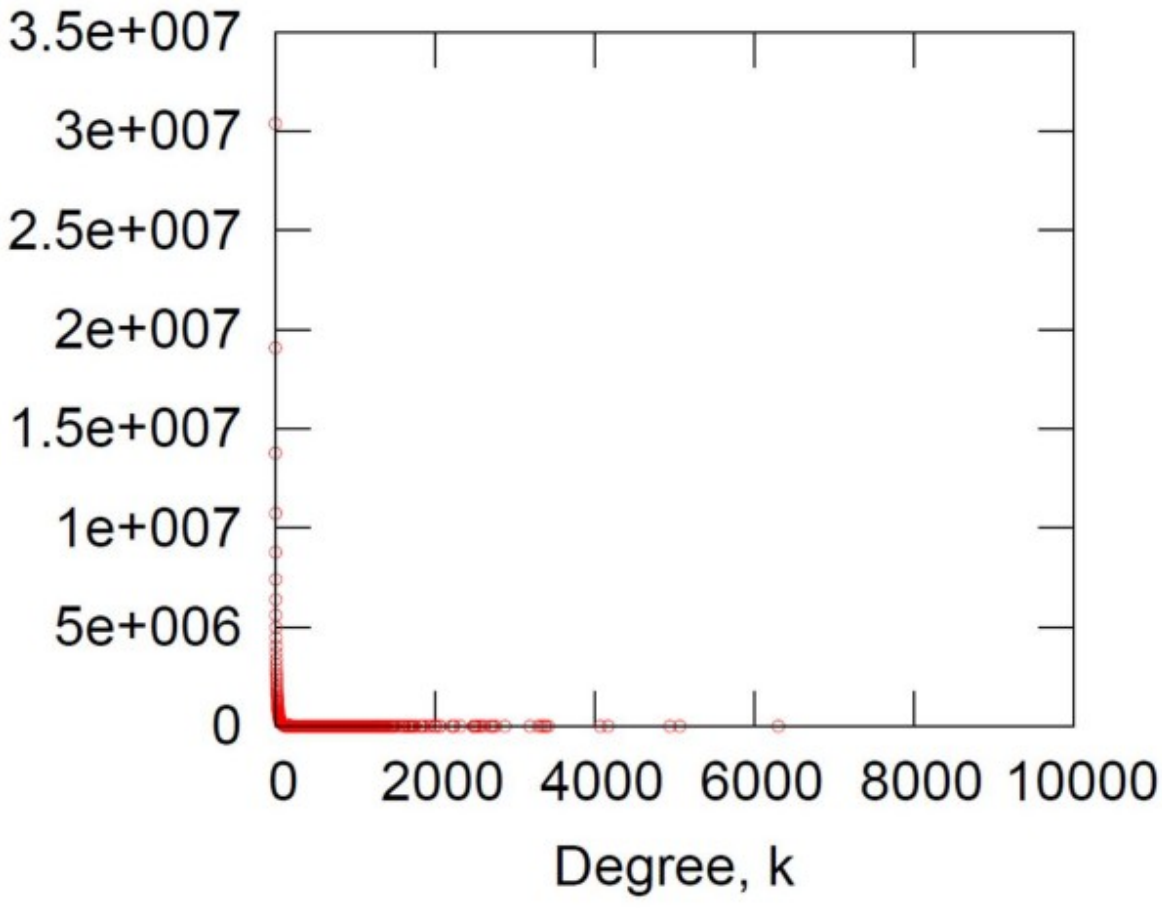
Ejercicio: Calcular el coeficiente para los demás nodos y hacer el promedio.

Caso de Estudio

Como ejemplo vamos a mostrar las propiedades que encontramos para el grafo de una porción de mensajes via MSN, en donde los nodos son los usuarios y las aristas son los mensajes enviados entre los mismos.

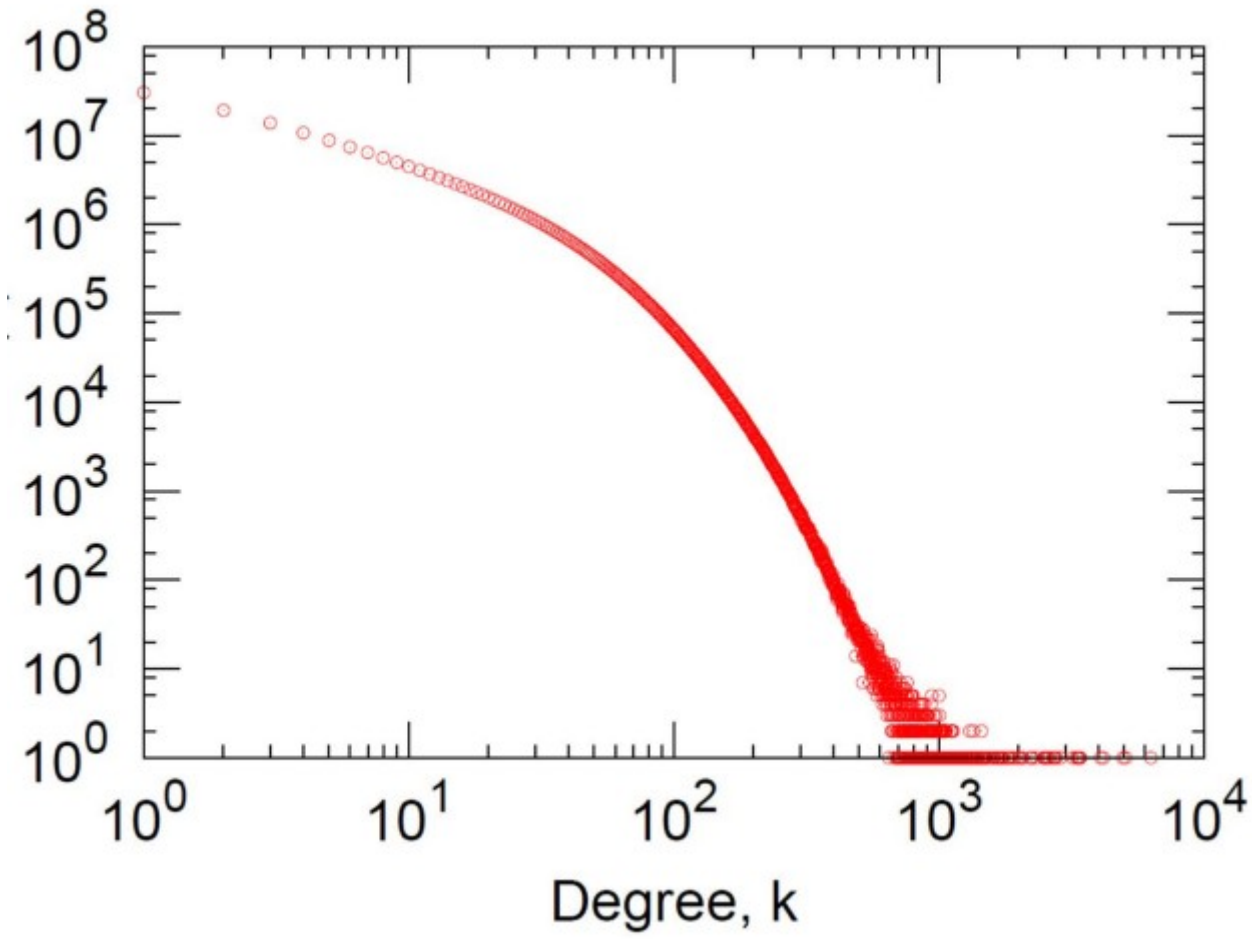
Grado Promedio

El siguiente grafo muestra el grado del grafo (Eje x) y la cantidad de valores (Eje y)



Como podemos ver es una distribución terriblemente desbalanceada. Hay muchos nodos con grado

bajo y muy pocos nodos con grado muy alto. Esta distribución sigue una "Ley de Potencias" de la forma y = c * x ^ -a es decir que si la graficamos usando escala logarítmica deberíamos tener una distribución aproximadamente lineal.



Las leyes de potencias aparecen en inumerable cantidad de casos, tanto en redes como en la naturaleza, sociología, biología etc. Comunmente se las asocia con el principio 80-20 en nuestro caso el 20% de los usuarios acumulan el 80% de los mensajes enviados por MSN.

Coeficiente de Clustering

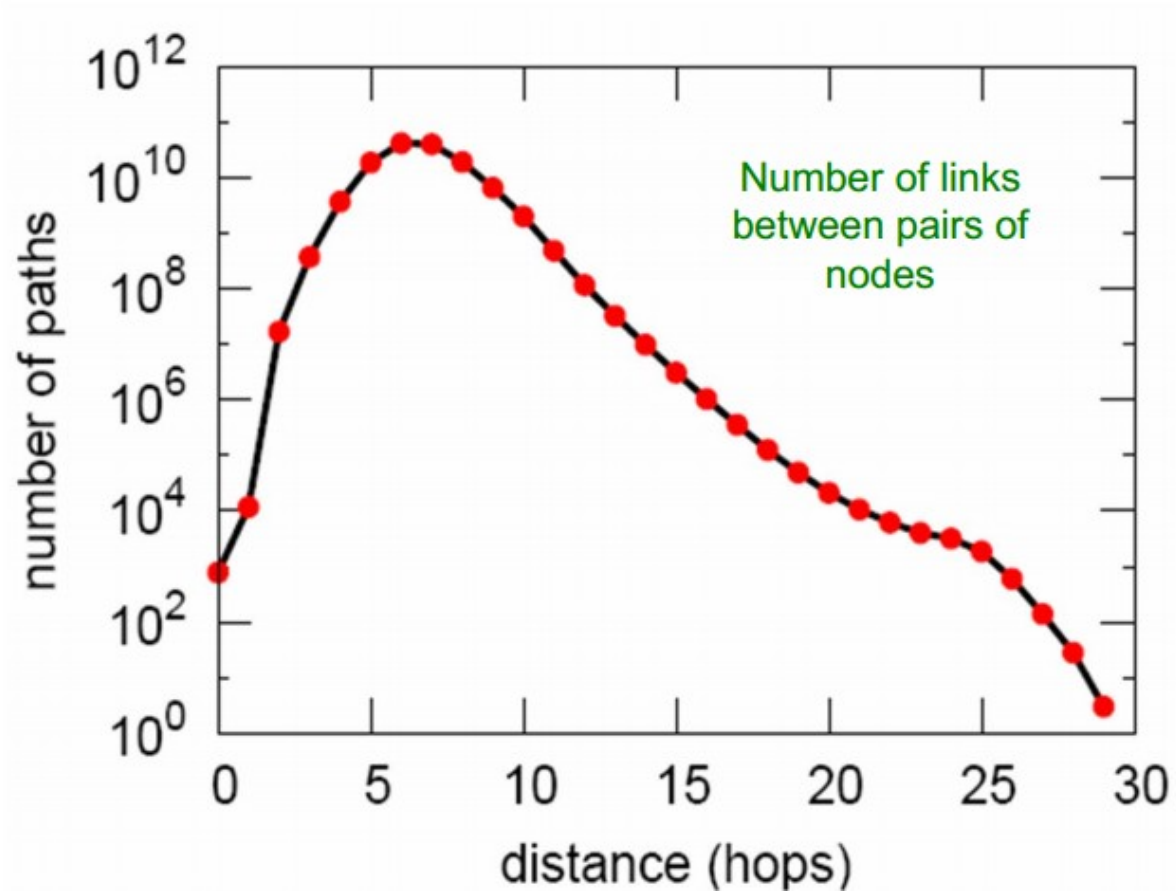
Para el fragmento de red MSN analizado se encontró que el coeficiente de clustering es

C = 0.1140

Longitud promedio de un camino

Para MSN el resultado dió que un camino promedio tiene longitud 6.6

El siguiente gráfico muestra la cantidad de nodos que hay para cada grado



El 90% de los nodos puede alcanzarse con menos de 8 aristas. Esto es conocido como "Principio del Mundo Chico" cuyo ejemplo mas popular son los famosos grados de separación entre Kevin Bacon y cualquier otro actor.

En definitiva para la red MSN tenemos las siguientes características:

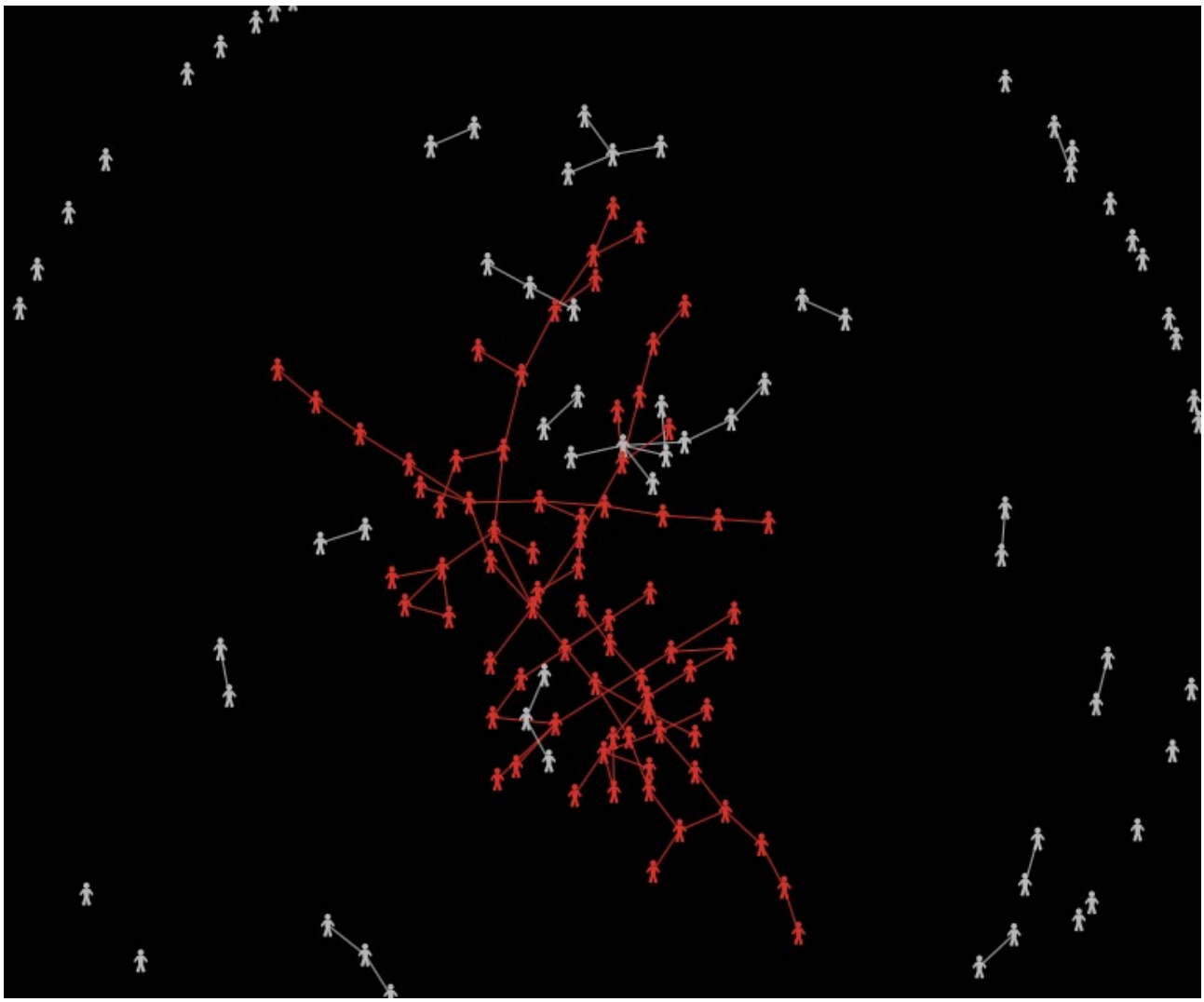
Distribución del grado: Ley de potencias (grado promedio 14.4)
Camino promedio: 6.6
Coeficiente de clustering: 0.1140

Estos resultados por sí mismos no tienen ningún valor, es necesario tener algún patrón o standard contra el cual poder comparar para saber que significado tiene la caracterización de un cierto grafo. Es por esto que resulta útil el estudio de los grafos aleatorios.

Grafo Aleatorio Gn,p o Grafo de Erdös-Renyi

Gn,p es un grafo aleatorio con n nodos y donde cada arista posible aparece con probabilidad "p"

Ejemplo:



Notar la aparición del componente conexo gigante en este caso en color rojo.

Nos interesa calcular los valores de la distribución del grado, camino promedio y coeficiente de clustering para un grafo aleatorio Gn,p

Se demuestra que:

P(k) = Combinatorio[n-1,k] * p^k * (1-p) ^ (n-1-k)

Esto tiene una distribución binomial.

El Coeficiente de clustering es: C = p = k' / n

Y el camino promedio es O(log n)

(La demostración de estas propiedades para el grafo aleatorio Gn,p escapa a este apunte, los interesados consultar paper de Erdös-Renyi)

Si comparamos las características de un grafo aleatorio con los grafos que caracterizan las redes sociales vamos a encontrar lo siguiente:

- El componente conexo gigante esta presente en ambos casos

- El camino de longitud promedio es muy similar
- El coeficiente de clustering es diferente
- La distribución del grado es diferente

La conclusión que podemos sacar de aqui es muy simple: Los grafos correspondientes a las redes sociales no son aleatorios!

El estudio del Grafo aleatorio Gn,p es sumamente importante para entender las propiedades de los grafos de las diferentes redes sociales y su comportamiento y significado.

PARTE II ALGORITMOS

En esta sección vamos a estudiar algunos algoritmos específicos para grafos con su aplicación para redes sociales.

Clustering en Redes Sociales

Las relaciones entre usuarios de una red social tienden a formar clusters, sin embargo estos no son explícitos (creados por el usuario) sino que deben ser inferidos automaticamente por las relaciones entre los mismos. Por ejemplo todos los usuarios que son alumnos de una misma universidad generan un pequeño cluster y suelen relacionarse entre sí.

Los algoritmos de clustering en redes sociales son importantes pues nos permiten encontrar comunidades de usuarios, esto sirve para recomendar contactos a usuarios, realizar campañas de publicidad orientadas a una comunidad específica, detectar patrones de comportamiento y muchas otras cosas mas.

Sin embargo los algoritmos de clustering típicos que se suelen usar en otras áreas no suelen tener buenos resultados en redes sociales. Esto se debe a varios factores pero principalmente hay dos motivos:

1) Un usuario en una red social pertenece a varios clusters al mismo tiempo (familia, trabajo, universidad, aficionados a la astronomía, etc)
2) Es muy díficil definir la "distancia" entre dos nodos de una red social.

El problema de la distancia es de especial consideración. En los algoritmos de clustering tradicionales es posible conocer la distancia entre dos nodos o si se usa una distancia euclideana entre un nodo y un punto cualquiera, como el que se usa como centroide en el algoritmo k-means.

En una red social la distancia es todo o nada, dos usuarios o bien están relacionados o bien no lo están, podemos llamar a esto 1 e infinito o como querramos pero cualquier convención que tomemos va a ser problemática para un algoritmo de clustering.

Pensemos por ejemplo en un método jerárquico, el mismo toma los dos nodos mas cercanos y los une, en una red social este algoritmo puede tomar cualquier par de usuarios relacionados y es probable que tome usuarios que en realidad no pertenecen a un mismo cluster.

Para establecer algún algoritmo de clustering que sea efectivo en una red social es necesario definir alguna métrica que nos sirva en reemplazo de la "distancia". Una de las opciones es el concepto de "betweenness". "Betweenness" es un concepto que no pensamos traducir, de hacerlo llegaríamos a

algúna construcción del estilo de "enmedianez" palabra que no solo no existe sino que es muy fea.

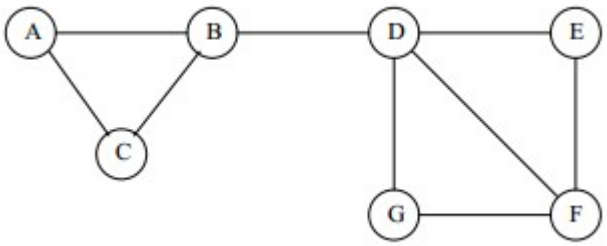
Betweenness

La idea de "betwenness" es intentar detectar que tan probable es que una arista sea una conexión entre dos clusters. Si logramos calcular esto podemos construir un esquema de clustering simple en el cual tomemos siempre las aristas con menor número de "betweenness".

Definición: el betwenness de una arista (a,b) es el número de pares de nodos tales que (a,b) pertenece al camino mínimo entre dichos nodos.

Es decir que si por ejemplo una cierta arista pertenece al camino mínimo entre (x,z)(j,k)(x,k) su betweenness será igual a 3.

Ejemplo:



Podemos calcular los caminos mínimos entre los nodos de este grafo usando Dijsktra, Floyd-Warshall o simplemente a ojo. Notar que AB o BA son la misma arista. Notar tambien que para algunos pares hay mas de un camino minimo. Notar que el camino minimo entre A-G y G-A es el mismo por lo que solo debemos contarlo una vez.

Todos los caminos mínimos son

A-B = AB
A-C = AC
A-D = AB,**BD**
A-E = AB,**BD**,DE
A-F = AB,**BD**,DF
A-G = AB,**BD**,DG
B-C = BC
B-D = **BD**
B-E = **BD**,BE
B-F = **BD**,DF
B-G = **BD**,DG
C-D = CB,**BD**
C-E = CB,**BD**,DE
C-F= CB,**BD**,DF
C-G= CB,**BD**,DG
D-E= DE
D-F= DF
D-G= DG
E-F= EF

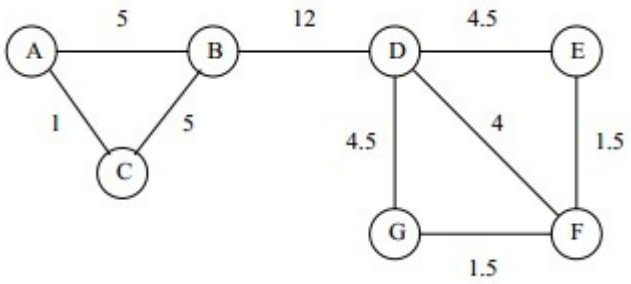
E-G= ED,DG y EF,FG
F-G= FG

Notar que para E-G hay dos caminos mínimos por lo que cada arista cuenta como 0.5 (1/caminos)

Por ejemplo si queremos el "betweenness" para BD vemos que esta en el camino minimo entre A-D, A-E, A-F, A-G, B-D, B-E, B-F, B-G, C-D, C-E, C-F,C-G (en negrita).

Es decir betweenness (B,D) = 12

Para nuestro grafo quedaría:



A partir del betweenness de cada arista podemos aplicar un algoritmo de clustering simple comenzando con un grafo sin aristas y agregando las mismas en orden de betweenness creciente. Es decir que en nuestro caso quedaria:

1 arista: A-C
2 aristas: A-C y E-F
3 aristas: A-C y E-F-G
4 aristas: A-C y D-E-F-G
5 aristas: A-B-C y D-E-F-G
etc...

No hemos visto, hasta ahora que algoritmo se puede usar para calcular el betweenness de cada arista. El algoritmo para calcular el betweenness de cada arista es el de Girwan Newman

Algoritmo de Girwan Newmann

Paso 1

El primer paso del algoritmo es realizar un recorrido BFS del grafo a partir de cada uno de sus nodos. Cada recorrido particiona el grafo en niveles. En nuestro caso las particiones son:

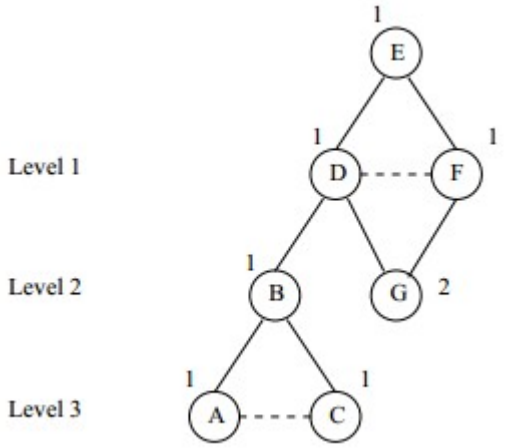
A – BC – D – EFG
B – ACD – EFG
C – AB – D - EFG
D – BEFG – AC
E – DF – BG - AC
F – DEG – B – AC

G – DF – EB – AC

Paso 2

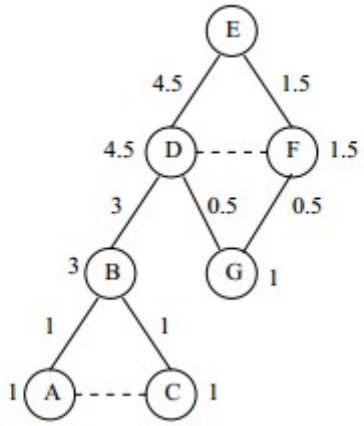
En el segundo paso se rotula cada nodo del grafo comenzando desde el nivel superior se rotula la raiz con "1", luego se procede nivel por nivel. Si el nodo tiene un solo padre su rotulo sera el del padre. Si el nodo tiene n padres su rotulo es la suma de los rotulos de los padres.

En nuestro ejemplo rotulamos "E" con 1, luego D y F con 1 porque solo tienen un padre.
B con 1, G con 2 porque tiene 2 padres. B con y A y C con 1.



Paso 3

En el tercer paso se rotulan nuevamente los nodos y tambien las aristas comenzando desde las hojas. Las hojas se rotulan con 1. Los nodos padres se rotulan con 1+ la suma de los rotulos de sus aristas descendentes.
Las aristas se rotulan en base a su nodo inferior, repartiendo el rotulo del nodo entre las aristas ascendentes.
Por ejemplo A y C reciben rotulo 1 por ser hojas. Y sus aristas hacia B reciben 1 c/u,
B es 1+A+C = 3
La arista de B a D recibe entonces 3.
G tiene rotulo 1 por ser una hoja.
Y como de G suben 2 aristas cada una recibe 0.5
F tiene rotulo 1+0.5
Y asi sucesivamente...



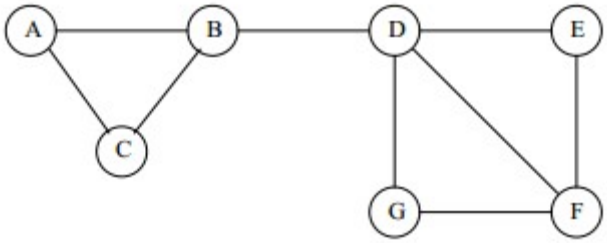
Este procedimiento se hace para todos los nodos del grafo y luego se suman los pesos de las aristas en cada BFS finalmente se divide por 2 porque sino estariamos contando el camino X-Y y tambien el Y-X que son el mismo camino.

Dejamos como ejercicio hacer todo el proceso y llegar a que B-D tiene peso 12 :-)

Cortes en Grafos

Otra forma de particionar un Grafo en clusters es partir del grafo final y remover las aristas que nos proporcionen el mejor "corte" existe un método bastante intrincado para realizar este corte o división en un grafo que explicamos a continuacion.

Vamos a usar como ejemplo el mismo grafo que en los casos anteriores:



Vamos a representar la matriz de adyacencias del grafo indicando 1 o 0 en cada Aij segun exista camino directo entre i y j.

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

por otro lado vamos a tener la matriz que representa el grado de cada nodo:

$$G = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Y calculamos ahora A-G

$$A-G = \begin{bmatrix} -2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -4 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -3 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & -2 \end{bmatrix}$$

La matriz A-G es laplaciana es decir que todas sus filas y columnas suman 0.
Vamos a calcular ahora los autovalores de la matriz:

EigenValues = [0, -0.398, -5.262, -3, -3.340, -4 , -2]

El primer autovalor de una matriz laplaciana es siempre 0, por lo que nos interesa el segundo autovalor y para dicho autovalor el autovector correspondiente.

El autovector es:

EigenVector2 = [0.493, 0.297, 0.493, -0.214, -0.356, -0.356, -0.356]

Notemos que hay 3 elementos positivos: A,B,C y 4 negativos: D,E,F y G. Esta division la podemos usar para particionar la matriz en dos clusters ABC y DEFG.

Si fuera necesario el proceso se repite luego con cada sub-grupo sucesivamente hasta llegar a la cantidad de clusters satisfactoria para el problema en cuestión.

El método aqui presentado se denomina "Analisis espectral" y es muy usado en problemas de particionamiento de grafos.

Notar que ninguno de los métodos hasta ahora presentados admite que un elemento pertenezca a varios clusters y esto es algo muy importante en las redes sociales.

PageRank en redes sociales

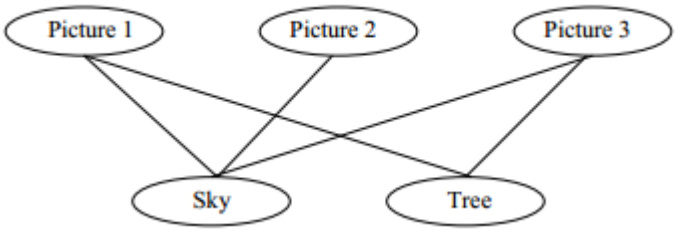
El algoritmo de PageRank se basa en el concepto de un navegante de internet aleatorio que va de página en página navegando los links en las mismas. El mismo concepto puede aplicarse a una red social tomando un navegante al azar que navegue las relaciones entre los usuarios de las mismas. Con una muy pequeña modificación esto puede usarse para encontrar usuarios que son "cercanos" o "parecidos" entre si.

Supongamos que partimos de un cierto usuario "S" y queremos navegar al azar el grafo de relaciones para encontrar a los usuarios mas relacionados con "S", podemos aplicar la misma idea que usamos en PageRank para ranquear en forma temática usando una matriz de teletransportación en donde solo figure el usuario S.

Es decir que la idea es que un navegante al azar parte de S y en cada paso con probabilidad "b" elige un "link" al azar y con probabilidad (1-b) se teletransporta de regreso a S. Luego de esto los nodos con mayor PageRank son aquellos mas similares o relacionados con S.

Si realizamos esto para cada nodo del grafo tenemos una enorme cantidad de datos que nos permiten luego agrupar los usuarios en clusters admitiendo incluso que un usuario pertenezca a mas de un cluster.

Ejemplo:



La matriz de "PageRank" para este grafo es:

$$\begin{bmatrix} 0 & 0 & 0 & 1/3 & 1/2 \\ 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/2 \\ 1/2 & 1 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \end{bmatrix}$$

Recordemos: Desde "Picture1" el usuario puede ir a "Sky" o "Tree" entonces la probabilidad de ir a "Sky" es 1/2 y la de Tree es 1/2. La columna 1 representa las probabilidades desde Picture1, luego Picture2, Picture3, Sky y Tree.

Recordemos que para el PageRank usaremos la formula:

$v' = bMv + (1-b)\ T$

Donde b es el parametro de teletransportacion (0.7 – 1.0 usualmente)
M es la matriz
v es el vector sobre el cual iteramos
T es la matriz de teletransportacion

Supongamos que queremos calcular el PageRank para "Picture1" con b=0.8 (4/5) el calculo quedaria:

$$\mathbf{v}' = \begin{bmatrix} 0 & 0 & 0 & 4/15 & 2/5 \\ 0 & 0 & 0 & 4/15 & 0 \\ 0 & 0 & 0 & 4/15 & 2/5 \\ 2/5 & 4/5 & 2/5 & 0 & 0 \\ 2/5 & 0 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v} + \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Notar que multiplicamos cada elemento de M por 4/5 (0.8) y que el vector de teletransportacion solo tiene 1 en "Picture1" y que a este valor lo multiplicamos por 1-b = 0.2 = 1/5

Iterando la fórmula nos queda:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1/5 \\ 0 \\ 0 \\ 2/5 \\ 2/5 \end{bmatrix} \begin{bmatrix} 35/75 \\ 8/75 \\ 20/75 \\ 6/75 \\ 6/75 \end{bmatrix} \begin{bmatrix} 95/375 \\ 8/375 \\ 20/375 \\ 142/375 \\ 110/375 \end{bmatrix} \begin{bmatrix} 2353/5625 \\ 568/5625 \\ 1228/5625 \\ 786/5625 \\ 690/5625 \end{bmatrix} \cdots \begin{bmatrix} .345 \\ .066 \\ .145 \\ .249 \\ .196 \end{bmatrix}$$

Podemos notar que "Picture3" es mas parecido a "Picture1" que "Picture2" lo cual tiene sentido si analizamos el grafo con el cual estabamos trabajando.

Analisis de Comunidades

Supongamos ahora que tenemos una lista de nodos en una red social que forman una cierta comunidad. Queremos analizar si esta lista es realmente una comunidad o no. Alternativamente podriamos querer encontrar cual es la comunidad mas fuerte de una serie de candidatos. En definitiva buscamos una métrica para analizar el valor como comunidad de una serie de nodos.

La forma de realizar esto es sorprendentemente simple y consiste en contar la cantidad de triangulos que existen dentro de la comunidad.
Un triangulo es de la forma A-B, B-C, A-C.

La lógica detrás de este método tan sencillo es muy simple, en una comunidad en donde A-B y B-C estan relacionados tarde o temprano A y C van a descubrirse y es muy probable que se cree una relación entre los mismos. Estudios realizados muestran que a medida que una comunidad madura la cantidad de triángulos en la misma crece.

Para contar triángulos en una comunidad debemos encontrar links de la forma

A-B B-C A-C

Si suponemos que tenemos un archivo o lista de links de la forma "Origen-Destino" podemos realizar el conteo de triángulos mediante un triple join del archivo consigo mismo.

En SQL la consulta seria algo de la forma:

```
SELECT count(*)
FROM links A, links B, links C
WHERE A.destino = B.origen AND A.origen=C.origen and B.destino = C.destino
```

Como hemos visto anteriormente los "JOINs" de un archivo consigo mismo pueden resolverse eficientemente con MapReduce dejamos como ejercicio para el lector programar en MapReduce las tareas necesarias para contar la cantidad de triángulos en una cierta comunidad.