

Caracterización de Redes y Topologías Biológicas

Resumen

En esta asignatura se estudian los principales tipos de conectividad que se pueden dar en una red biológica. Se describen además cuales puede ser la mejor estrategia de conexión entre los elementos de una red sujetos a una determinada dinámica. También se proporcionan métodos para calcular los principales parámetros topológicos y de rendimiento de una red dada. Además se estudian redes resistentes a una determinada estrategia de ataque o frente a errores en la red.

Índice general

I	Introducción y descripción de algunas redes reales	3
I.1	Qué es una red	3
I.2	Algunos ejemplos de redes y algunas de sus propiedades	3
I.2.1	World Wide Web	3
I.2.2	Internet	4
I.2.3	Red de actores	5
I.2.4	Red de colaboración científica	5
I.2.5	Red de contactos sexuales	5
I.2.6	Red de llamadas telefónicas	5
I.2.7	Redes lingüísticas	6
I.2.8	Redes eléctricas	6
I.3	Algunos ejemplos de redes biológicas y algunas de sus propiedades	6
I.3.1	Redes de ecología	6
I.3.2	Redes celulares	6
I.3.3	Redes neuronales	7
I.3.4	Redes de interacción de proteínas	7
I.3.5	Redes genéticas	7
II	Teoría de grafos y métricas	8
II.1	Introducción a la teoría de grafos	8
II.2	Bucles y ramas paralelas	10
II.3	Grafos dirigidos y ponderados	10
II.4	Grado de un nodo	11
II.5	Subgrafos	12
II.6	Paseos, caminos, circuitos y ciclos	12
II.7	Medidas de centralidad, betweeness y closeness	13
II.8	Conexidad	14
II.9	Bosques y árboles	15
II.10	Grafos bipartitos	16
II.11	Representación de grafos	16
II.12	Métricas sobre grafos	16
II.12.1	Coeficiente de agrupamiento C	17
II.12.2	Camino característico L	18
II.13	Topologías	18
II.13.1	Grafos aleatorios	18
II.13.2	Grafos regulares	19
II.13.3	Mundo pequeño	19
II.13.4	Grafos libres de escala	19
II.14	Algoritmos sobre grafos	19

II.15	Cálculos sobre grafos: NetworkX	20
II.15.1	Creación de grafos	20
II.15.2	Importación y exportación de grafos	21
II.15.3	Información sobre el grafo	23
II.15.4	Visualización	24
II.15.5	Algoritmos	25
III	Grafos aleatorios, grafos regulares y redes de mundo pequeño	29
III.1	Grafos aleatorios	29
III.1.1	Modelo de Erdős y Rényi	29
III.1.2	Propiedades	29
III.1.3	Subgrafos	30
III.1.4	Clusters	30
III.1.5	Distribución de grado	31
III.1.6	Conexidad y diámetro	31
III.1.7	Índice de clusterización	31
III.2	Grafos regulares	31
III.3	Redes de mundo pequeño	32
III.3.1	Sustrato inicial	32
III.3.2	Sustratos bi-conexos	32
III.3.3	Sustratos dirigidos y ponderados	33
III.3.4	Parámetros	33
IV	Grafos libres de escala, ataques a redes y aplicaciones	34
IV.1	Grafos libres de escala	34
IV.1.1	Modelo de Barabasi y Albert	34
IV.2	Ataques a redes	35
IV.2.1	Ataques óptimos	36
IV.2.2	NP-Complejidad	36
IV.2.3	Ataques mediante algoritmos aproximados	36
IV.2.4	Topologías	37
IV.3	Aplicaciones	37
IV.3.1	Click - algoritmo de clustering	37
IV.3.2	ProtoMap	38
IV.3.3	Redes de interacción	38

Capítulo I

Introducción y descripción de algunas redes reales

Aunque lo vayamos a utilizar como sinónimos, un grafo y una red no es lo mismo; el grafo es la representación matemática de la red. En una red aleatoria, no hay que medir nada; si una red biológica sale aleatoria, se ha medido mal. Las redes biológicas son todas de mundo pequeño. Además, casi todas son libres de escala.

I.1. Qué es una red

Una red es un conjunto de elementos (personas, ciudades, proteínas, especies animales, productos químicos, etc) de las cuales algunas están conectadas con otras y otras no. Se puede representar en bolas que se unen con líneas con otras líneas. Las bolitas se denominan como nodos.

Las redes se estudian con NetworkX y Cytoscape.

I.2. Algunos ejemplos de redes y algunas de sus propiedades

I.2.1. World Wide Web

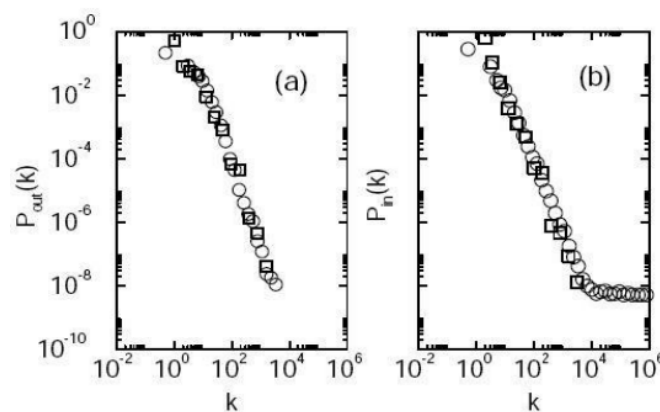
La World Wide Web es la mayor red para la cual existe información topológica. Los nodos de la red son los documentos, y las ramas son los enlaces (hyperlinks) entre documentos. El tamaño actual de esta red es de más de 1000 millones de nodos. Esta red es dirigida: la página A apunta a la página B, pero sin que la página B apunte a la página A. El grupo CAIDA se dedica a analizar la red. Esta red es enorme, pudiendo dibujar solo a un nivel muy alto.

La distribución del grado de las páginas web tiene una distribución libre de escala tanto en los enlaces de salida como en los enlaces de entrada. Esto es una distribución de probabilidad. Por ejemplo, en el queso de Gruyere, los agujeros son de distinto tamaño, los cuales tienen una distribución de tamaño. Se llama libre de escala porque

se pueden encontrar diez veces más los agujeros de un tamaño mayor y diez veces menos los agujeros de tamaño pequeño. Así, no hay una escala fija de la distribución (no se puede representar con ninguna escala, ni logarítmica ni nada). Esto con el queso manchego no pasa. Si en la WWW vemos cuántas páginas web tienen 100 enlaces de salida, 10, 1000, etc, y se dibuja en escala logarítmica logarítmica, sale una recta. Esto pasa también con los enlaces de entrada.

La distancia entre dos páginas de la WWW es pequeña (entre 11 y 16). Los nodos de la WWW están muy clusterizados.

La cola de la derecha parece que rompe la recta. Las redes libres de escala se pueden producir por muchas razones, pero al utilizar un proceso evolutivo en el que cada tiempo se generan nuevos nodos y tienen mayor preferencia para conectarse a otros nodos, e incluso pueden desaparecer algunos nodos antiguos. Esto produce las colas residuales.



Las redes libres de escala son muy resistentes a ataques aleatorios (fallos en la red) en cuanto a la conectividad, por lo que hay una razón evolutiva por la que las redes biológicas son libres de escala. La red regulatoria de P53 está muy estudiada y caracterizada. Uno de los elementos más importantes es MDM5.

1.2.2. Internet

Internet es la red de enlaces físicos entre ordenadores u otros servicios de comunicación. La topología de internet se suele estudiar a dos niveles: Enrutadores y Sistemas autónomos. Los enrutadores son las máquinas que mandan los "paquetes" a otros enrutadores. Hay algoritmos de enrutación que deciden hacia dónde enviar las cosas. Los sistemas autónomos son conjuntos de máquinas que organizan y gestionan otras máquinas.

Para ambos tipos de red (enrutadores y sistemas autónomos) el grado de cada nodo seguía una distribución libre de escala. De nuevo la red está altamente clusterizada (coeficiente de clustering entre 0,18 y 0,3) y los caminos entre nodos son cortos (aproximadamente 9).

El **índice de clusterización** es una medida de la probabilidad de que los dos vecinos de un nodo sean vecinos entre sí, favoreciendo la creación de triángulos. Es decir, en redes sociales, que mis amigos también sean amigos entre sí. En biología, si dos

proteínas son expresadas por una tercera proteína, las dos mantienen una relación entre sí (aunque puede no pasar). Los vecinos de un mismo nodo tienen una probabilidad alta de ser vecinos entre sí. En una red aleatoria, los vecinos de un nodo dependen de la probabilidad de rama de que esos nodos también sean vecinos entre sí (como cualquier otro).

La métrica de caminos cortos o largos se hace en comparación con el grafo aleatorio con el mismo número de nodos y ramas. En biología, los caminos también suelen ser cortos, y si son largos se puede deber a una enfermedad o patología.

1.2.3. Red de actores

Los nodos son actores, y dos de ellos están conectados si han participado juntos en alguna película. Actualmente, la red consta de unos 450.000 actores. La distancia media entre actores es 3,65. La red está altamente clusterizada (100 veces más que un grafo aleatorio). La distribución de grados sigue una ley de potencias (libre de escala).

1.2.4. Red de colaboración científica

Los nodos están constituidos por científicos. Dos nodos están conectados si alguna vez publicaron un trabajo en común. La red de nuevo presenta una distribución libre de escala, caminos cortos entre los nodos y una alta clusterización.

El **centro de la red** es el nodo que está a una menor distancia promedio del resto de nodos de la red. Este centro lo tiene un científico húngaro llamado Paul Erdős que trabajaba en teoría de grafos.

Para una red de citaciones científicas, los nodos de la red son artículos científicos. Las ramas son citaciones entre artículos. Se tiene una base de datos de unos 750.000 artículos. Tanto los grados de entrada como los de salida siguen una distribución libre de escala.

1.2.5. Red de contactos sexuales

Los nodos y las ramas tienen una definición obvia. Tiene interés por la difusión de enfermedades (especialmente aquellas de transmisión sexual como el SIDA). Presenta una distribución libre de escala. Se sospecha que los datos de esta red no son totalmente fiables (es defectuosa al tener muchos datos falsos). Entre un 10-15 % es falsa.

Se define como k-core un grafo no dirigido creado a partir de un grafo más grande en el que se crean jerarquías o grupos en el que los nodos están separados por k vértices.

1.2.6. Red de llamadas telefónicas

Los nodos son números de teléfono. Las ramas son llamadas de larga distancia entre nodos. De nuevo la red presenta una distribución libre de escala.

I.2.7. Redes lingüísticas

Los nodos son palabras. Dos nodos están conectados si están juntas en alguna frase y hay solamente una palabra entre ambas. Un estudio realizado en inglés sobre 440.902 palabras presentó una distancia media de 2,62 y un índice de clusterización de 0,43.

Otra red lingüística considera de nuevo los nodos como palabras. Dos nodos están conectados si se considera que ambas palabras son sinónimas (de acuerdo con el Merrian Webster Dictionary). El camino medio es de 4,7, el índice de clusterización es de 0,7 y los nodos presentan una distribución libre de escala.

En la red semántica, cada nodo es un objeto o un concepto. Dos nodos se relacionan entre sí, si existe una relación de la forma "es un" o "tiene un" entre ambos nodos. Se ha estudiado poco, pero parece presentar un camino medio corto, alta clusterización y una distribución de nodos libre de escala.

I.2.8. Redes eléctricas

La red eléctrica del Oeste de los Estados Unidos está compuesta por nodos (generadores, transformadores y subestaciones) y ramas (cables físicos entre nodos). La red tiene 4.941 nodos y un grado medio por nodo de 2,41. Esta red se aparta del patrón habitual teniendo una estructura muy jerárquica y en forma de estrella. Esto hace que sea muy frágil y condicionada a cuestiones económicas y políticas. Ocurre de forma similar con las redes de internet. No se utiliza el camino más rápido o corto, si no el camino más barato (como a la hora de buscar vuelos).

I.3. Algunos ejemplos de redes biológicas y algunas de sus propiedades

I.3.1. Redes de ecología

En las redes alimentarias, los nodos de la red son especies, y las ramas relaciones predador-presa entre especies. Las distancias son cortas entre los elementos de la red. En general, son redes con pocos nodos.

Al ser redes pequeñas es difícil dibujar la distribución del grado de los nodos. Parecen presentar una distribución libre de escala, con un exponente inusualmente pequeño. Esta red es dirigida (aunque pueda haber dobles ramas).

I.3.2. Redes celulares

Se presentan al estudiar el metabolismo de organismos. Los nodos son sustratos químicos (ATP, ADP, etc), y las ramas presentan reacciones químicas entre los sustratos. Esta red va de arriba a abajo, empezando con unos productos de entrada de la célula y terminando con productos de salida que la célula no puede descomponer más.

I.3.3. Redes neuronales

Cada nodo es una neurona (biológica o artificial), y las ramas son conexiones sinápticas entre neuronas. La primera red estudiada de este tipo es la del gusano *Caenorhabditis elegans*, del cual se tiene el mapa neuronal completo.

Las redes neuronales artificiales están ahora en auge para las inteligencias artificiales al utilizarse para el aprendizaje profundo.

I.3.4. Redes de interacción de proteínas

Cada nodo es una proteína. Las ramas representan relaciones de expresión entre las proteínas. Una de las redes más importantes es la red p53 de control de crecimiento del cáncer. Un paper muy bueno es [Surfing the p53 network \(DOI 10.1038/35042675\)](https://doi.org/10.1038/35042675).

Esta es la red en la que más se trabaja en biología. Se buscan los efectos entre los nodos (aumenta la expresión, inhibe), los componentes clave, los parámetros, etc.

I.3.5. Redes genéticas

Cada nodo es expresión genética (nucleótidos). Las ramas conectan los nucleótidos que presentan un alto índice de similitud entre ambas. Una vez representada la red, se buscan familias o grupos de genes similares. Hay que diferenciar identidad con similitud (sobre todo con desajuste de fase). Se utiliza programación dinámica para calcular la mayor longitud de subsecuencia idéntica, como por ejemplo con el algoritmo Soldier's Walk.

Si clusterizamos y obtenemos 2 cluster, cada cluster indica un gen con errores, o dos individuos distintos. Luego hay que interpretar por qué hay ese número de cluster. Normalmente hay muchos clusters que se quieren clasificar, y en cada cluster suele aparecer el mismo gen que se ha mutado.

Las máquinas de microarrays ahora dan un conjunto de nucleótidos muy grandes, pero antes se obtenían fragmentos que había que unir. Para ello, se debían utilizar algoritmos sobre grafos para calcular cadenas largas a partir de las cadenas cortas, pero ahora ya no se usa por las mejoras tecnológicas.

Capítulo II

Teoría de grafos y métricas

II.1. Introducción a la teoría de grafos

La teoría de grafos ha sido utilizada recientemente para:

- Clasificación automática de secuencias de proteínas.
- Detección de jerarquías de proteínas.
- Análisis de redes genéticas.
- Reconstrucción de redes genéticas grandes obtenidas mediante modificación de genes.

Un grafo G es un par de conjuntos (V, E) donde $V = \{v_1, v_2, \dots, v_n\}$ es el conjunto de vértices o nodos y $E = \{(v_i, v_j), (v_{i'}, v_{j'}), \dots\}$ es un conjunto de pares no ordenados de elementos de V y se denomina conjunto de ramas del grafo. El número de nodos se denomina **orden** del grafo, y el número de ramas es el **tamaño** del grafo.

Pregunta de test: define orden y tamaño, dado un grafo dar el orden y tamaño, etc.

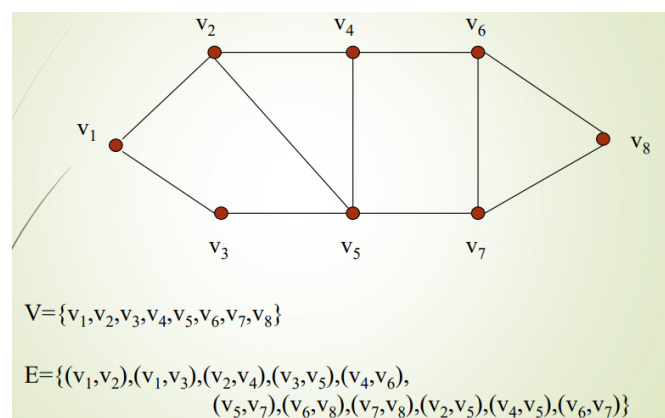
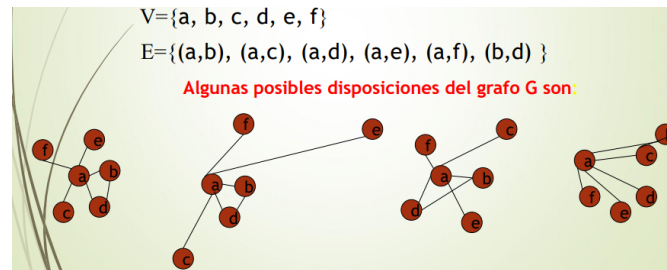


Figura II.1: Ejemplo de grafo de orden 8 y tamaño 11.

Para una red de proteínas, cada proteína sería un nodo del grafo, y una rama indicaría interacción entre ambas proteínas.

Una disposición (layout) es una posible colocación de los nodos y las ramas en un espacio 2D o 3D. Un mismo grafo puede tener múltiples colocaciones. Ejemplo, consideremos el grafo $G=(V,E)$.



Existen programas de ordenador que nos permiten obtener colocaciones predefinidas (Gephy, Pajek). Cuando no se especifica ninguna colocación, se entiende que los nodos se sitúan aleatoriamente sobre el plano o espacio. Algunos de los tipos más habituales de colocaciones son:

- Colocaciones regulares
- Basadas en la física (atracción-repulsión)
- Basadas en propiedades topológicas (jerarquías, número de vecinos, etc)

Un hipergrafo H es un también par de conjuntos (V,E) donde $V = \{v_1, v_2, \dots, v_n\}$ es el conjunto de vértices o nodos y $E = \{(v_{i1}, v_{i2}, \dots), (v_{i'1}, v_{i'2}, \dots), \dots\}$ es una familia de subconjuntos no ordenados de elementos de V . E se denomina conjunto de hiperramas o hiperaristas del hipergrafo. El número de hiperramas $|E|$ se denomina cardinalidad del hipergrafo. El valor $|E| * |V|$ se denomina tamaño o volumen del grafo. Si tenemos un grafo de n nodos, ¿cuántas parejas podemos tener como máximo? $(n \cdot n - 1)/2$ Por tanto, en un grafo con n nodos, ¿cuántas ramas puede tener? Igual, $(n \cdot n - 1)/2$

Pregunta
examen

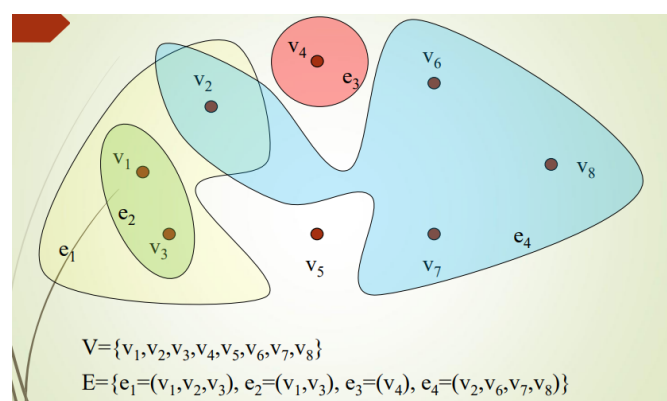


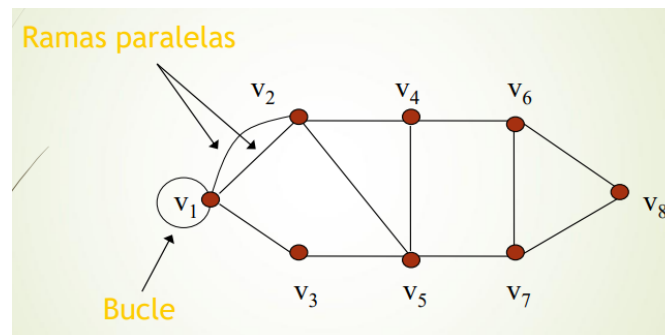
Figura II.2: Ejemplo de hipergrafo de cardinalidad 4 y tamaño 32.

Un hipergrafo H se dice que es **propio** si no es vacío ($V \neq \emptyset$) y no contiene ninguna arista vacía. Un hipergrafo H se dice que tiene **dominio completo** si todos los nodos están en al menos una arista, en caso contrario se dice que tiene **dominio parcial**. Si en un hipergrafo todas las hiperramas tienen el mismo número de nodos, entonces se denomina **hipergrafo k-uniforme**.

Ejercicio: Indicar si el hipergrafo del ejemplo anterior es propio, tiene dominio completo y si es k uniforme. Es propio (el conjunto de vértices tiene 8 elementos y todas las ramas e tienen vértices dentro), es de dominio parcial (v_5 no está en ninguna rama) y no es k -uniforme (e_1 tiene 3 elementos, e_2 tiene 2, e_3 tiene 1 y e_4 tiene 4).

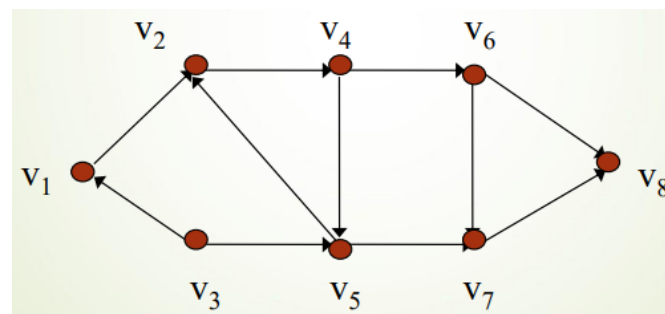
II.2. Bucles y ramas paralelas

Un bucle es una rama que empieza y termina en el mismo nodo (v_i, v_i). Cuando dos ramas conectan el mismo par de vértices se denominan paralelas. Un grafo con bucles se denomina pseudografo. Un grafo con ramas paralelas pero sin bucles se denomina multigrafos. Un grafo sin bucles ni ramas paralelas se denomina grafo simple.

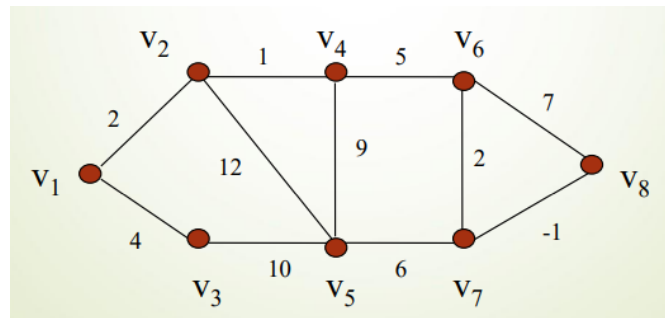


II.3. Grafos dirigidos y ponderados

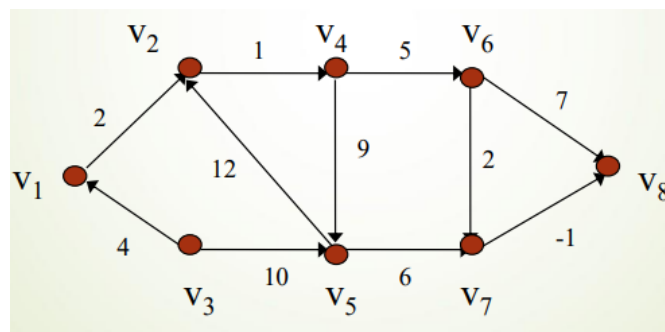
Se puede considerar que los enlaces entre nodos son dirigidos (v_i, v_j) = (v_j, v_i). Los grafos dirigidos se denominan también **digrafos**.



En los grafos ponderados, a cada rama del grafo se le puede asociar un número. El número asociado a cada rama puede indicar entre otras cosas una distancia, una capacidad, un valor temporal, etc.

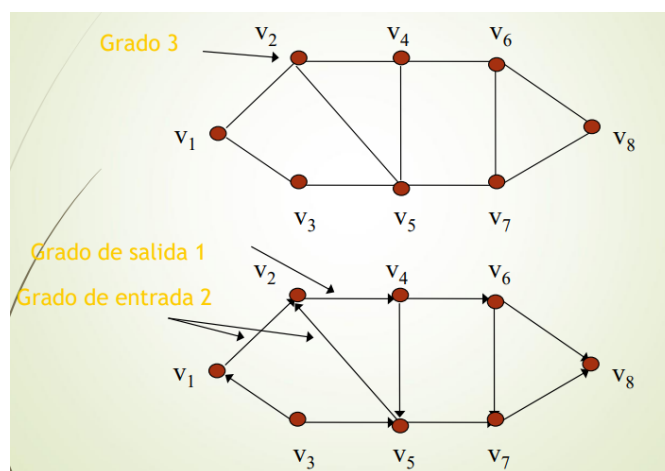


Los grafos dirigidos y ponderados poseen ramas dirigidas a las que se asocia un número.



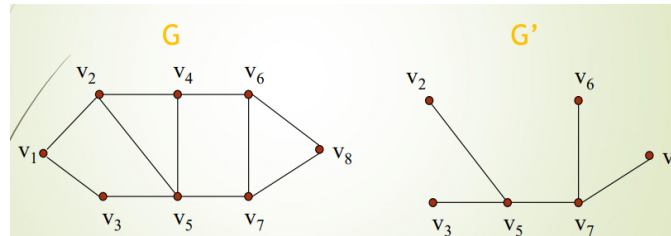
II.4. Grado de un nodo

Dos nodos de un grafo son **vecinos o adyacentes** si existe una rama que los conecta. El **grado** de un nodo es el número de vecinos que tiene dicho nodo. En los grafos dirigidos se calcula el **grado de entrada** y el **grado de salida**. En los grafos ponderados, el grado se puede promediar por el número asociado a las ramas. Un grafo se dice que es **regular** si todos los nodos tienen el mismo grado.



II.5. Subgrafos

Un grafo $G'=(V',E')$ es un subgrafo de un grafo $G=(V,E)$ si V' es un subconjunto de V y E' es un subconjunto de E . En otras palabras, un subgrafo es un trozo de un grafo más grande.



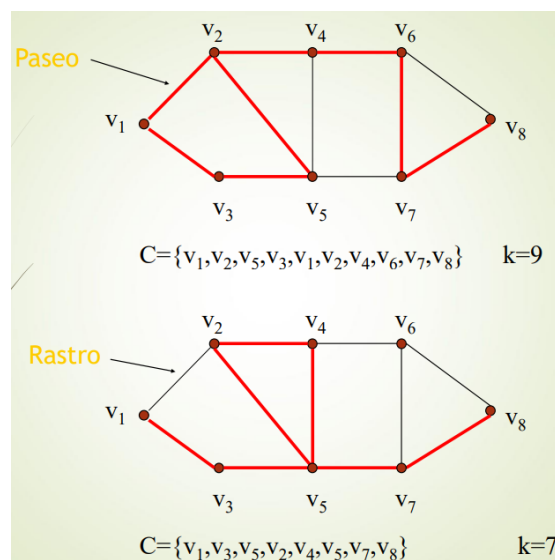
Un subgrafo $G'=(V',E')$ de un grafo $G=(V,E)$ se dice que es **abarcador** si $V=V'$, es decir, si están todos los nodos, pero faltan algunas ramas.

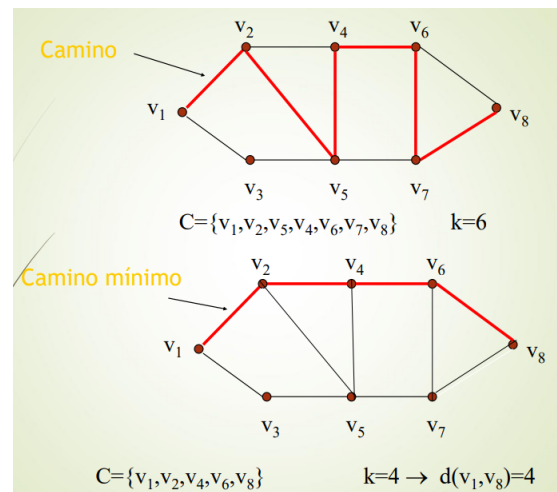
Un grafo es un subgrafo de sí mismo. Además, un grafo vacío es un subgrafo de cualquier grafo.

II.6. Paseos, caminos, circuitos y ciclos

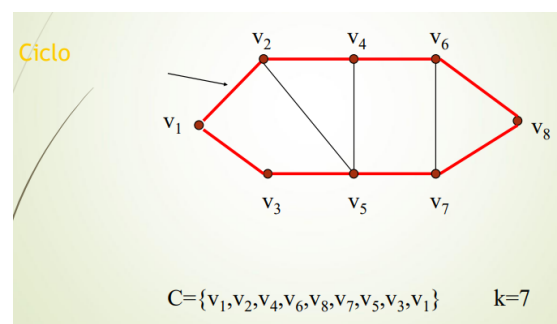
Un **paseo** de un nodo u a un nodo v es una secuencia de vértices $\{v_0, v_1, \dots, v_k\}$ con $v_1 = uv_k = v$ y (v_{i-1}, v_i) rama del grafo. El número de ramas del paseo es su **longitud**. Un paseo en el cual no se repiten ramas se denomina **rastro**. Un paseo en el cual todos los vértices $\{v_0, v_1, \dots, v_k\}$ son distintos se denomina **camino**. Un camino siempre debe ser un rastro y un paseo. Si algo no es rastro, no puede ser camino, y si no es paseo, no puede ser ni rastro ni camino. Cada uno es cada vez más restrictivo.

Entre dos nodos, puede haber varios caminos posibles. Un **camino mínimo** entre dos nodos es aquel de menor longitud de entre todos los posibles caminos entre ambos nodos. La **distancia** entre dos nodos del grafo se define como la longitud de cualquier camino mínimo que los una.





Un **paseo cerrado** es un paseo $\{v_0, v_1, \dots, v_k\}$ tal que $v_0 = v_k$. Un paseo cerrado en el que no se repiten ramas es un **circuito**. Un **ciclo** es un circuito en el que no se repiten vértices. Los ciclos son importantes, porque las redes biológicas tienen ciclos (que suelen ser largos), pero en las redes aleatorias no aparecen ciclos, o éstos son muy pequeños.



El nodo con menor distancia entre los demás es muy importante, denominándose como **centro del grafo**.

Para un grafo con excesivos nodos, los caminos mínimos y las distancias se calculan con un algoritmo. Si el grafo es no ponderado, se utiliza el algoritmo búsqueda en anchura, mientras que si es ponderado, utiliza Dijkstra.

II.7. Medidas de centralidad, betweenness y closeness

Pregunta examen:
Betweenness/Closeness/Farness se define como...

Dado un nodo v_i se define su **betweenness** $C_B(v_i)$ como la fracción de caminos mínimos que hay entre el resto de nodos del grafo y que pasan por el nodo v_i . Es decir, se hacen parejas de todos los nodos del grafo excluyendo el nodo de interés, y se calculan los caminos mínimos. Algunos pasarán por el nodo de interés, que son los que nos quedamos. Con eso se evalúa el cociente (los que pasan por ese nodo entre todos), que será el betweenness (un valor entre 0 y 1). La centralidad de un nodo es muy costosa de calcular, usualmente se emplean algoritmos aproximados.

Dado un nodo v_i se define su **lejanía o farness** $C_F(v_i)$ como la suma de las distancias de v_i al resto de nodos del grafo.

Dado un nodo v_i se define su **cercanía o closeness** $C_C(v_i)$ como la inversa de su lejanía $C_C(v_i) = 1/C_F(v_i)$.

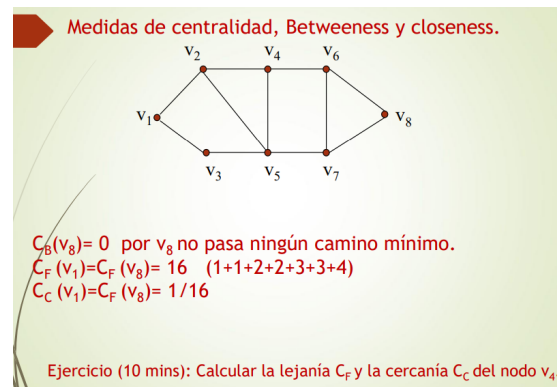


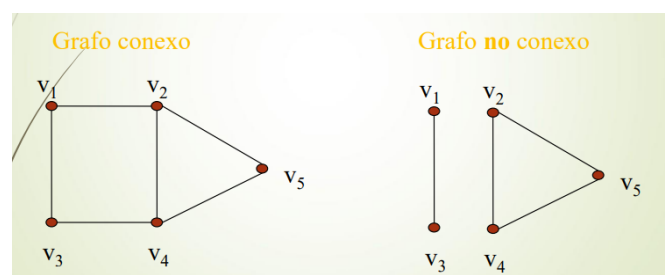
Figura II.3: Respuesta al ejercicio: Cogiendo v_4 , la lejanía será $2+1+2+1+1+2+2 = 11$, y la cercanía $1/11$.

La cercanía y lejanía tiene un problema: su valor numérico depende del orden del grafo. Por tanto, sirve para comparar dentro del mismo grafo, pero no entre grafos. Para eso, habría que normalizar dividiendo por el número total de nodos. A esto se le conoce como **camino característico**.

Pregunta
examen:
Calcular
camino
caracterís-
tico

II.8. Conexidad

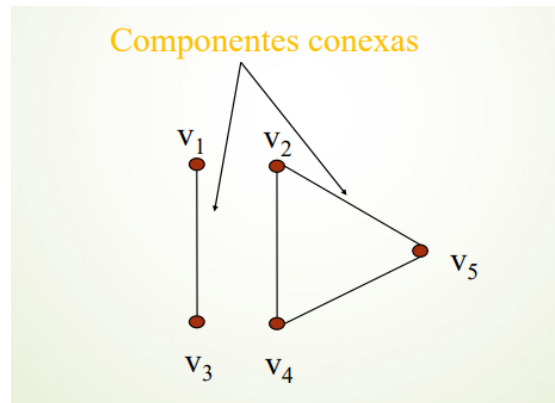
Un grafo es **conexo** si para cada par de nodos del grafo existe al menos un camino que los une. En otras palabras, que no esté separado en distintos trozos.



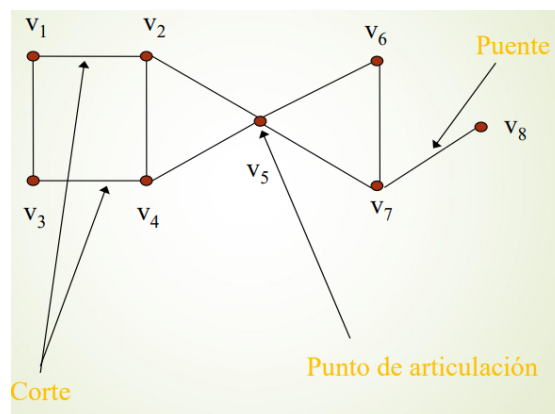
Hay un algoritmo muy rápido y eficiente que calcula si un grafo es conexo o no.

Una **componente conexa** de un grafo es cada uno de los subgrafos maximales conexos. Esto quiere decir que el subgrafo no puede ser más grande, que no se le puede añadir más nodos.

Un **punto de articulación** es un nodo que desconecta un grafo conexo. Un **corte** es un conjunto de ramas que desconecta un grafo conexo. Si un corte está compuesto por una única rama, se denomina **punto de articulación**. Un **corte mínimo** de un grafo es el mínimo número de ramas que al ser eliminadas desconectan el grafo.



El algoritmo CLICK (CLuster Identification via Connectivity Kernels) calcula una aproximación al corte mínimo. Esto lo hacían cogiendo los dos nodos más lejanos. Los puentes suelen ser muy malos para la conexidad de los grafos.

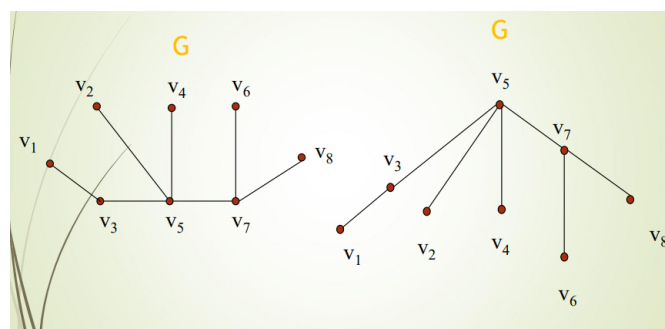


La máxima distancia entre cualquier par de nodos se denomina como diámetro.

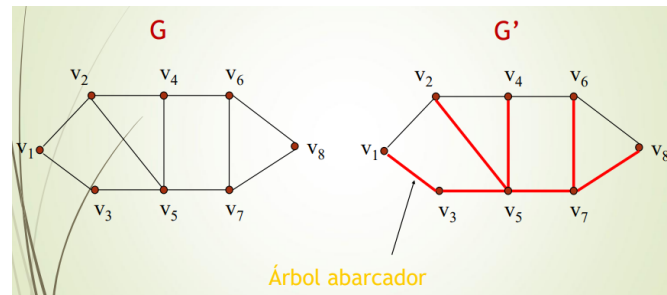
El corte mínimo entre dos nodos es siempre mayor que el corte mínimo de todo el grafo.

II.9. Bosques y árboles

Un grafo sin ciclos (acíclico) se denomina bosque. Un árbol es un grafo acíclico conexo. Cada componente conexas de un bosque es un árbol.



Un subgrafo abarcador acíclico de un grafo G se denomina un **bosque abarcador**. Un subgrafo abarcador conexo acíclico de un grafo G se denomina un **árbol abarcador**.



II.10. Grafos bipartitos

Un grafo se dice que es bipartito si:

- El conjunto de vértices V se puede romper en dos subconjuntos disjuntos V_1 y V_2 .
- El vértice inicial de cada rama de E pertenece a V_1 y el vértice final a V_2

Ejercicio: el grafo anterior (el del árbol abarcador), ¿es bipartito? No, porque no es posible realizar la partición. Si V_1 pertenece al conjunto 1, V_2 y V_3 deben estar en el conjunto 2, por lo que V_4 y V_5 tienen que estar en V_1 , pero esto no es posible porque están conectados entre sí. La condición necesaria para que un grafo sea bipartito es que no tenga triángulos. Pero esto no es suficiente; se puede construir un grafo sin triángulos, pero que tampoco sea bipartito. Si cogemos solo el cuadrado V_4 - V_7 , sí se podría generar un grafo bipartito: V_4 y V_7 en un conjunto y V_5 y V_6 en otro.

II.11. Representación de grafos

Hay dos formas estándar de representar un grafo en un ordenador:

- **Matriz de adyacencia:** consume mucha memoria, pero es fácil de añadir o eliminar ramas. Es fácil saber si existe una rama, pero es lento enumerar los vecinos de un nodo. Se pueden calcular los autovalores y autovectores.
- **Lista de adyacencia:** tiene un consumo limitado de memoria, pero es costoso añadir o eliminar ramas. También es costoso saber si existe una rama, pero rápido enumerar los vecinos de un nodo.

II.12. Métricas sobre grafos

Los grafos se clasifican en función de unas determinadas métricas topológicas. Las métricas más empleadas son:

- Tamaño $|E|$ y orden $|V|$

- Dispersión: $\frac{2|E|}{|V|(|V|-1)}$ para un grafo no dirigido y $\frac{|E|}{|V|(|V|-1)}$ para un grafo dirigido. Si el coeficiente es pequeño (0), el grafo es disperso, si es cercano a 1, es denso. En redes biológicas, los grafos suelen ser dispersos.
- Distribución del grado de los nodos: división del grado de todos los nodos entre el número de nodos. El resultado es una distribución de probabilidad. En un grafo aleatorio, la distribución es de Poisson (como la gaussiana, pero sin valores negativos). En las redes biológicas, la distribución no será de Poisson, por lo que este será el primer test que se haga a los datos.
- Grado medio ($\langle k \rangle$): media del grado de todos los nodos.
- Coeficiente de agrupamiento (C)
- Camino característico (L)

Pregunta
examen:
Calcular
C o L de
un grafo

II.12.1. Coeficiente de agrupamiento C

El coeficiente de agrupamiento (C) es un valor métrico **local** que mide el nivel de agrupamiento de los nodos. Es decir, mira un nodo y sus vecinos y mira el índice de clusterización. En redes biológicas, el índice de clusterización es alto. Para cada nodo v del grafo se obtiene su vecindario, es decir, el conjunto de nodos que son vecinos de v , el tamaño del vecindario coincide con el grado de v (k_v). Se calcula el coeficiente

$$C_v = \frac{N_v}{k_v(k_v - 1)/2} = \frac{\text{número real de ramas entre vecinos sin incluir el nodo } v}{\text{número máximo de ramas entre vecinos}}$$

donde N_v es el número de ramas que hay entre los vecinos de v . El valor anterior se promedia entre todos los nodos del grafo.

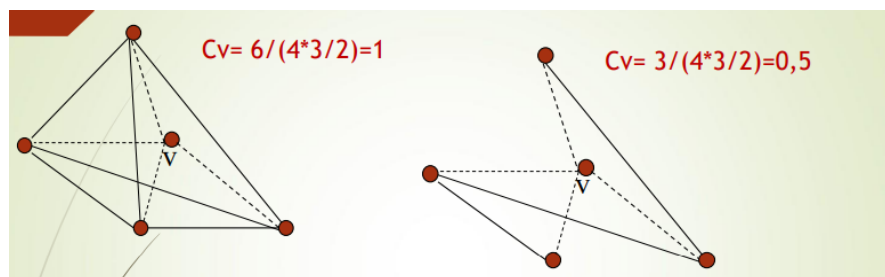


Figura II.4: Figura de la izquierda: todos los vecinos del nodo v son vecinos entre sí. Figura de la derecha: la mitad de mis amigos son amigos entre sí (0,5).

Para un grafo, se calcula el coeficiente de agrupación es el valor de cada nodo dividido por el número de nodos - es decir, el promedio. Para simetría, se calcula el valor de una fracción de los nodos y se divide por esa fracción.

Ejercicio: calcular el coeficiente de agrupamiento C del siguiente grafo. Se sugiere utilizar simetrías para reducir el trabajo. Esto será igual en el examen.

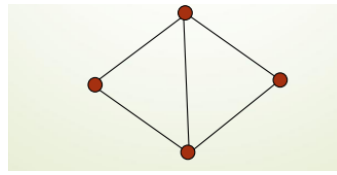


Figura II.5: Se deben calcular dos índices de clusterización: nodo superior (igual al nodo inferior) y nodo izquierdo (igual al nodo derecho). Empezando por el nodo superior, la fórmula quedaría como $2/(3 * 2/2) = 2/3 = 0,66$. Para el nodo izquierdo, quedaría $3/(3 * 2/2) = 1$. Ahora hay que sumar esos dos valores y dividir entre el número de nodos calculados: $(1 + 0,66)/2 = 1,66/2 = 0,83$.

II.12.2. Camino característico L

El camino característico (L) es un valor métrico **global** que mide el nivel grado de separación de los nodos. Para cada nodo v se calcula la distancia promedio a todos los demás nodos del grafo:

$$L_v = \sum_{k=1}^{|v|} d(v, v_k) / (|v| - 1)$$

Se calcula el promedio del valor anterior entre todos los nodos del grafo. Es como un farness normalizado al número de nodos del grafo.

$$L = \sum_{v=1}^{|v|} L_v / (|v| - 1)$$

En las redes biológicas, el camino característico suele ser cortito.

Ejercicio: Calcular el camino característico L del grafo de la figura anterior. Se puede (y debe) volver a utilizar simetrías.

$$L_{v1} = L_{v3} = 1/3 + 1/3 + 1/3 = 1$$

$$L_{v2} = L_{v4} = 1/3 + 1/3 + 2/3 = 4/3 = 1,33$$

$$L = 1/4 + 1,33/4 + 1/4 + 1,33/4 = 7/6 = 1,165$$

II.13. Topologías

II.13.1. Grafos aleatorios

Fueron estudiados principalmente por Erdos y Renyi en los años 50. Cada rama del grafo existe con una determinada probabilidad p . Erdos y Renyi estudiaron los valores de las métricas topológicas para diferentes valores de p . Para la grafos dispersos (p pequeña) se puede comprobar que tanto C (aproximadamente 0) como L (aproximadamente $\ln(|V|)$) son pequeños.

II.13.2. Grafos regulares

Son los mejor conocidos de forma analítica. Existen expresiones cerradas para todas las métricas. Para la grafos dispersos se puede comprobar que tanto C (aproximadamente 0.75) como L (aproximadamente $|V|/\langle k \rangle$) son grandes.

II.13.3. Mundo pequeño

Son grafos que presentan altos valores de C (aprox .8) y bajos valores de L (aprox $\ln(|V|)$). Se obtienen introduciendo un pequeño número de “atajos” en un grafo regular. Representan bien un gran número de redes tales como redes sociales.

Las redes biológicas son todas de mundo pequeño.

Un grafo de mundo pequeño es aquel cuyo índice de clusterización es el mismo de un grafo regular con el mismo número de nodos y ramas, y cuyo camino característico es el mismo de un grafo aleatorio con el mismo número de nodos y ramas.

II.13.4. Grafos libres de escala

Son grafos que presentan bajos valores de C (aprox 0) y bajos valores de L (aprox $\ln(|V|)$). Se obtienen mediante crecimiento de la red y enlace preferencial. Cuando la distribución de los nodos se dibuja en escala log-log aparece una línea recta. Representan bien un gran número de redes tales como internet o redes de reacciones químicas.

Para diferenciar estas redes de un grafo aleatorio en la escala log-log, ya que los grafos libre de escala son una recta mientras que los aleatorios siguen una distribución de Poisson.

	L	C
Anillo Regular	125.438	0.643
Mundo Pequeño	14.2	0.626
Libre de escala	3.409	0.0186
Aleatorio	3.89	0.004

$|V|=2000 \quad k=8$

Un grafo con 2000 vecinos en el que cada nodo tiene 8 vecinos, ¿cuántas ramas tiene el grafo? 8000. De cada nodo entran y salen 8 ramas. $8 \cdot 2000/2$ porque cada rama se cuenta dos veces, una en cada vecino.

II.14. Algoritmos sobre grafos

El algoritmo de **búsqueda en anchura** permite calcular un camino mínimo entre dos nodos de un grafo. Dijkstra es una versión del algoritmo anterior para grafos

ponderados. Ambos algoritmos funcionan tanto en grafos dirigidos como no dirigidos. Los algoritmos nos permiten calcular las métricas sobre el grafo.

Si los grafos tienen bucles y ramas paralelas, el algoritmo no funciona.

En el algoritmo de búsqueda en anchura, se marcan los nodos vecinos. Símil con los corredores de la antigua Grecia: las noticias se transmitían por los corredores. Había en una ciudad tantos corredores como ciudades vecinas que tenía una ciudad. Si Atenas está siendo invadida por los turcos, no se necesita ningún corredor que vaya a Atenas. Los corredores de Atenas van a los vecinos, marcándolos como visitados y añadiéndolos en la cola. Una vez terminado, se repite con los vecinos de los vecinos de Atenas, así hasta haber completado el grafo.

El algoritmo de Búsqueda en profundidad permite calcular puntos de articulación de un grafo. El algoritmo de Ford-Fulkerson permite calcular cortes mínimos.

II.15. Cálculos sobre grafos: NetworkX

Existen multitud de herramientas para facilitar el cálculo de las diferentes métricas sobre grafos. Algunas herramientas también permiten la visualización de grafos de orden y/o tamaño reducido. Las herramientas pueden ser de tipo interactivo o librerías que se pueden emplear sobre lenguajes de programación.

NetworkX es un paquete Python orientado al análisis de grafos. Permite la creación de digrafos, multigrafos y pseudografos. Además incorpora el cálculo de un elevado número de métricas y algoritmos sobre el grafo. Sin embargo, da un soporte muy reducido a la visualización de grafos mediante el paquete Matplotlib.

La parte práctica se encuentra en la carpeta ejercicios en el fichero nx.ipynb.

II.15.1. Creación de grafos

NetworkX permite la creación de grafos mediante tres métodos:

- **Creación de un grafo vacío al que posteriormente se añaden nodos y ramas:** se pueden usar varias funciones en networkx: Graph, DiGraph, MultiGraph, MultiDiGraph. Para añadir nodos a un grafo existente podemos usar las siguientes funciones del objeto graph: `add_node(n)` donde `n` puede ser cualquier objeto hashable (int, str, estructura, grafo) y `add_nodes_from(container)` donde `container` puede ser cualquier objeto contenedor (una lista, conjunto, fichero, nodos de otro grafo).

Pregunta: Si `H` y `G` son dos grafos creados previamente, ¿Cuál es la diferencia entre `H.AddNode(G)` y `H.AddNodesFrom(G)`? `H.AddNode(G)` agregaría `G` como un solo nodo al grafo `H`, considerándose como una entidad individual. Por el contrario, `H.AddNodesFrom` agrega todos los nodos de `G` al grafo `H` como nodos individuales, es decir, cada nodo de `G` se copia a `H`.

Para añadir ramas a un grafo existente se puede utilizar `add_edge(n1, n2)` y si uno de los dos vértices no existe, se añade al grafo, o `add_edges_from(container)` donde `container` puede ser una lista o colección de

ramas. Las propiedades `nodes` y `edges` del grafo nos devuelven respectivamente la lista de vértices y ramas del grafo.

A cada objeto del grafo (el propio grafo, los vértices o las ramas) se le puede asignar uno o varios **atributos**. Un atributo es un objeto de la forma `clave=valor`, la clave debe ser un objeto hashable. Los atributos se pueden asignar durante la creación del objeto o una vez creado.

- **Creación de un grafo con una topología predefinida:** NetworkX permite la creación de grafos con topologías predefinidas. Los grafos predefinidos están clasificados por categorías, entre otras grafos generales, mallas, grafos aleatorios, árboles, redes sociales, geométricos, comunidades, grafos pequeños, etc.

Para grafos generales, se puede crear un grafo completo (clic) con `n` vértices con `complete_graph(n)`. Para crear una cadena con `n` vértices se utiliza `path_graph(n)`. `cycle_graph(n)` es lo mismo que el anterior, pero en el cual el primer y último nodo es el mismo. En ambos casos habrá `n-1` ramas. `graph_atlas(n)` crea un grafo número `n` del libro *An Atlas of Graphs* (en el cual el grafo 1 está vacío, el grafo 2 tiene un nodo, el grafo 3 tiene dos nodos pero sin conectar, el grafo 4 tiene dos nodos conectados, etc).

Las mallas se pueden crear con las funciones `grid_2d_graph(m, n, periodic=false)`, `grid_graph` y `hypercube_graph(n)`.

Para grafos aleatorios, `gnp_random_graph(n, p)` se crea un grafo con `n` nodos donde cada una de las posibles ramas del grafo existe con probabilidad `p` y no estará con probabilidad `1-p`. De media, habrá $(p * n * n - 1) / 2$ ramas. La función `gnm_random_graph(n, m)` crea un grafo aleatorio con `n` nodos y `m` ramas. La topología generada es la misma que en la función `gnp`, pero es difícil de demostrar. `watts_strogatz_graph(n, k, p)` crea un grafo de mundo pequeño con `n` nodos, cada uno conectado a `k` vecinos, y `p` la probabilidad de reconexión de cada nodo. Por último, `barabase_albert_graph(n, m)` crea un grafo libre de escala con `n` nodos y `m` ramas.

Se puede crear un árbol aleatorio de `n` nodos con `random_tree(n)`. También está la función `prefix_tree(path)` para crear un árbol prefijo generado a partir del iterable de listas `path`.

- **Carga de un grafo desde un fichero externo**

II.15.2. Importación y exportación de grafos

NetworkX permite importar grafos y exportar grafos a un fichero con diferentes formatos, los más usuales son adjacency list, multiline adjacency list, edge list, GEFX y otros formatos como JSON, YAML, Pajek, etc.

En la lista de adyacencia, cada línea del fichero representa la lista de adyacencia de un nodo, el primer elemento es el identificador del nodo y a continuación los identificadores de sus vecinos. La lectura y escritura del fichero se realiza mediante las siguientes funciones:

- `read_adjlist(path[, comments, delimiter, ...])`: Lee el grafo indicado por el fichero `path`.

Pregunta
examen:
gnp
random
graph
vs gnm
random
graph,
ramas
en gnp
random
graph

- `write_adjlist(G, path[, comments, ...])`: Guarda el grafo G en el fichero indicado por path.
- `parse_adjlist(lines[, comments, delimiter, ...])`: Interpreta las líneas de un grafo representado mediante lista de adyacencia.
- `generate_adjlist(G[, delimiter])`: Genera una línea del grafo G en formato de lista de adyacencia.

La lista de adyacencia multilínea incluye cada vecino en una línea separada, siendo útil cuando los índices son cadenas. La lectura y escritura del fichero se realiza mediante las siguientes funciones:

- `read_multiline_adjlist(path[, comments, delimiter, ...])`: Lee el grafo indicado por el fichero path.
- `write_multiline_adjlist(G, path[, comments, ...])`: Guarda el grafo G en el fichero indicado por path.
- `parse_multiline_adjlist(lines[, comments, delimiter, ...])`: Interpreta las líneas de un grafo representado mediante lista de adyacencia multilínea.
- `generate_multiline_adjlist(G[, delimiter])`: Genera una línea del grafo G en formato de lista de adyacencia multilínea.

La lista de ramas contiene una línea para cada rama más sus posibles atributos. Una posible línea podría ser `a b {'weight':7, 'color':'green'}`. La lectura y escritura del fichero se realiza mediante las siguientes funciones:

- `read_edgelist(path[, comments, delimiter, ...])`: Lee el grafo indicado por el fichero path.
- `write_edgelist(G, path[, comments, ...])`: Guarda el grafo G en el fichero indicado por path.
- `read_weighted_edgelist(path[, comments, delimiter, ...])`: Lee el grafo ponderado indicado por el fichero path.
- `write_weighted_edgelist(G, path[, comments, ...])`: Guarda el grafo ponderado G en el fichero indicado por path.
- `parse_edgelist(lines[, comments, delimiter, ...])`: Interpreta las líneas de un grafo representado mediante lista de ramas.
- `generate_edgelist(G[, delimiter])`: Genera una línea del grafo G en formato de lista de ramas.

II.15.3. Información sobre el grafo

NetworkX incorpora una serie de funciones que permiten obtener información sobre el propio grafo, los nodos o las ramas. Las principales funciones de **información sobre el grafo** son:

- `degree(G[, nbunch, weight])`: Devuelve el grado de un nodo o de un grupo de nodos.
- `degree_histogram(G)`: Devuelve la distribución de grado del grafo.
- `density(G)`: Devuelve la densidad del grafo.
- `info(G[, n])`: Devuelve un conjunto de información sobre el grafo G o el nodo n.
- `is_directed(G)`: Devuelve True si el grafo es dirigido.

Las principales funciones de **información sobre los vértices** son:

- `nodes(G)` Devuelve un iterator sobre los nodos del grafo.
- `number_of_nodes(G)` Devuelve el orden del grafo.
- `all_neighbors(graph, node)` Devuelve todos los vecinos de un vértice.
- `non_neighbors(graph, node)` Devuelve los no-vecinos de un vértice.
- `common_neighbors(G, u, v)` Devuelve los vértices comunes a dos vértices.

Las principales funciones de **información sobre las ramas** son:

- `edges(G[, nbunch])` Devuelve las ramas del grafo entre los vertices indicados en nbunch (todas si no se indica nbunch).
- `number_of_edges(G)` Tamaño del grafo.
- `non_edges(graph)` Devuelve las ramas que no están en el grafo.

Las principales funciones de **gestión de atributos** son:

- `set_node_attributes(G, values[, name])` Establece los atributos de un vértice desde un valor o un diccionario de valores.
- `get_node_attributes(G, name)` Obtiene los atributos de un vértice
- `set_edge_attributes(G, values[, name])` Establece los atributos de una rama desde un valor o un diccionario de valores.
- `get_edge_attributes(G, name)` Obtiene los atributos de una rama

II.15.4. Visualización

NetworkX permite una visualización muy simple (pero a veces suficiente) del grafo. Cuando se desea una visualización más avanzada se suelen realizar los cálculos necesarios con NetworkX y luego el grafo se exporta a otras herramientas como Gephy o Cytoscape. La visualización siempre debe realizarse de grafos o subgrafos pequeños, los algoritmos de visualización son lentos y poco eficientes.

Las principales primitivas de visualización son:

- `draw(G[, pos, ax])`: Dibuja el grafo G con Matplotlib, no dibuja colores, ni etiquetas.
- `draw_networkx(G[, pos, arrows, with_labels])`: Dibuja el grafo G con Matplotlib, permite añadir etiquetas a los objetos
- `draw_networkx_nodes(G, pos[, nodelist, ...])`: Dibuja los nodos del grafo G en las posiciones indicadas por pos.
- `draw_networkx_edges(G, pos[, edgelist, ...])`: Dibuja las ramas del grafo G en las posiciones indicadas por pos.
- `draw_networkx_labels(G, pos[, labels, ...])`: Dibuja las etiquetas de los nodos del grafo G en las posiciones indicadas por pos.
- `draw_networkx_edge_labels(G, pos[, ...])`: Dibuja las etiquetas de las ramas del grafo G en las posiciones indicadas por pos.

NetworkX permite algunos **layouts** muy simples:

- `draw_circular(G, **kwargs)`: Layout circular.
- `draw_kamada_kawai(G, **kwargs)`: Layout dirigido por fuerzas Kamada-Kawai.
- `draw_random(G, **kwargs)`: Layout aleatorio.
- `draw_spectral(G, **kwargs)`: Layout espectral.
- `draw_spring(G, **kwargs)`: Layout de muelle.
- `draw_shell(G, **kwargs)`: Layout tipo concha.

También es posible recuperar la lista de posiciones de los nodos sin pintarlos, las rutinas devuelven un diccionario con las posiciones de cada nodo.

- `circular_layout(G, [scale, center, dim])`: Layout circular.
- `kamada_kawai_layout(G, [scale, center, dim])`: Layout dirigido por fuerzas Kamada-Kawai.
- `random_layout(G, [scale, center, dim])`: Layout aleatorio.
- `spectral_layout(G, [scale, center, dim])`: Layout espectral.

- `spring_layout (G, [, scale, center, dim])`: Layout de muelle.
- `shell_layout (G, [, scale, center, dim])`: Layout tipo concha.
- `rescale_layout(pos[, scale])` Devuelve el array de numpy pos reescalado a (-scale, scale) en todos los ejes.

II.15.5. Algoritmos

NetworkX incluye una cantidad enorme de algoritmos aplicables a grafos (actualmente unos 250 algoritmos). Los algoritmos se clasifican por categorías en función del problema que resuelven. NetworkX tiene algoritmos en 50 categorías diferentes. Cada una de estas 50 categorías está a su vez dividida en subcategorías. Las categorías más utilizadas en bioinformática son algoritmos para centralidad, cliques, clustering, conectividad, k-cores, operadores, caminos mínimos, árboles y algoritmos aproximados (problemas Np completo como el cálculo del betweenness).

II.15.5.1. Centralidad

La centralidad de grado es la fracción de los nodos con los que está conectado cada nodo. Están las funciones `degree_centrality`, `in_degree_centrality` y `out_degree_centrality`.

La centralidad de carga es similar al betweenness, pero usa un algoritmo diferente propuesto por Newman. Se puede obtener con `load_centrality` para nodos y `edge_load_centrality` para las ramas.

El closeness tiene la función `closeness_centrality`. El betweenness se puede calcular con varias funciones: `betweenness_centrality`, `edge_betweenness_centrality`, `betweenness_centrality_subset` y `edge_betweenness_centrality_subset`.

II.15.5.2. Cliques

Un cliqué es un subgrafo completo, es decir, un subgrafo en el que todos los nodos están unidos con todos. `enumerate_all_cliques(G)` devuelve todos los cliques de un grafo no dirigido. `find_cliques(G)` devuelve los cliques maximales para cada nodo del grafo. `make_max_clique_graph` devuelve el cliqué maximal del grafo, y `graph_clique_number` el orden del cliqué maximal del grafo.

II.15.5.3. Clustering

La función `triangles` calcula el número de triángulos del grafo. También está `transitivity`, que calcula la fracción de los triángulos que existen en G sobre el total del triángulos posibles. La función `clustering` calcula el índice de clusterización para un conjunto de nodos, y `average_clustering` el índice de clusterización del grafo.

II.15.5.4. Conectividad

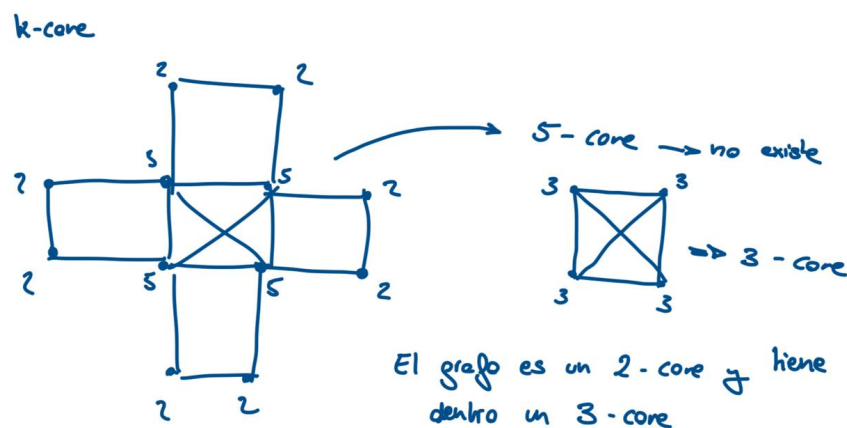
Para grafos no dirigidos, `is_connected` devuelve True si el grafo es conexo. `number_connected_components` devuelve el número de componentes conexas, y `connected_components` los componentes conexos del grafo. `node_connected_component` da el componente conexo al que pertenece un nodo.

Para grafos dirigidos, pueden tener una conectividad fuerte (cualquier nodo se puede alcanzar desde cualquier otro nodo) o débil (cualquier nodo se puede alcanzar desde cualquier otro nodo al duplicar cada rama del grafo). Esto se mide con `is_strongly_connected` y con `is_weakly_connected`. También se pueden ver los componentes conexos del grafo (`strongly_connected_components` y `weakly_connected_components`) y su número.

II.15.5.5. K-cores

Un k-core es un subgrafo maximal tal que el grado de todos sus nodos es k o mayor. El cálculo de k-cores se emplea para la creación de jerarquías de proteínas. `core_number` devuelve el número de core de cada nodo del grafo, es decir, el máximo k para todos los k cores que contienen al nodo. `k_core` devuelve el k-core de mayor k del grafo.

Pregunta
examen:
algo de
k-cores
entra
seguro



II.15.5.6. Operadores

`complement(G)` devuelve el grafo complemento de G, y `reverse(G[, copy])` el inverso del grafo dirigido G. `compose(G, H)` devuelve el grafo G compuesto con H, es decir, une los conjuntos de vértices y ramas de ambos grafos, los grafos no tienen por qué ser disjuntos. `union(G, H[, rename, name])` devuelve la unión de G y H, ambos grafos deben ser disjuntos. `intersection(G, H)` devuelve un grafo que contiene solo los nodos que están a la vez en G y H. `difference(G, H)` devuelve un grafo con las ramas que están en G pero no en H.

II.15.5.7. Caminos mínimos

Para grafos ponderados:

- `shortest_path(G[, source, target, weight])` calcula el camino más corto entre dos nodos dados.
- `all_shortest_paths(G, source, target[, weight])` calcula todos los caminos mínimos en el grafo entre `source` y `target`.
- `shortest_path_length(G[, source, target, weight])` calcula la longitud del camino mínimo entre dos nodos dados.
- `average_shortest_path_length(G[, weight])` calcula la el camino mínimo promedio del grafo(parámetro `L`).
- `has_path(G, source, target)` devuelve `True` si existe un camino entre `source` y `target`.
- `dijkstra_path(G, source, target[, weight])` calcula el camino ponderado más corto entre `source` y `target`.
- `dijkstra_path_length(G, source, target[, weight])` calcula la longitud del camino ponderado más corto entre `source` y `target`.
- `dijkstra_predecessor_and_distance(G, source)` calcula el camino ponderado más corto y los predecesores desde `source`.
- `single_source_dijkstra(G, source[, target, ...])` calcula el camino ponderado más corto y las distancias desde `source`.
- `single_source_dijkstra_path(G, source[, ...])` calcula el camino ponderado más corto desde `source`.
- `single_source_dijkstra_path_length(G, source)` calcula las distancias desde `source`.

Para grafos no ponderados:

- `single_source_shortest_path(G, source[, cutoff])` calcula el camino más corto entre `source` y el resto de nodos del grafo.
- `single_source_shortest_path_length(G, source)` calcula la longitud del camino más corto entre `source` y el resto de nodos alcanzables del grafo.
- `all_pairs_shortest_path(G[, cutoff])` calcula el camino más corto entre todas las parejas de nodos del grafo.
- `all_pairs_shortest_path_length(G[, cutoff])` calcula la longitud del camino más corto entre todas las parejas de nodos del grafo.

II.15.5.8. Algoritmos aproximados

Muchos de los algoritmos vistos en las anteriores secciones son terriblemente lentos y otros incluso no se implementan por su enorme lentitud. Para solucionar este problema se recurre a algoritmos aproximados:

- `max_clique(G)` encuentra el máximo cliqué
- `average_clustering(G[, trials])` estima el coeficiente de clustering C .
- `k_components(G, min_density=0.95)` una k -componente es una componente que necesita eliminar al menos k nodos para desconectarse.
- `all_pairs_node_connectivity(G, nbunch=None, cutoff=None)` la conectividad entre dos nodos es el mínimo número de nodos que es necesario remover para desconectarlos.
- `astar_path(G, source, target[, heuristic, ...])` devuelve el camino más corto entre `source` y `target` según el algoritmo A^*

Capítulo III

Grafos aleatorios, grafos regulares y redes de mundo pequeño

III.1. Grafos aleatorios

III.1.1. Modelo de Erdős y Rényi

En las redes reales muy raramente suele aparecer una topología aleatoria. Sin embargo los grafos aleatorios han sido muy estudiados por las siguientes razones. Si encontramos una propiedad que ocurre con probabilidad 1 para ciertos parámetros de la red, podemos saber si nuestra red tiene esa propiedad solo con mirar sus parámetros. Para predecir el comportamiento de ciertas propiedades en función de parámetros de la red. Para comprobar si nuestra red tiene sesgos estructurales.

El modelo de Erdős y Rényi se estudió en los años 50-60. Cada rama del grafo existe con una probabilidad p , que suele seguir una distribución uniforme.

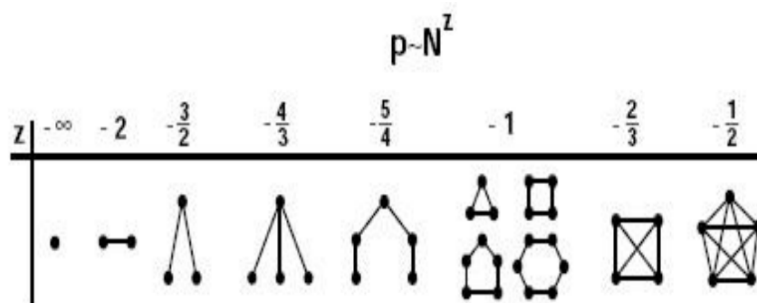
Otra forma de definir los grafos aleatorios es seleccionar parejas de nodos aleatoriamente. Se seleccionan exactamente $pN(N-1)/2$ parejas. Ambos tipos de grafos aleatorios son equivalentes.

III.1.2. Propiedades

El estudio de grafos aleatorios se centra sobre todo en averiguar para que probabilidad p aparece cierta propiedad Q :

- Cuando el grafo es conexo.
- Cuando la distancia media es menor que cierto número.
- Cuando el índice de clusterización es mayor que cierto número.

Erdős y Rényi descubrieron que las propiedades Q aparecían de forma repentina según crecía p . Para muchas propiedades Q se verifica que existe una **probabilidad crítica** $p_c(N)$ tal que:



- Con probabilidad 0 el grafo no tiene Q si $p(N) < p_c(N)$
- Con probabilidad 1 el grafo tiene Q si $p(N) > p_c(N)$

III.1.3. Subgrafos

La primera propiedad que estudiaron fue la aparición de subgrafos. Por ejemplo, a que probabilidad crítica p casi todo grafo G contiene un árbol de orden 3. La probabilidad crítica $p_c(N)$ de encontrar algunos subgrafos se ve en la imagen:

Ejercicio: Calcular la probabilidad crítica p para que una red aleatoria de 1000 nodos contenga tanto un ciclo de orden 5 como un cliqué de orden 5 (cuidado, tiene truco). El ciclo de orden 5 aparece con una Z de -1. Un cliqué de 5 aparece con una Z de -0.5. Para que haya un cliqué de orden 5 es obligatorio que haya un ciclo de orden 5. Por tanto, teniendo $N = 1000$ nodos, solo habría que calcular la probabilidad de un cliqué: $1000^{-1/2} = 0.031$.

Pregunta
examen
antigua

III.1.4. Clusters

Un subgrafo aislado y conexo es un cluster. Erdős y Rényi demostraron que la estructura de clusters de un grafo cambia abruptamente cuando el grado medio (número de ramas dividido entre el número de nodos) se acerca a 1.

$$\langle k \rangle = \frac{\frac{N \cdot (N-1)}{2} \cdot p}{N} = \frac{p \cdot (N-1)}{2} \xrightarrow{N \rightarrow \infty} p \frac{N}{2}$$

Si el grado medio $\langle k \rangle$ está entre 0 y 1, casi todos los clusters son árboles (en su mayor parte) o clusters que contienen un solo ciclo. El número de clusters es de orden $N-n$ (número de nodos menos número de ramas), y el cluster mayor es un árbol de tamaño proporcional a N .

Si el grado medio $\langle k \rangle$ es mayor que 1, la estructura anterior cambia completamente. Aparece un cluster gigante con $[1-f(\langle k \rangle)]N$ nodos donde f es una función que decae exponencialmente de 1 a 0 cuando x va a infinito. Los demás clusters pertenecen a árboles con $Nf(\langle k \rangle)$ nodos.

III.1.5. Distribución de grado

El grado de cada nodo sigue una distribución binomial. La distribución del grado de los nodos sigue una distribución de Poisson.

III.1.6. Conexidad y diámetro

El diámetro de un grafo es la máxima distancia entre cualquier par de nodos. Si p no es demasiado pequeño los grafos aleatorios tienden a tener poco diámetro. Casi todos los grafos aleatorios tienen el mismo diámetro (más o menos) para la mayor parte de los valores de p .

En general se tiene:

- Si $\langle k \rangle < 1$: el grafo tiene árboles aislados.
- Si $\langle k \rangle > 1$: aparece el cluster gigante y el diámetro del grafo es el del cluster gigante. Para $\langle k \rangle > 3.5$, el diámetro es proporcional a $\ln(N)/\ln(\langle k \rangle)$.
- Si $\langle k \rangle \geq \ln(N)$: el grafo es conexo y su diámetro es próximo a $\ln(N)/\ln(\langle k \rangle)$

El camino característico se comporta de forma similar al diámetro. En particular:

$$\ell_{rand} \sim \frac{\ln(N)}{\ln(\langle k \rangle)}$$

III.1.7. Índice de clusterización

En un grafo aleatorio la probabilidad de que dos vecinos de un nodo dado están conectados es igual a la que dos nodos elegidos al azar estén conectados. El índice de clusterización de un grafo aleatorio es p , la probabilidad de existencia de cada rama.

Pregunta
típica de
examen

III.2. Grafos regulares

Son los mejor conocidos de forma analítica debido a sus simetrías. Existen expresiones cerradas para todas las métricas. Son los más usados en modelos de redes neuronales artificiales. También se usan como sustrato inicial para la generación de otros tipos de redes.

Un caso particular de grafos regulares son las mallas (grids) donde cada nodo se conecta a sus $2^d \cdot k$ vecinos más próximos, donde d es la dimensión del grid.

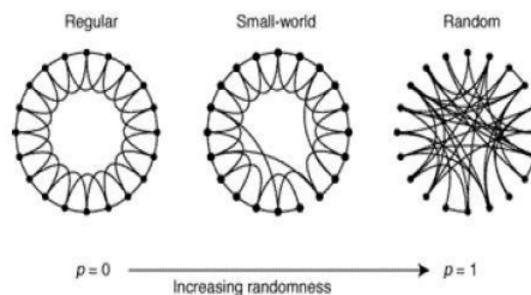
En las mallas monodimensionales, el índice de clusterización es de 0.75, el camino característico es $|V|/\langle k \rangle$, y la distribución del grado de los nodos es una delta en el valor $\langle k \rangle$. Es decir tanto L como C son más grandes de lo que correspondería en el grafo aleatorio equivalente y la distribución de grado se puede ver como el límite cuando la varianza tiende a 0.

III.3. Redes de mundo pequeño

Buena
pregunta
de 1
punto

En 1998 Watts y Strogatz (Nature, 1998) proponen un modelo de red dependiente de un parámetro p , que es distinto al parámetro p de las redes aleatorias. En redes aleatorias, significa la existencia de las ramas, pero aquí es la probabilidad de reasignación o recolocación de cada una de las ramas que ya tiene el grafo. Este modelo interpola entre un grafo regular y un grafo aleatorio. Se colocan inicialmente los nodos en un anillo y cada nodo se conecta con los $2k$ vecinos a izquierda y derecha.

Para cada rama de este grafo, con probabilidad p se decide si la rama se modifica o no. Si la rama se modifica, se elige un nuevo nodo al azar con probabilidad uniforme. Se evitan ramas dobles y autoconexiones. Este proceso produce pNk atajos en el grafo.



El método de Watts y Strogatz puede producir grafos no conexos. Un método alternativo propuesto por Newman consiste en añadir de forma aleatoria un número pequeño de ramas sobre el sustrato inicial (duplicación y manteniendo la rama original).

III.3.1. Sustrato inicial

Como sustrato inicial se suele tomar un grid monodimensional cumpliendo las siguientes condiciones:

$$N \gg k \gg \log(N)$$

Cada nodo esta conectado con sus $2k$ vecinos a izquierda y derecha.

Así, el grafo siempre será conexo ($k \gg \log(N)$) y disperso ($N \gg k$).

III.3.2. Sustratos bi-conexos

Un grafo conexo tiene un camino que une cualquier par de nodos. Una red biconexa es aquella que para cualquier par de nodos tiene al menos dos caminos que los unen. Por ejemplo, un anillo es una red biconexa porque se puede llegar por la izquierda y por la derecha. Las redes biológicas suelen ser biconexas, pero no todas lo son. Calcular tres o más caminos es un problema NP completo - buscar uno es fácil mediante un algoritmo en anchura y buscar dos se puede hacer con el algoritmo de Tarjan.

El modelo de red original de Watts y Strogatz tiene el mismo número de componentes biconexas que los grafos regulares. Las redes reales suelen tener un número mayor de componentes biconexas (p. ej. redes de comunicaciones o muchas redes biológicas).

Los grafos de mundo pequeño tienen el mismo índice de clusterización que un grafo aleatorio regular con el mismo número de ramas y nodos. Camino característico de un grafo aleatorio con el mismo número de nodos y ramas.

El modelo anterior permite crear redes de tipo Mundo-pequeño con un número elevado de componentes bi-conexas. El sustrato inicial es un grafo regular pero con un número elevado de componentes biconexas. Presenta una transición a aleatorio similar a la de los anillos.

III.3.3. Sustratos dirigidos y ponderados

El modelo original de Watts y Strogatz no contempla la posibilidad de grafos con dirección y peso. Existen redes en la naturaleza donde aparecen la direccionalidad y el peso (redes neuronales, redes sociales, redes de interacción, etc).

III.3.4. Parámetros

El procedimiento de Watts y Strogatz sobre el sustrato inicial produce una transición en el comportamiento del camino característico y del índice de clusterización. Ambos han de pasar de valores altos propios del grafo regular a valores pequeños propios del grafo aleatorio.

En una malla monodimensional $L = O(N)$ y $C = O(1)$. En un grafo aleatorio: $L = O(\log(N))$ y $C = 0 = p$.

Watts y Strogatz observaron que la transición en el comportamiento de los valores era diferente para el caso de camino característico L y para el índice de clusterización C . El camino característico presentaba la transición de régimen mucho antes que el índice de clusterización.

Cuando se consideran otros sustratos tales como grafos bi-conexos o grafos con dirección peso también se puede observar el mismo fenómeno en el comportamiento de C y L . Además las propiedades iniciales del grafo (bi-conectividad, dirección, peso) no se pierden durante la transición

L no empieza a decrecer hasta que $p > 1/Nk$ (es decir hasta que no aparece al menos un atajo). Por tanto, el valor de p para el cual se entra en la zona de mundo pequeño es dependiente de N y k . Para una probabilidad fija p , existe un valor N' tal que $L = O(N)$ si $N < N'$ y $L = O(\log(N))$ si $N > N'$. Se puede demostrar el valor de p que para un valor fijo de N y k produce la transición de L es $p = 1/kN$.

La distribución del grado de los nodos debe pasar de una delta (vale 1 en 0 y 0 en todos los demás) en el grado medio de cada nodo ($2k$) a una distribución de Poisson.

La distribución espectral (de la matriz de autovalores) también sufre una transición en función de p .

Capítulo IV

Grafos libres de escala, ataques a redes y aplicaciones

IV.1. Grafos libres de escala

En los grafos libres de escala, no hay longitud característica. Algunos parámetros siguen una distribución por ley de potencias, y existe un cutoff.

Una red de mundo pequeño es aquella con un índice de clusterización de la red regular y un camino característico como una red aleatoria.

Una red libre de escala tiene una distribución libre de escala del grado de los nodos. Así, pintarlo en escala log-log, sale una recta.

Estas redes son muy comunes en el mundo real: internet, redes sociales y redes biológicas. En estas redes existen hubs, y son robustas frente a ataques aleatorios, aunque frágiles frente a ataques calculados (si se puede elegir, elegiríamos el hub y nos lo cargamos entero).

IV.1.1. Modelo de Barabasi y Albert

Hasta ahora, todos los métodos de red son estáticas, la construcción de la red no cambia. Las redes libres de escala son dinámicas: crecen conforme pasa el tiempo y tienen una conexión preferencial, es decir, los nodos nuevos se unen a aquellos que estén muy conectados. La topología es el subproducto de la dinámica de la red.

Crecimiento: Los grafos aleatorios y de mundo pequeño parten de número fijo de nodos. Los grafos libres de escala parten de un número pequeño de nodos y se van añadiendo a la red.

Conexión preferencial: La probabilidad de conexión en grafos aleatorios y de mundo pequeño es independiente de grado del nodo. En mundo real, nodos muy conectados tienden a atraer a los nuevos nodos: la probabilidad depende del grado.

Algoritmo: Se parte de un número pequeño de nodos (m_0). En cada paso de tiempo, se añade un nodo con m ramas. La probabilidad de conexión a un nodo va

según su grado. La probabilidad de conexión es:

$$\Pi(k_i) = \frac{k_i}{\sum_i k_i}$$

Tanto el crecimiento como la conexión preferencial son características necesarias para generar una red libre de escala. Si se elimina alguna de las dos, la red resultante no se parece a una red libre de escala. Si se elimina la conexión preferencial, sale una distribución exponencial negativa, que no es una libre de escala. Si se elimina el crecimiento, sale una ley no estacionaria; hay un momento en el que parece una libre de escala, pero termina divergiendo a una Poisson.

Al igual que un grafo aleatorio, tiene un camino característico e índice de clusterización bajo. La diferencia radica en la distribución de grado: una red aleatoria tiene una de Poisson, y la libre de escala ley de potencias.

¿Una red puede ser libre de escala y de mundo pequeño a la vez? Sí, nada lo impide, mientras cumpla con las tres condiciones, puede ser las dos cosas a la vez. Para ser libre de escala debe tener una distribución recta si se pinta en escala log-log, un camino característico como un grafo aleatorio y un índice de clusterización como un grafo regular, ambos con el mismo número de nodos y ramas. Las redes biológicas suelen ser ambos.

IV.2. Ataques a redes

Un ataque es un conjunto de objetos de la red (nodos y/o ramas) que son deshabilitados o eliminados de la red. El objetivo de un ataque dado (consciente) es producir el máximo daño posible en términos de conectividad de la red.

La conectividad (resistencia) de un grafo después de un ataque se puede definir de diferentes maneras. Número de nodos que se desconectan de un nodo fuente determinado (server networks). Orden de la componente conexa con mayor número de nodos. La eficiencia (daño) de un algoritmo de ataque para un grafo dado es la inversa de la resistencia del grafo ante dicho ataque.

Para modelizar una red atacable se necesita un conjunto de nodos, un conjunto de ramas, una función de coste y una función de importancia. La función de coste indica cuánto le cuesta (en términos económicos, computacionales, etc) eliminar el elemento a un posible enemigo. La función de importancia indica la importancia del nodo entendida en cómo es de malo que otros nodos de la red queden desconectados de este nodo.

El coste de un ataque dado es la suma de los costes de los elementos eliminados del grafo. La importancia de un conjunto de nodos es la suma de las importancias de los nodos del conjunto. Puede que la suma no sea real, pero es un supuesto con el que se trabaja como generalización.

Para un grafo dado, definimos el core (núcleo) del grafo como la componente conexa con mayor importancia (no necesariamente el más grande). Definimos el daño producido en una red como la suma de las importancias de los elementos que no

pertenecen al núcleo del grafo después del ataque. La resistencia de la red a un ataque se define como la importancia del núcleo.

IV.2.1. Ataques óptimos

El problema de un ataque óptimo se puede definir ahora en los siguientes términos: Problema OPT_ATTACK: Dada una red sin servidor CN y dos valores constantes C y D ¿ existe un ataque A tal que $C(A) \leq C$ y $D(A) \geq D$? Esto es un problema NP-Completo.

Buscamos ataques que producen el máximo daño posible al mínimo con el mínimo coste posible. El problema de encontrar si tal ataque existe es NP-Completo incluso en el caso más simple de grafos no dirigidos, ramas no eliminables, coste idéntico en todos los nodos e idéntica importancia en todos los nodos.

IV.2.2. NP-Compleitud

Los problemas NP-Completo son problemas que verifican dos condiciones:

- Es muy costoso calcular una solución.
- Si nos presentan una posible solución, es fácil comprobar si esa solución verifica las condiciones del problema.

Esto implica que los problemas NP-Completo son adecuados para soluciones aproximadas.

IV.2.3. Ataques mediante algoritmos aproximados

Existen distintos métodos de ataque con algoritmos aproximados.

El **método browniano** es como un random walk en el grafo. En cada instante de tiempo se permite moverse a uno u otro vecino. Esto se repite durante muchas etapas. Si el grafo tiene sesgo estructural (una topología no aleatoria; small world o libre de escala), no se pasará por los elementos el mismo número de veces. Se ataca aquel nodo por el que más veces se haya pasado.

El **método de los cortes mínimos** busca hacer el máximo daño con el menor coste. Esto tiene el problema de encontrar el corte mínimo, ya que es un problema NP-Completo. Por tanto, no se hace el corte mínimo del grafo, sino entre dos nodos del grafo. Esto se puede hacer con un algoritmo de corte mínimo y flujo máximo. El corte mínimo del grafo siempre es menor o igual que el corte mínimo entre dos nodos, pero es muy difícil de encontrar. Por ello se suelen coger los nodos del diámetro para calcularlo.

Los fallos aleatorios eliminan a cada determinado tiempo un elemento del grafo, nodo o rama. Este ataque tiene una complejidad del orden del número de nodos ($O(n)$).

El método de grado máximo elimina aquel nodo que tenga el grado más alto, funcionando muy bien en los grafos libres de escala. El betweenness detecta elementos

importantes en la red, y es lo que se utiliza aquí. Esto tiene una complejidad computacional de $O(n^2)$.

NetworkX no tiene funciones para ataques, por lo que tocaría picar código.

IV.2.4. Topologías

Un ataque dado tiene diferentes eficiencias en diferentes tipos de grafos. Esto significa que antes de seleccionar un algoritmo de ataque usualmente es una buena idea saber que tipo de grafo vamos a atacar y seleccionar la mejor estrategia de ataque para ese tipo de grafo.

En redes biológicas, el ataque que más se utiliza es el método de cortes mínimos. El segundo es el de k-cores.

Las redes de mundo pequeño se deben atacar mediante caminos mínimos. Un grafo regular aguanta un poco menos que las de mundo pequeño los ataques de caminos mínimos. Un grafo libre de escala aguanta en su totalidad el ataque aleatorio, pero son igualmente sensibles a ataques de caminos mínimos y de grado máximo. Un grafo aleatorio aguanta bastante bien los ataques al no tener sesgo estructural, pero es algo más sensible al ataque por caminos mínimos. Una red jerárquica también se derrumba con un ataque por caminos mínimos.

IV.3. Aplicaciones

IV.3.1. Click - algoritmo de clustering

Click es un algoritmo de clustering aplicado al análisis de expresiones genéticas diseñado por Sharan y Shamir. Click también ha sido utilizado para clustering de conjuntos de datos de proteínas (ProtoMap). El algoritmo Click no hace ninguna suposición previa sobre la estructura, tamaño o número de clústers. El grafo (ponderado) es partido de forma recursiva usando cortes mínimos.

En Click, cada nodo puede ser:

- Una expresión genética representada mediante un número real que contiene n medidas del elemento (usualmente mediante un valor medio): expresiones de mRNA en diferentes condiciones o intensidades de hibridación de cDNA bajo diferentes oligos.
- Datos de similitud: un valor de similitud entre dos secuencias de proteínas.

Dos nodos se conectan si según un coeficiente de similitud que se obtiene a partir de los valores de los nodos.

El problema de clustering consiste en partir el conjunto V de vértices de un grafo en k conjuntos disjuntos de vértices tal que la unión de todos ellos es V . Dado un clustering, dos nodos del grafo son amigos si pertenecen al mismo clúster. Para comprobar la calidad del clustering se definen dos medidas: separación entre clústers y homogeneidad de cada clúster.

La dificultad está en ver si un conjunto de vértices son un clúster o no.

El cálculo de cortes mínimos en grafos ponderados es muy costoso. Para obtener el corte mínimo se aplica un algoritmo de corte mínimo s-t $O(|V||E|^{2/3})$ en el grafo no ponderado resultante de hacer todos los pesos 1. Los algoritmos de corte mínimo s-t consisten en buscar el corte mínimo que conecta dos nodos s y t dados. Los nodos s y t se eligen tal que su distancia coincida con el diámetro del grafo.

IV.3.2. ProtoMap

ProtoMap es un proyecto dedicado a la clasificación de secuencias de proteínas y jerarquización de familias de proteínas. Cada vértice es una secuencia y el peso de cada rama es un coeficiente de similitud entre las proteínas.

El particionado de la red se realiza de forma similar a Click, pero usando un criterio diferente para decidir de un grupo de nodos representa un clúster. Los clústers se obtienen buscando grupos de nodos altamente conectados entre sí. Los autores aplicaron el método a la base de datos SWISS-PROT.

IV.3.3. Redes de interacción

Cada nodo del grafo es una proteína. Una rama significa una interacción entre ambas proteínas. Las redes se construyen mediante los módulos de reconocimiento de péptidos y el uso de la técnica de phage-display. Dos proteínas se enlazan si ambas poseen ligandos preferidos por un determinado módulo de reconocimiento de péptidos.

Un k-core de un grafo G es un subgrafo G' tal que el grado de cada nodo de G' es al menos k. Un k-core puede contener k+1, k+2, ..., k+j cores. Este algoritmo produce una **jerarquía de subgrafos** basándose en el k de los k-cores obtenidos para cada posible k.

Para una red aleatoria similar en tamaño al dominio SH3 el tamaño medio del mayor kcore era k=4.01. Los autores suponen que este 6-core representa un simple compuesto. Los resultados obtenidos se intersectan con los datos de interacción obtenidos con otras técnicas (two-hybrid). Las interacciones más significativas predichas por el algoritmo se comprueban en vivo.