Proyecto Algoritmia

Conectividad y percolación

2020-2021 Q1

Lluís Mendoza Vandrell
Pablo Ortiz López

Índice

1. Introducción

La finalidad de este proyecto es el estudio experimental de la posible existencia de una o varias transiciones de fase del número de componentes conexos de un grafo sometido a un proceso de percolación.

En la teoría de grafos la percolación se refiere al proceso que ocurre en un grafo cuando le quitamos nodos o aristas. El análisis del proceso de percolación en grafos nos ayuda a simular el efecto de un daño progresivo a nodos o aristas individuales.

La percolación presenta un tipo de transición de fase, debido a que, en una fracción crítica de eliminación de aristas o nodos, el grafo se divide en componentes conexos significativamente más pequeños

El estudio de la percolación es un campo muy importante, ya que, por ejemplo, tiene aplicaciones en campos como la virología [1], la economía [2] o la ecología [3].

2. Representación del problema

Para poder representar el problema necesitábamos representar grafos, pero, si queríamos realizar una gran cantidad de experimentos, teníamos que utilizar una forma óptima para hacerlo.
Es por esto que descartamos la idea de utilizar matrices de adyacencia, ya que podíamos tener el riesgo de tener un coste cuadrático o cúbico en alguna de las operaciones.
También exploramos la opción de usar la estructura de datos *Union-Find* [4], ya que así podríamos consultar el número de componentes conexos de forma constante en cada iteración. Pero la tuvimos que descartar por los múltiples problemas que teníamos a la hora de la implementación.
Finalmente nos decantamos por el uso de listas de adyacencia para la representación del grafo, ya que nos permite una mayor rapidez y optimalidad en la ejecución.
Un detalle particular de nuestra solución es el uso de un *Set* que contiene todas las aristas para poder iterar de forma fácil cuando realicemos la percolación de aristas.

Se puede ver con mayor profundidad nuestra representación de un grafo en el archivo *Graph.h*

3. Métodos y algoritmos empleados

3.1. Percolación de nodos y aristas

En los algoritmos para la percolación de nodos o de aristas se precisa de un parámetro '*q*' que nos describe la probabilidad que tiene un nodo o una arista de ser eliminado del grafo.

Percolación de nodos:

- Se recorren todos los nodos del grafo y se comprueba que sea un nodo válido.
- Si el nodo es válido, generamos un número aleatorio módulo 1000 y, si este es más pequeño que el parámetro '*q*' multiplicado por 1000, entonces llamamos al método "treure_node" donde le pasamos por parámetro el nodo.

Dado un nodo n, el método "treure_node" recorre toda su lista de adyacencia para saber con qué nodo tiene una arista y, por cada uno de ellos, se elimina el nodo n de la lista de adyacencia. Finalmente, invalidamos el nodo n asignándole un valor de -1.

Percolación de arista:

- Recorremos todas las aristas y, como en la percolación de nodos, generamos un número aleatorio modulo 1000 y, si este es más pequeño que el parámetro '*q*' multiplicado por 1000, entonces llamamos al método "treure_aresta" donde le pasamos por parámetro la arista.

En el método "treure_aresta", dada una arista 'a', eliminamos 'a' del *Set* de aristas y eliminamos la relación de "arista" de las listas de adyacencia de los nodos que conforman 'a'.

3.2. Número de componentes conexos de un grafo

El algoritmo que implementamos para obtener el número de componentes conexos de un grafo es muy sencillo.

1. En primera instancia inicializamos el contador de componentes conexos a 0 y a todos los nodos como no visitados (empleando un vector de booleanos).
2. Para cada nodo 'v' del grafo
    a. Si el nodo 'v' no ha sido visitado, incrementamos el contador de componentes conexos y llamamos a la función DFS pasándole como parámetro el nodo v y el vector de visitados.

El DFS que utilizamos en el algoritmo anterior tiene una pequeña modificación, ya que solo visita los nodos que están marcados como no visitados y, cuándo visitamos a un nodo, lo marcamos como visitado.
Este algoritmo tiene un coste O(V+E)

En adelante, expresaremos el número de componentes conexos de un grafo como la razón entre las mismas y el número total de nodos. Por tanto, un grafo con N componentes conexas tendría una razón de 1, y un grafo conexo tendría 1/N.

4. Generador de grafos

4.1. Modelo C: grafo cuadrado NxN

Para poder crear el grafo para el modelo C, necesitamos crear un grafo cuadrado NxN. En teoría de grafos, un grafo cuadrado es un grafo no dirigido que puede ser dibujado en el plano de modo que cada superficie acotada es un cuadrilátero. Su implementación es muy sencilla:

- Dada una N, creamos NxN nodos y los inicializamos. O(n^2)
- Recorremos todos los nodos del grafo y, por cada nodo, se comprueba si puede poner una arista a su izquierda, derecha, encima o debajo y añade todas las aristas que cumplen esa condición. O(n^2)

Este algoritmo tiene un coste O(n^2)

4.2. Modelo D: un grafo geométrico aleatorio

Un grafo geométrico aleatorio [5] es un grafo no dirigido que se construye a partir de colocar un número de nodos de forma aleatoria en un espacio 2D y después conectar los nodos con las aristas si su distancia es menor que un radio r.

Nuestra implementación del generador de grafos geométricos aleatorios tiene un coste O(n^2), ya que debemos mirar que, por cada par de nodos de un grafo, se cumple la condición de tener una distancia entre ellos menor que un determinado radio r o no. La inicialización de los nodos tiene un coste O(n).

5. Experimentación

5.1. Modelo C

Para llevar a cabo la experimentación con el modelo C, disponemos de un fichero donde introducimos un número n de nodos, un número de iteraciones y un valor diciendo si queremos o no utilizar percolación de nodos o de aristas. Después, empleamos un bucle donde el valor 'q' va aumentando de 0 a 1 con incrementos de 0.05 en cada iteración y, por cada una de ellas, creamos un grafo NxN y realizamos una percolación de aristas o de nodos según se haya indicado. Finalmente normalizamos el resultado del número de componentes conexos de cada iteración, dividiéndolo en este caso entre NxN, ya que, al hacerlo, el número máximo de componentes conexos que pueden existir (NxN componentes en este caso) será igual a 1.

Realizamos los experimentos con una N=100, es decir, 10000 nodos y 100 iteraciones por cada 'q'. A su vez, ejecutamos otra prueba empleando una N=150 (22500 nodos) y 100 iteraciones.

Percolación de nodos

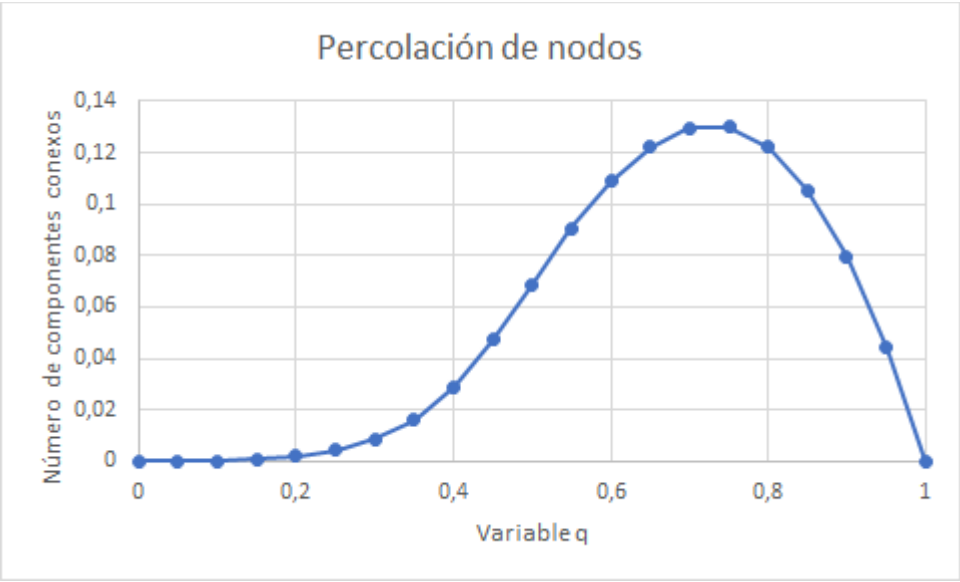

Figura 1: Percolación de nodos con N = 100

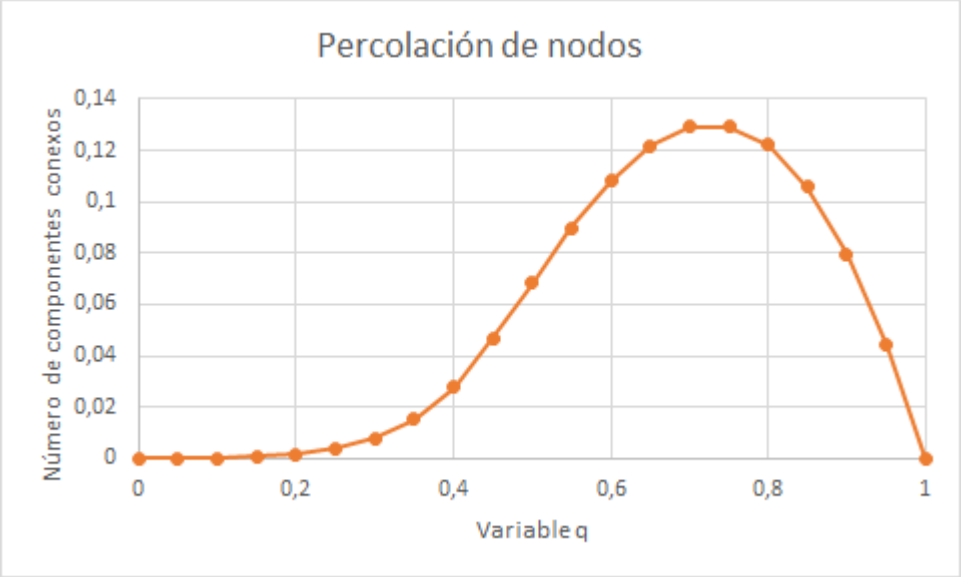

Figura 2: Percolación de nodos con N = 150

Percolación de aristas

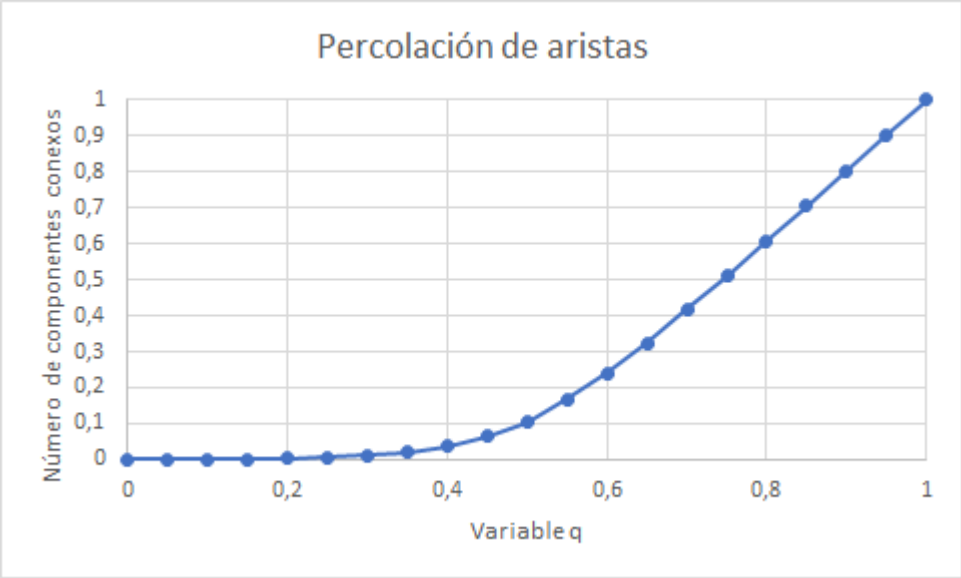

Figura 3: Percolación de aristas con N = 100

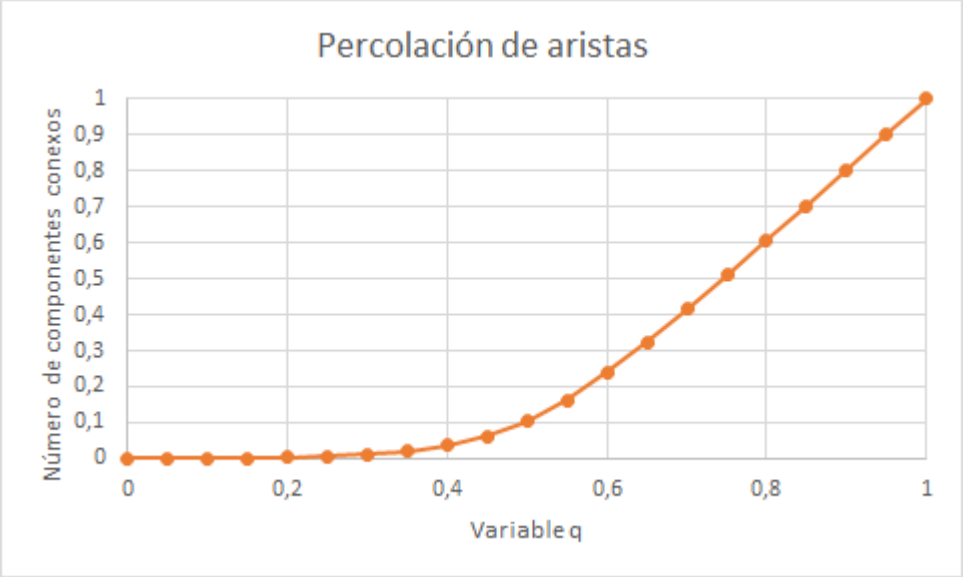

Figura 4: Percolación de aristas con N = 150

Consideramos que la N utilizada en las gráficas es adecuada, ya que al aumentar de N=100 a N=150, es decir, añadiendo 12500 nodos más, no hay un cambio significativo en la función.

Como podemos observar en las gráficas, no se observa ningún cambio de fase en la percolación de aristas o de nodos en los grafos cuadrados NxN.

5.2. Modelo D

En este modelo, se estudiará si hay una transición de fase, en el rango de q∈[0,1], para el número de componentes conexas de un *Grafo Geométrico Aleatorio*.

Este estudio busca encontrar el comportamiento para un número de nodos N asintótico, por lo que se observarán los resultados para varias N superpuestas.

Nuestra hipótesis inicial es, que, para un número infinito de nodos, la distancia entre ellos sería infinitamente pequeña y el valor de 'r' escogido sería indiferente, pues no habría nodos aislados. En este grafo, se tendrían que eliminar absolutamente todas las aristas para que no sea conexo, por lo que imaginamos que, para cualquier 'q' distinta de 1, el grafo tendría 1 componente conexa.

Haremos varios experimentos, para distintos valores de r, de forma que si observamos el comportamiento esperado, podamos corroborar que se mantiene en todo el rango de r∈[0,1].

Resultados

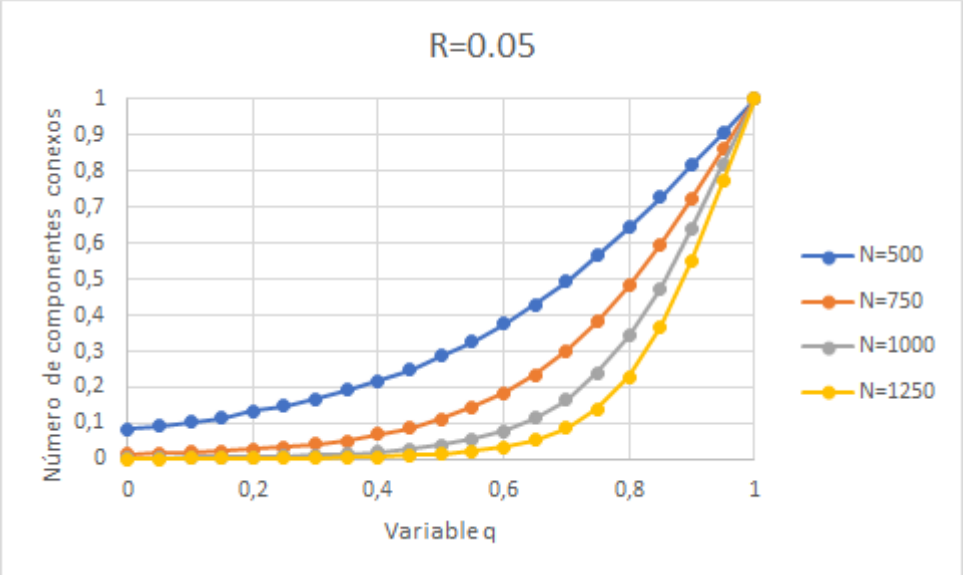

Figura 5: Percolación de aristas con R=0.05

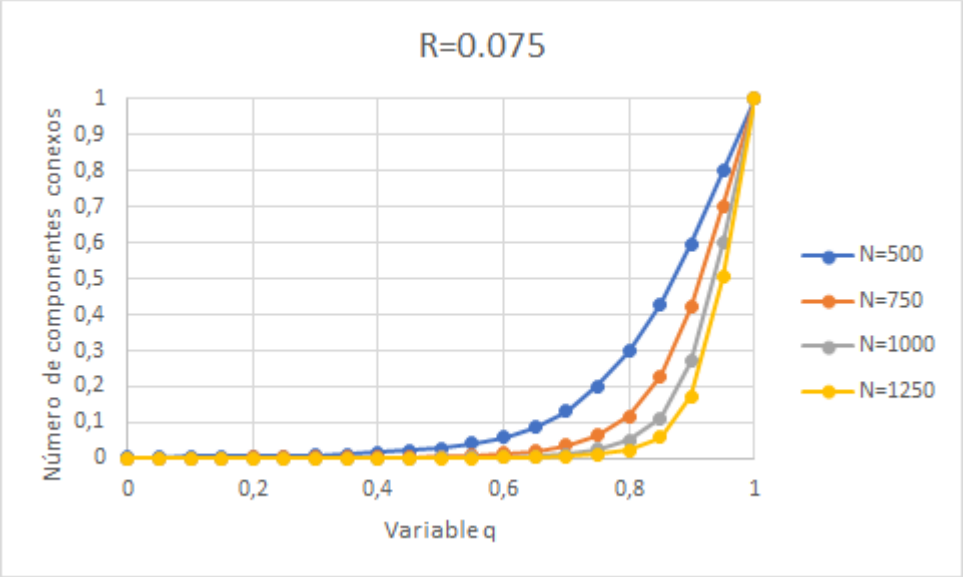

Figura 6: Percolación de aristas con R=0.075

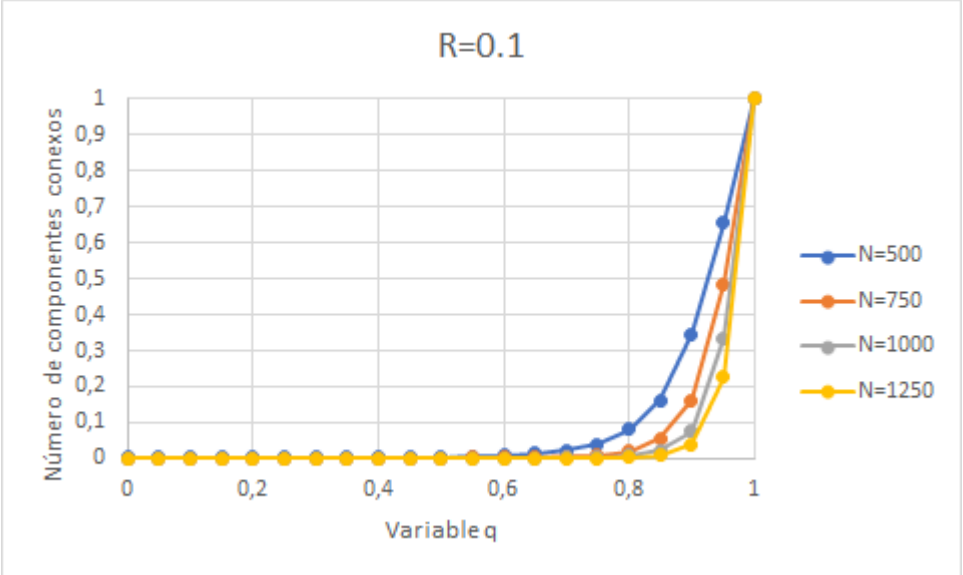

Figura 7: Percolación de aristas con R=0.1

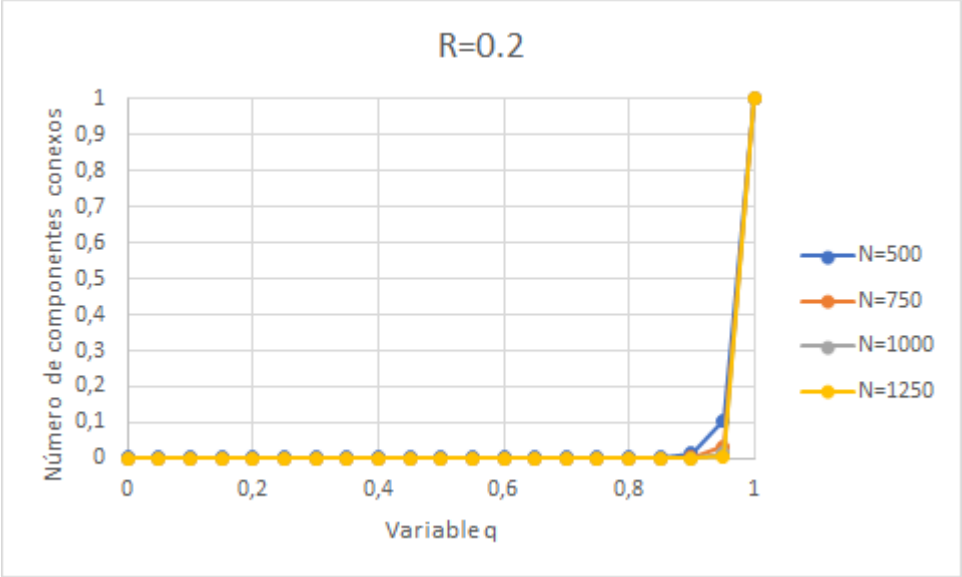

Figura 8: Percolación de aristas con R=0.2

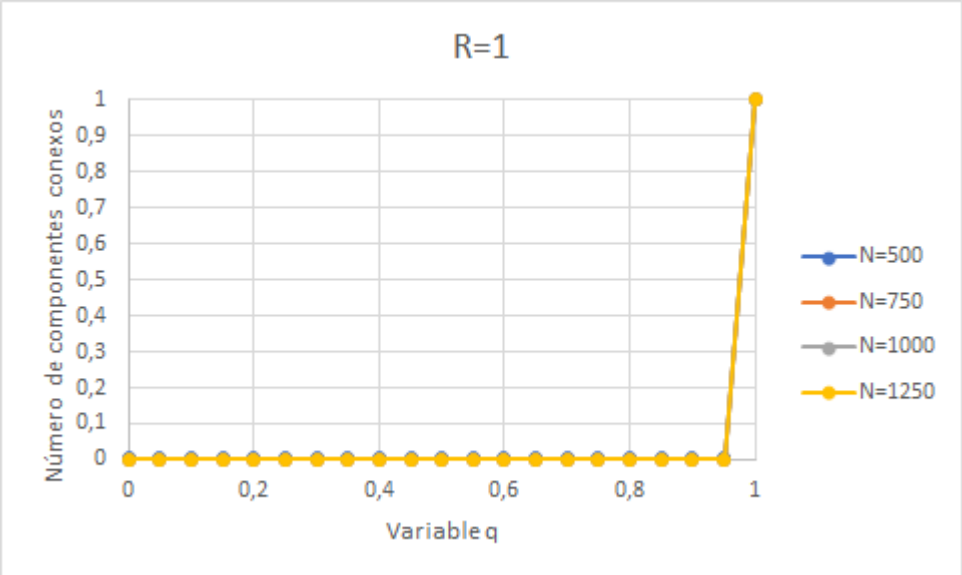

Figura 9: Percolación de aristas con R=1

Como podemos observar, a medida que aumentamos el número de nodos y el parámetro r, se va aproximando más a 1 el punto de inflexión en el cual se comienza a pasar de grafo conexo a N componentes conexas.
Vamos por tanto, a realizar más ejecuciones para q∈[0.9,1], y observaremos en detalle qué ocurre en esta franja del gráfico.

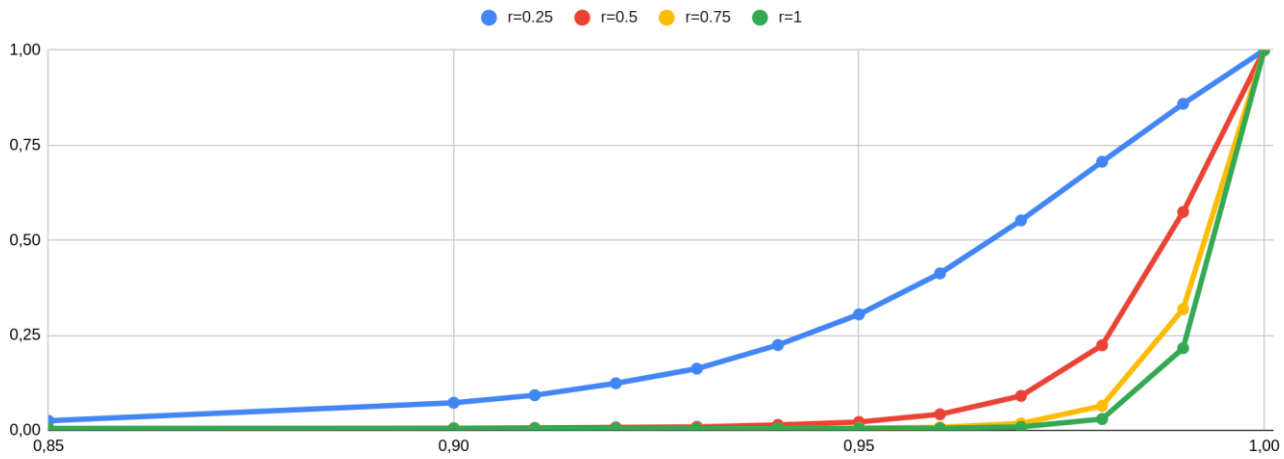

Figura 5: Percolación de aristas con R={0.25, 0.5, 0.75, 1}

Efectivamente, el comportamiento es claramente exponencial, para todas las r. Solo se obtendrán N componentes conexas para un valor de q = 1.

Análisis

Podemos observar como, para todos los valores probados a lo largo del rango r∈[0,1], se cumple nuestra hipótesis inicial: cuando la cantidad de nodos N tiende a infinito, el valor de q para el cual se pasa de grafo conexo a N componentes conexas tiende a 1.

Por tanto, **para valores asintóticos de N**, se podría decir que hay un cambio de fase en q=1, ya que para q<1, el número de componentes conexas será 1, mientras que para q = 1, este será N. Esto se cumple en todos los grafos geométricos aleatorios, ya que hemos comprobado que este comportamiento es cierto para todos los valores de r.

6. Conclusión

Mediante la realización de este proyecto, hemos comprobado la importancia de disponer tanto de estructuras de datos como de algoritmos eficientes para experimentaciones de esta índole. Cuanto más rápido se pueda hacer los experimentos, más se podrán realizar, aumentando por tanto la fiabilidad de los mismos y permitiendo encontrar con mayor exactitud los puntos de inflexión buscados.

También hemos profundizado en conceptos de conectividad de grafos, ya que en un principio pensábamos que, si empezábamos a quitar de forma aleatoria nodos o aristas, el grafo dejaría de ser conexo rápidamente, hecho que, como ya hemos comprobado anteriormente, no se cumple.

Finalmente hemos aprendido la importancia del campo de la percolación puesto que, a priori, pueda parecer un estudio con poca aplicabilidad en el día a día. Pero, tal y como hemos mencionado previamente, forma una parte importante en los estudios de virología, economía o ecología.

7. Referencias

[1] Brunk, N. E.; Lee, L. S.; Glazier, J. A.; Butske, W.; Zlotnick, A. (2018). "Molecular Jenga: the percolation phase transition (collapse) in virus capsids".

[2] Mira Kantemirova, Zaur Dzakoev, Zara Alikova, Sergei Chedgemov, Zarina Soskieva (2018) "Percolation approach to simulation of a sustainable network economy structure "

[3] Boswell, G. P.; Britton, N. F.; Franks, N. R. (1998). "Habitat fragmentation, percolation theory and the conservation of a keystone species".

[4] M. T. Goodrich and R. Tamassia, Wiley (2015) "Algorithm Design and Applications"

[5] Wikipedia "Random geometric graph"