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.

Sandra Mingo Ramírez

UAM - 2024/25

8 de abril de 2025 19:59

Universidad Autónoma de Madrid
Bioinformática y Biología Computacional

Código en Github

Índice general

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

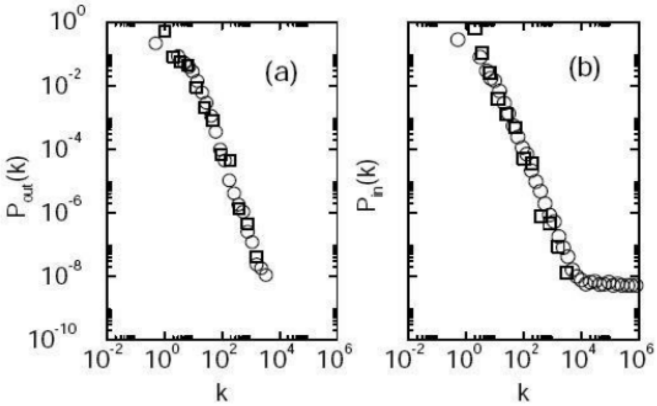
I.2.1. World Wide Web

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.

I.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.

I.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).

I.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 Erdös 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.

I.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.

I.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 linguística 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 jerárquita 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).

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 $V = \{v_1, v_2, \ldots v_n\}$ es el conjunto de vértices o nodos y $E = \{(v_i, v_j), (v_{i'}, v_{j'}), \ldots\}$ 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.

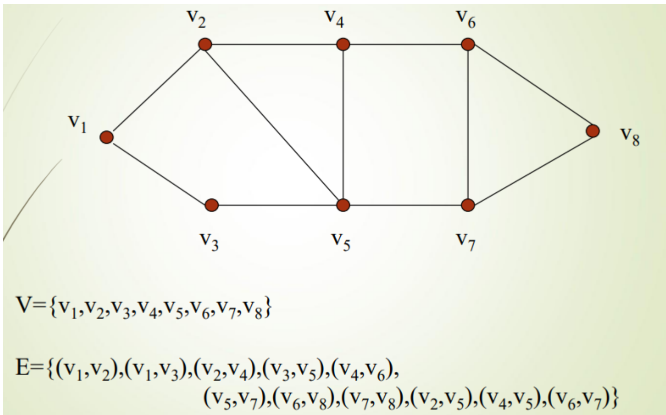
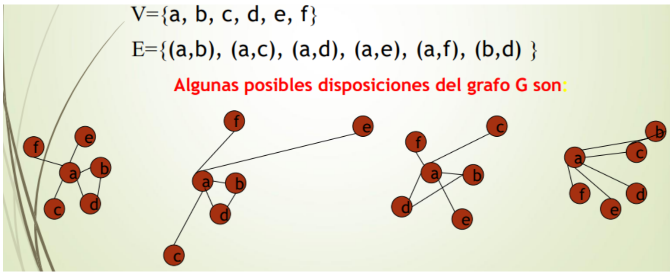


Figura II.1: *Ejemplo de grafo de orden 8 y tamaño 11.*

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 gráfo 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 $V = \{v_1, v_2, \ldots v_n\}$ es el conjunto de vértices o nodos y $E = \{(v_{i1}, v_{i2}, \ldots), (v_{i'1}, v_{i'2}, \ldots), \ldots\}$ 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

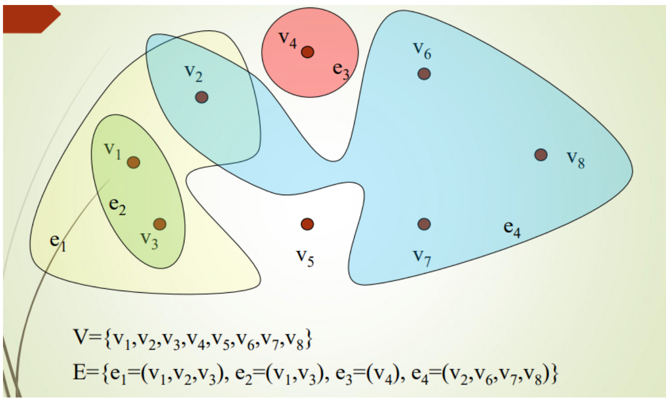


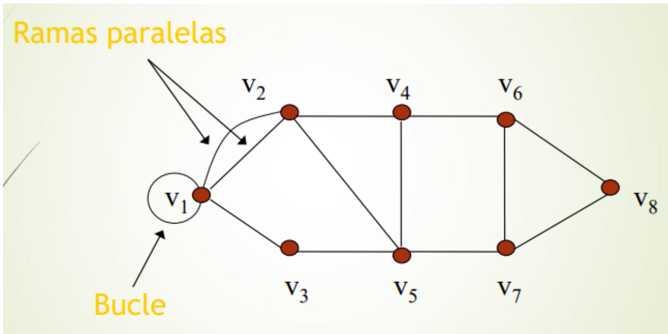
Figura II.2: *Ejemplo de hipergrafo de cardinalidad 4 y tamaño 32.*

Un hipergrafo H se dice que es **propio** si no es vacío ($V \neq \varnothing$) 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 (v5 no está en ninguna rama) y no es k-uniforme (e1 tiene 3 elementos, e2 tiene 2, e3 tiene 1 y e4 tiene 4).

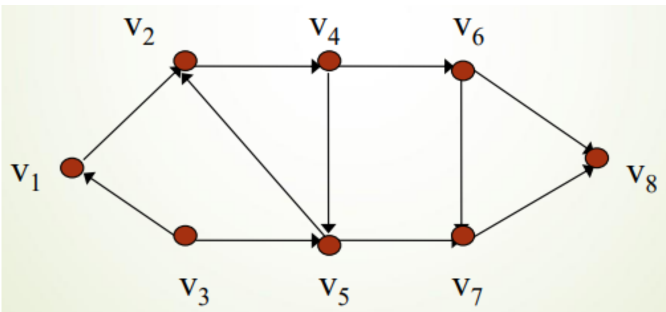
II.2. Bucles y ramas paralelas

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.

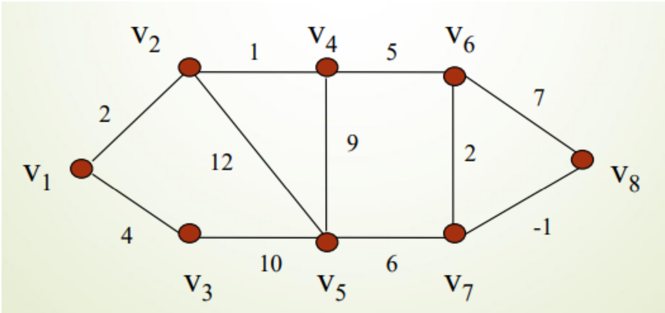


II.3. Grafos dirigidos y ponderados

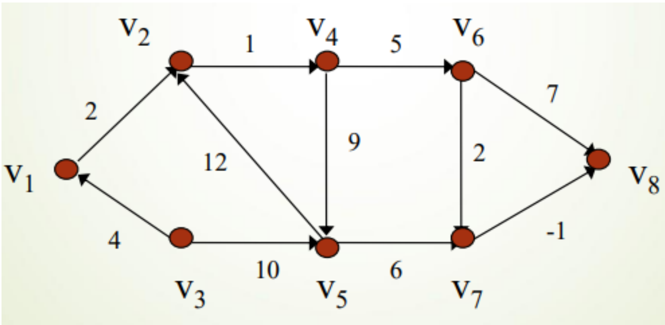
Se puede considerar que los enlaces entre nodos son dirigidos $(v_i, v_j) = (v_j, v_i)$. Los grafos dirigidos se denominan también **digrafos**.



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.

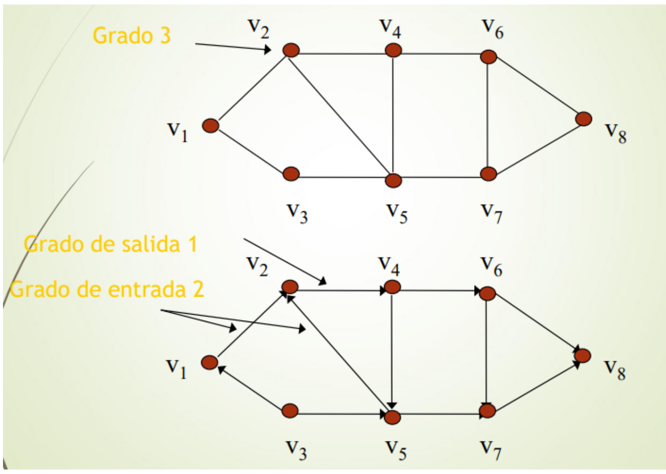


Los grafos dirigidos y ponderados poseen ramas dirigidas a las que se asocia un número.



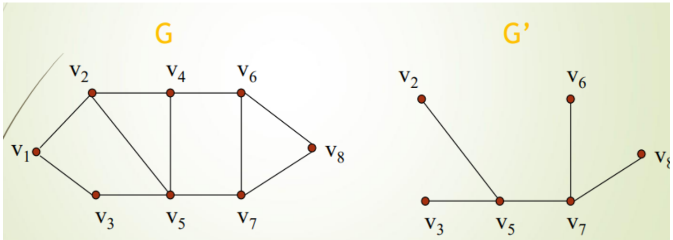
II.4. Grado de un nodo

Dos nodos de un grafo son **vecinos o adyacentes** si existe una rama que los conecta. El **grado** de un nodo es el número 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.



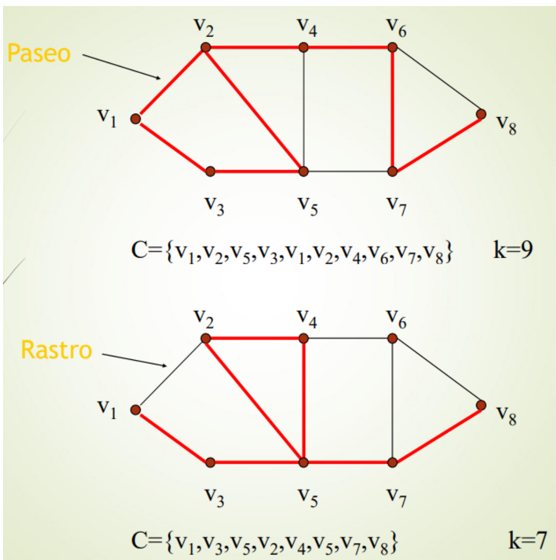
Un subgrafo G'=(V',E') de un grafo G=(V,E) se dice que es **abarcador** si V=V', es decir, si están todos los nodos, pero faltan algunas ramas.

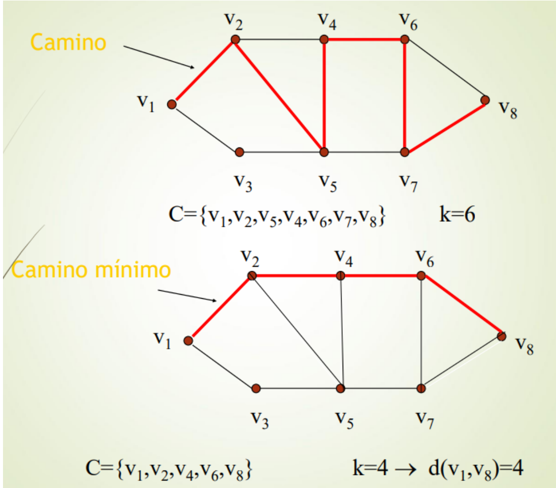
Un grafo es un subgrafo de sí mismo. Además, un grafo vacío es un subgrafo de cualquier grafo.

II.6. Paseos, caminos, circuitos y ciclos

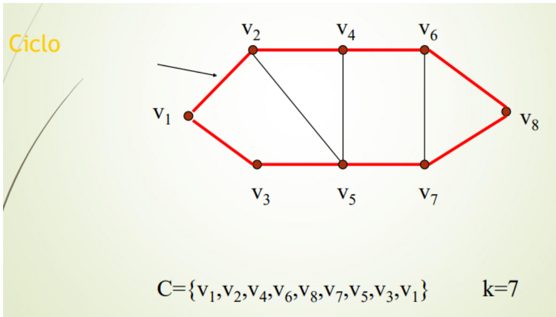
Un **paseo** de un nodo u a un nodo v es una secuencia de vértices $\{v_0, v_1, \dots, v_k\}$ con $v_1 = uv_k = v$ y (v_{i-1}, v_i) rama del grafo. El número de ramas del paseo es su **longitud**. Un paseo en el cual no se repiten ramas se denomina **rastro**. Un paseo en el cual todos los vértices $\{v_0, v_1, \dots, v_k\}$ 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 $\{v_0, v_1, \ldots, v_k\}$ 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, betweeness y closeness

Pregunta examen: Betweeness/Closeness/-Farness se define como...

Dado un nodo v_i se define su **betweeness** $C_B(v_i)$ 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 betweeness (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 $C_C(v_i) = 1/C_F(v_i)$.

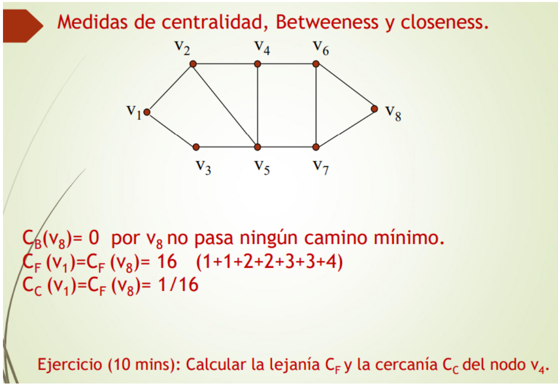


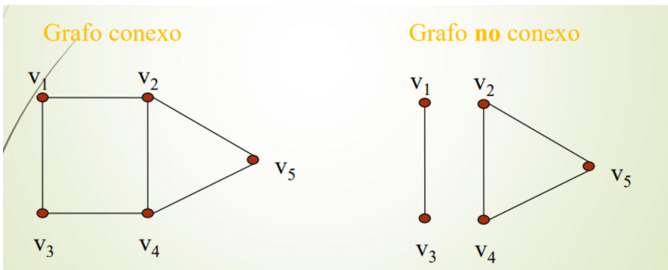
Figura II.3: *Respuesta al ejercicio: Cogiendo v4, la lejanía será 2+1+2+1+1+2+2 = 11, y la cercanía 1/11.*

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 característico

II.8. Conexidad

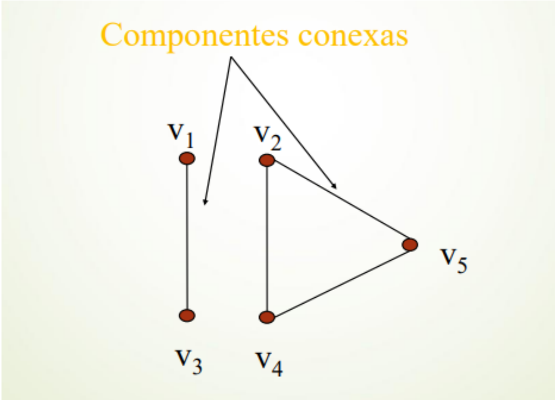
Un grafo es **conexo** si para cada par de nodos del grafo existe al menos un camino que los une. En otras palabras, que no esté separado en distintos trozos.



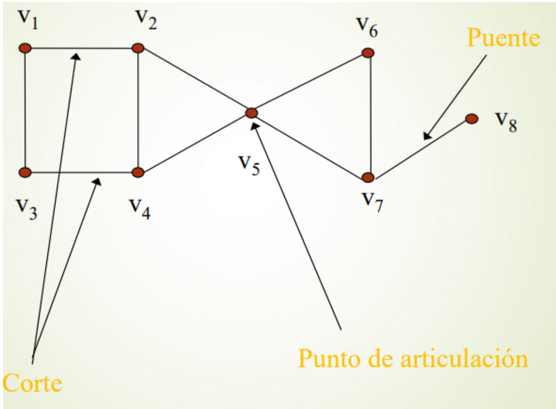
Hay un algoritmo muy rápido y eficiente que calcula si un grafo es conexo o no.

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 esta compuesto por una única rama, se denomina **puente**. 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.

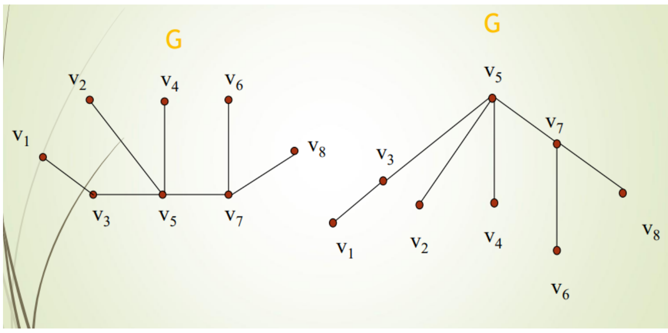


La máxima distancia entre cualquier par de nodos se denomina como diámetro.

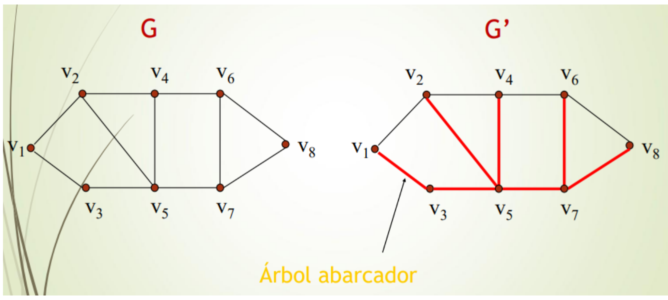
El corte mínimo entre dos nodos es siempre mayor que el corte mínimo de todo el grafo.

II.9. Bosques y árboles

Un grafo sin ciclos (acíclico) se denomina bosque. Un árbol es un grafo acíclico conexo. Cada componente conexa 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 V1 y V2.
- El vértice inicial de cada rama de E pertenece a V1 y el vértice final a V2

Ejercicio: el grafo anterior (el del árbol abarcador), ¿es bipartito? No, porque no es posible realizar la partición. Si V1 pertenece al conjunto 1, V2 y V3 deben estar en el conjunto 2, por lo que V4 y V5 tienen que estar en V1, 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 V4-V7, sí se podría generar un grafo bipartito: V4 y V7 en un conjunto y V5 y V6 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 (<k>): 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 (kv). Se calcula el coeficiente

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

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

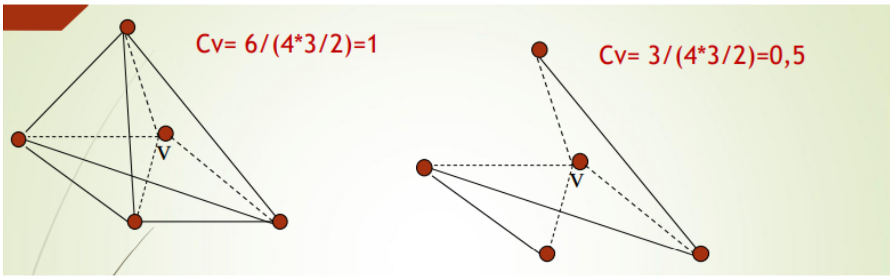


Figura II.4: *Figura de la izquierda: todos los vecinos del nodo v son vecinos entre sí. Figura de la derecha: la mitad de mis amigos son amigos entre sí (0,5).*

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.

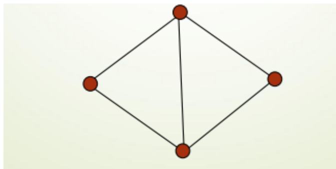


Figura II.5: *Se deben calcular dos índices de clusterización: nodo superior (igual al nodo inferior) y nodo izquierdo (igual al nodo derecho). Empezando por el nodo superior, la fórmula quedaría como* $2/(3*2/2) = 2/3 = 0,66$. *Para el nodo izquierdo, quedaría* $3/(3*2/2) = 1$. *Ahora hay que sumar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

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

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 (aproximadamemte 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|/<k>) 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 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 objejo 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 premite 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 (cliqué) 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)`.

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

  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.

- 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 multillinea 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 multilinea.
- generate_multiline_adjlist(G[, delimiter]): Genera una línea del grafo G en formato de lista de adyacencia multilinea.

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, cliqués, clustering, conectividad, k-cores, operadores, caminos mínimos, árboles y algoritmos aproximados (problemas Np completo como el cálculo del betweeness).

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. Cliqués

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 cliqués 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.