Percolación en grafos conexos ponderados

Francisco Gerardo Meza-Fierro
Posgrado en Ingeniería de Sistemas
Universidad Autónoma de Nuevo León

***Resumen*—Se estudia el flujo sobre un grafo conexo ponderado a medida que cierto porcentaje de sus aristas son saturadas hasta el punto en que no percole; es decir, hasta que sea imposible encontrar un camino que genere un costo factible. Se procede el mismo experimento con diferentes tamaños de grafos y se comparan los resultados obtenidos.**

***Palabras Claves*—percolación, grafo conexo, nodo, arista**

I. Introducción y teoría

La percolación trata sobre un proceso que cierto fluido lleva a través de algún medio poroso; dos de los ejemplos más usados para explicar de manera más sencilla este proceso son: a) la filtración de agua a través de una roca o suelo y b) el agua en la cafetera que recorre los granos del café hasta llegar al vaso. Este es un proceso tan común que se puede encontrar y/o reproducir en muchos problemas que incluso podría encontrarse en el fluido que corre a través de un grafo conexo ponderado. Entiéndase por "grafo conexo" a un grafo donde para cualquier par de nodos existe al menos un conjunto de aristas dentro del grafo que los conectan; por lo tanto un "grafo conexo ponderado" es un grafo que cumple lo anterior y además sus aristas tienen un valor o peso establecido el cual puede denotar, por mencionar algunas, el flujo máximo o mínimo, costo o distancia que existe de viajar de un nodo a otro.

El proceso de percolación también podría encontrarse en un grafo conexo ponderado si éste tuviera al menos una arista saturada, es decir, si esa arista tiene un valor excesivamente grande o cero (dependiendo de la representación de dicha ponderación), esto con la intención de simular un medio poroso por el cual el fluido viajará. Y esto es precisamente lo que se hará en este trabajo:

se supondrá que se desea viajar de un depósito/ciudad ubicado en el nodo inicial de un grafo conexo ponderado al cliente/ciudad ubicado en el nodo final, donde la ponderación de las aristas representará el costo que existe en viajar de un nodo a otro conectados por una arista. Pero ¿qué sucederá cuando algunas aristas del grafo estén saturadas, es decir, cuando alguna carretera o camino se encuentre bloqueado o cerrado por algún motivo (reparación, tráfico, desastre natural, condiciones climáticas, huelgas o revueltas, etcétera)? Ese es el objetivo de este trabajo: ¿qué tantas carreteras como máximo pueden estar cerradas para que aún existan caminos que conecten al depósito con el cliente de manera que el costo del viaje sea aún razonable? O dicho de otra manera: ¿qué tantas aristas como máximo pueden ser saturadas para que el flujo aún pueda percolar?

No es tarea complicada encontrar, en un grafo, la ruta "más corta" que conecte el nodo inicial con el nodo destino. El algoritmo de Dijkstra es un algoritmo eficiente que encuentra el camino de costo menor del nodo inicial al nodo destino. Edsger Dijkstra fue quien lo descubrió, razón por la cual este algoritmo lleva su nombre. De manera general, para un grafo conexo y ponderado de n vértices, los pasos que este algoritmo sigue [1] son:

1. Seleccionar vértice de partida, es decir un origen.
2. Marcar el punto de partida como el punto de inicio.
3. Determinar los caminos especiales desde el nodo de partida, es decir, el de inicio.
4. Camino especial es aquel que solo puede trazarse a través de los nodos o vértices ya marcados.
5. Para cada vértice no marcado, se debe determinar si es mejor usar el camino especial antes calculado o si es mejor usar el nuevo camino especial que resulte al marcar este nuevo vértice.
6. Para seleccionar un nuevo vértice no marcado como referencia, deberá tomarse aquel cuyo camino especial para llegar a él es el mínimo
7. Cada camino mínimo corresponde a la suma de los pesos de las aristas que forman el camino para ir del vértice principal al resto de vértices, pasando únicamente por caminos especiales, es decir los vértices marcados.

II. Solución propuesta y resultados

Para la resolución de este problema, se trabajó en el lenguaje de programación `R`. Como apoyo visual, se generaron grafos como el mostrado en la figura 1.

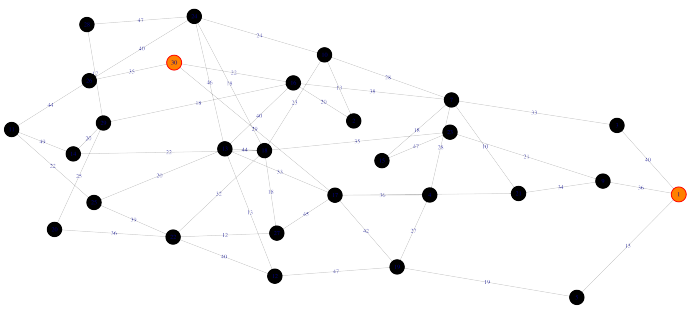

Figura 1. Grafo generado con treinta nodos, resaltando los nodos inicio y destino

Una vez generados, aplicar el algoritmo de Dijkstra para encontrar el camino de menor costo fue también una tarea sencilla ya que en `R` existe una función llamada `shortest.paths()` la cual lleva a cabo los pasos de dicho algoritmo, arrojando como resultado el costo que se obtendría de ir de un nodo a otro.

Para simular las carreteras bloqueadas que se mencionaron en el planteamiento del problema, se procedió de la siguiente manera:

se seleccionó de manera aleatoria el $5\,\%$ de las aristas totales y se cambiaron sus valores por un valor igual a infinito, se hizo uso de la función `shortest.paths()`, se guardó el resultado, se incrementó el porcentaje de aristas a diez y se repitió el experimento hasta haber saturado el $95\,\%$ de las aristas totales. Cada experimento se repitió quinientas veces para grafos de $50, 100, 150$ y 200 nodos obteniendo resultados como los mostrados en la figura 2.

Lo obtenido en esas gráficas muestran el porcentaje mínimo de carreteras que se necesitan bloquear para que la mitad de las corridas generadas dejen de percolar; por ejemplo: en el experimento hecho con el grafo de 50 nodos se obtuvo que a partir de que se satura el $66\,\%$ de las aristas la mitad de las corridas dejan de percolar, en el grafo de 100 nodos se necesita saturar el $79\,\%$ para el mismo efecto, en el grafo de 150 nodos se necesita saturar el $86\,\%$ y en el de 200 se necesita saturar el $91\,\%$.

III. Conclusiones

Resulta evidente que este porcentaje de percolación no es proporcional acorde al número de nodos de los grafos y esto se entiende debido a que el incremento en la cantidad de aristas es exponencial acorde al número nodos, lo que resulta que a mayor cantidad de nodos mucho mayor el número de aristas, por lo que el saturar aristas para encontrar ese porcentaje de percolación (mostrados en la figura 2) debe ser cada vez mayor conforme la cantidad de nodos de un grafo aumente, debido a la enorme cantidad de caminos posibles que se forman para llegar del nodo inicial al nodo destino.

Otro problema interesante derivado de esto resultaría en analizar de qué manera se incrementa tanto el costo como en el número de aristas en cada paso de la experimentación que se hizo en este trabajo y estudiar si estos incrementos siguen algún comportamiento o tendencia en relación a la cantidad de nodos que tenga el grafo.

Referencias

[1] O. D. Avilez Olea, D. Pernett Gonzalez, H. A. Perez Coronado, (2017). Algoritmo de Dijkstra.

[2] S.R. Broadbent, J.M. Hammersley, Proc. Cambridge Philos. Soc. 53 (1957) 629

[3] G. S. Torrubia, V. M. L. Terrazas. Algoritmo de Dijkstra . Un Tutorial Interactivo.

[4] C.F. Ramirez-Gutierrez, J.C. Mosquera-Mosquera, M.E. Rodríguez-García, (2014). Study of percolation and modeling of the orderedisorder transition for zincblendeediamond structures: Percolation and the existence of a unique band of events

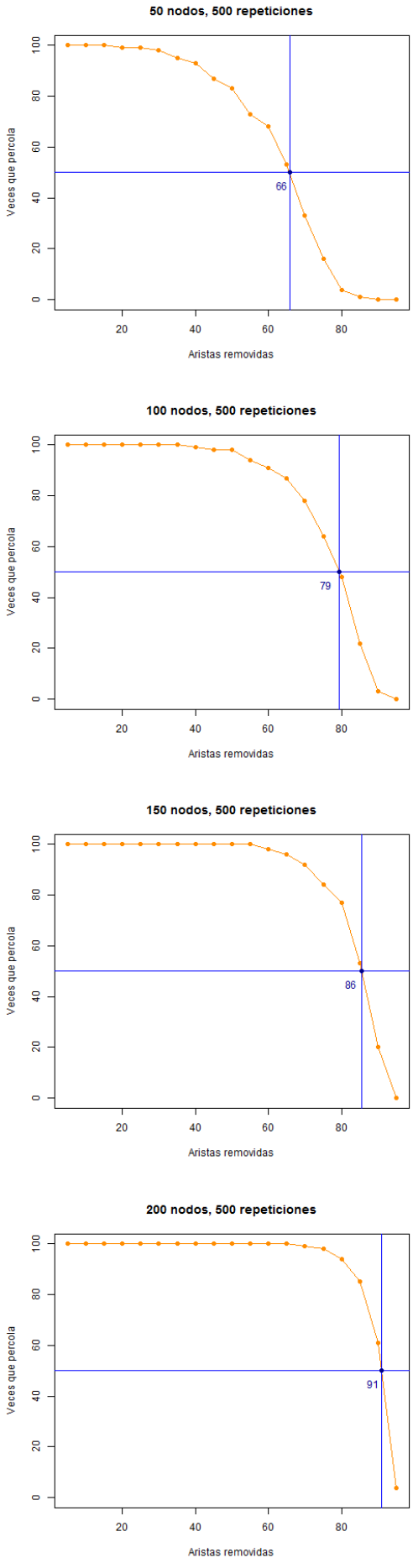

Figura 2. Grafos generados con sus porcentajes de percolación