PRÁCTICA 1 – Amazon + Amazonymás

ALGORITMIA PARA PROBLEMAS DIFÍCILES

Francisco Ferraz 737312 – Félix Bernal 738059

Resumen

En esta práctica se ha implementado el algoritmo de Karger-Stein para resolver el problema de mínimo corte sobre la división de productos en dos conjuntos para minimizar las pérdidas económicas de una empresa virtual teniendo en cuenta la relación entre los mismos.

Desarrollo

1. Generador aleatorio

En primera instancia, se implementó para la primera tarea un generador de datos de prueba aleatorios sobre la idea de que dos productos podían haber sido comprados juntos o no, sin tener en cuenta el número de veces que esto había sucedido. Esto se realiza desde el fichero *matrixGenerator.cpp*, donde se genera un número K de ficheros con matrices de dimension NxN de 1s y 0s para una serie de probabilidades dadas de que dos productos hayan sido comprados juntos, probando con distintas probabilidades (0.1, 0.2, 0.3...). El resultado de esto es una serie de matrices aleatorias generadas con una densidad definida. Las matrices son simétricas por representar la relación entre los productos (M[i][j] == M[j][i]). Además, se han generado también casos especiales como en los que un producto está conectado con todos los demás pero los demás no lo están.

2. HashMap

Se ha implementado desde cero una clase HashMap para poder guardar los datos de cada producto, basándose en la especificación de que los atributos de productos vienen dados por una línea de texto en un fichero. Basándonos en esto, se recomienda utilizar el tipo *string* del estandar de C++ a la hora de instanciar la clase hashmap para su uso con atributos de producto. Su implementación se encuentra en los ficheros *hashMap.hpp* y *hashMap.cpp*.

3. Análisis del problema

Para entender el problema como un problema de mínimo corte, tenemos que trasladar nuestro problema a grafos. En este caso, supondremos que los **productos** son los **vértices** del grafo y, si dos productos han sido comprados juntos (**valor 1** en la matriz) se establecerá una **arista** entre ellos.

El problema de mínimo corte establece que, para un grafo, se encuentra la separación de sus vértices en dos grupos de tal modo que se pierde el menor número de aristas. En nuestro caso, esto implica separar los productos en las dos divisiones de la empresa minimizando las pérdidas por productos relacionados.

4. Algoritmo de Karger-Stein

El algoritmo Karger-Stein es una mejora del algoritmo de Karger basada en la siguiente idea:

El algoritmo de Karger falla cuando contrae una arista equivocada (una que no se corta en el corte mínimo); la probabilidad de que esto ocurra aumenta cuanto más avanzado está el proceso del algoritmo. Basándose en esto, lo que se plantea es realizar el algoritmo hasta un punto determinado y, a partir de ese punto, repetir la ejecución varias veces intentando evitar

cortar aristas indeseadas. En concreto, se decide contraer hasta V/sqrt(2) vértices, llegados a este punto la probabilidad de no cortar una arista indeseada es del 50%; después, se contrae el resto de aristas por dos caminos distintos y se elige el mejor de los dos.

5. Generador aleatorio con valores

De cara al apartado opcional, se ha implementado también un generador de matrices de enteros aleatorias (con valores de 0 a 9, pero valdría cualquier otro caso) para poder probar el algoritmo de Karger-Stein con matrices con pesos. Está implementado en el fichero *matGenValue.cpp*.

6. Modificación de Karger con pesos

La única diferencia en la construcción de un grafo sin pesos y un grafo con pesos es aumentar el número de aristas entre los vértices. Es decir, si antes encontrar un valor 1 en la matriz nos hacía trazar una arista entre dos productos, ahora encontrar un valor 3 nos hará trazar 3 aristas entre estos dos productos. De este modo, el algoritmo resuelve automáticamente y sin modificar su comportamiento el problema, dado que elige las aristas de forma aleatoria, intentará contraer antes aquellas con mayor peso (aparecen más veces) y cortara con mayor probabilidad las de menor peso (aparecen menos veces). La única modificación que hemos implementado en el algoritmo ha sido el eliminar todas las ocurrencias de una arista al tratarla una vez.

7. Justificación temporal

Nuestro algoritmo, en el caso del Karger básico, está eligiendo aristas de forma aleatoria y colapsándolas, esto se realiza hasta que sólo quedan 2 vértices por colapsar: O(a) -> O(n^2)

Colapsar dos vértices se puede realizar con coste: O(n)

Así el coste de nuestro algoritmo de Karger básico tendría coste: O(n^3)

Por otro lado, el algoritmo de Karger-Stein, que encuentra el resultado con mayor probabilidad tendría coste O(2*(n/sqrt(2))^3) + O(n^3) -> O(n^3)

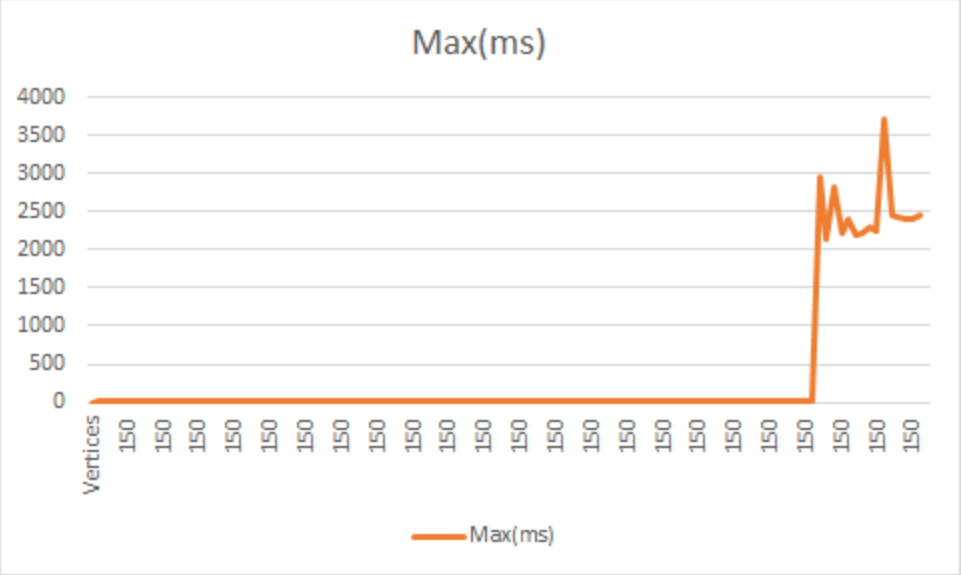
8. Banco de pruebas

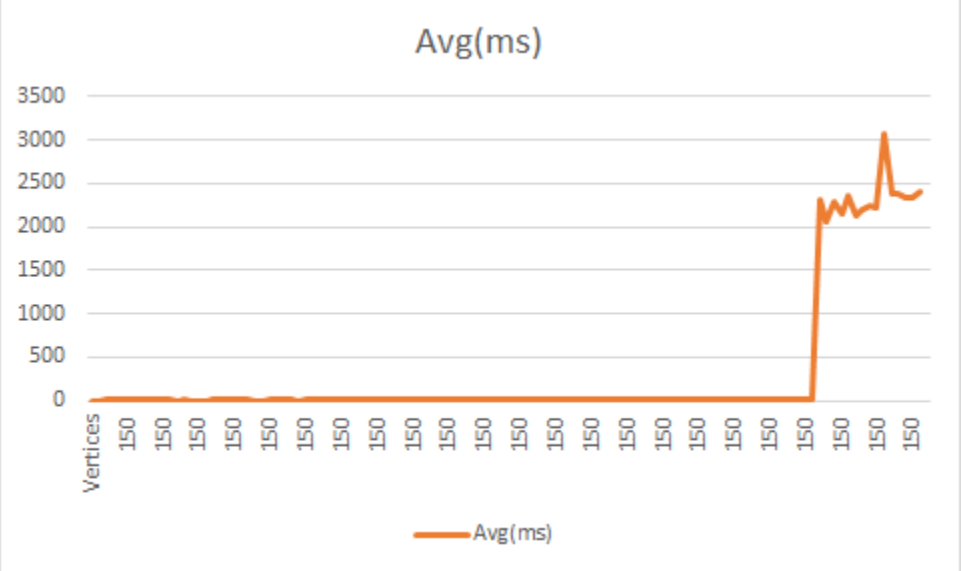
Se ha realizado un método exhaustivo de pruebas para el algoritmo utilizando como base los generadores aleatorios anterior desarrollados. Se ha recogido información sobre cómo avanza el coste en tiempo del algoritmo para distintas densidades de datos, que serán comparados a continuación.

En concreto, se ha realizado una serie de iteraciones sobre un total de 117 archivos con matrices 150x150 generados de forma aleatoria. Estos archivos se dividen en 2 grupos, matrices simples (valores 1 o 0) y matrices con peso (valores de 0 a 9). Para cada iteración se han calculado el tiempo medio de resolución del corte mínimo, el tiempo máximo y el mínimo corte encontrado.

TIEMPOS EN FUNCIÓN DEL NÚMERO DE VÉRTICES:

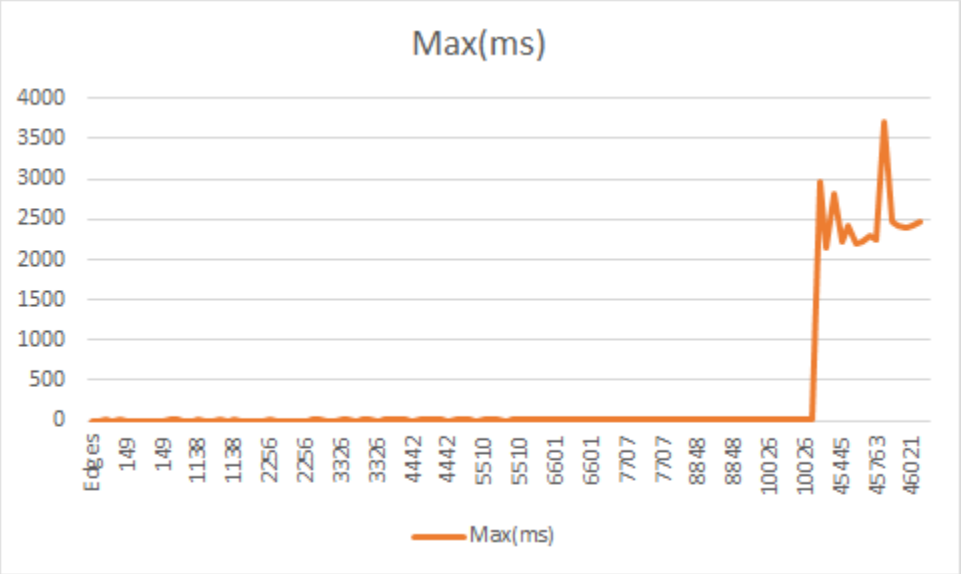
Con los datos recogidos, se puede apreciar cómo el crecimiento del tiempo no depende del número de vértices prácticamente, sino del número de aristas.

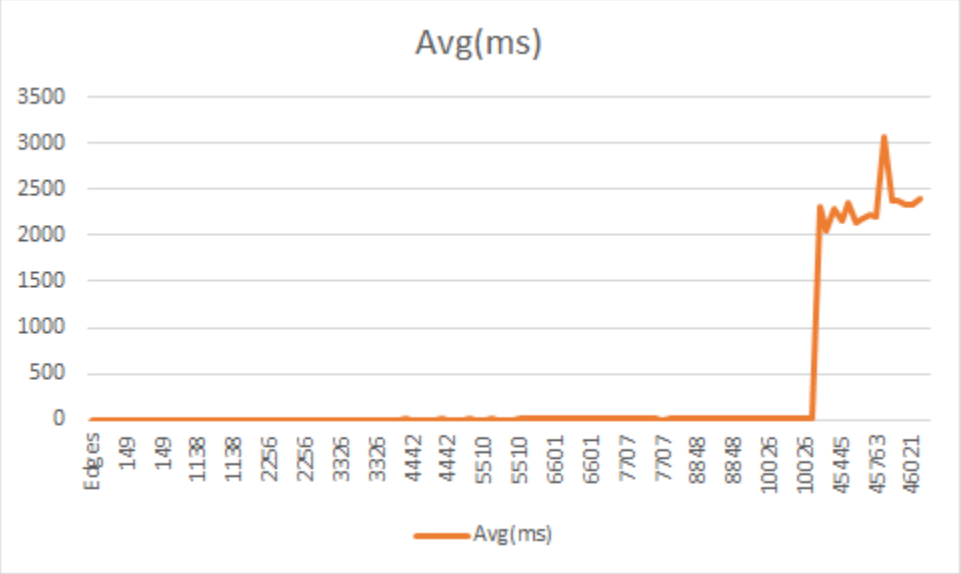




TIEMPOS EN FUNCIÓN DEL NÚMERO DE ARISTAS:

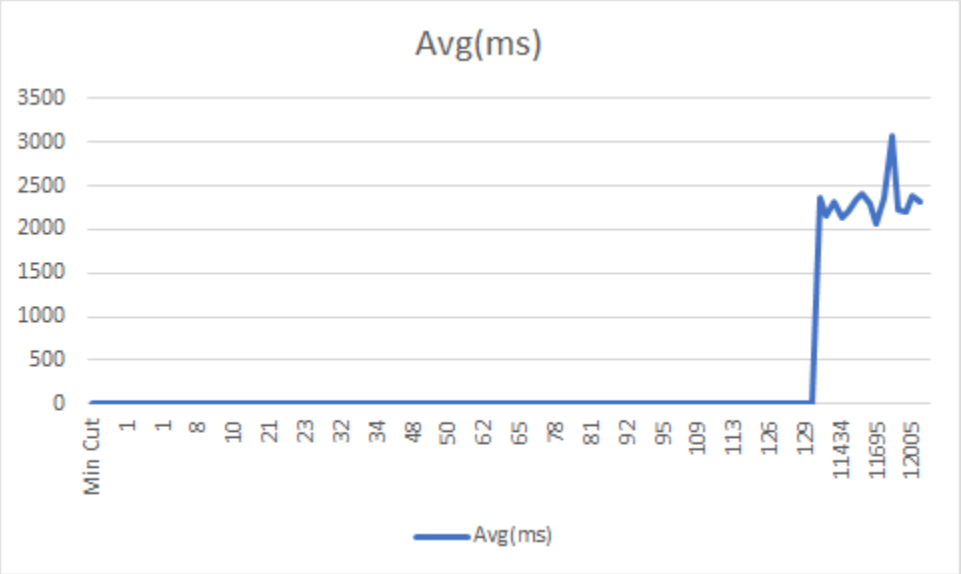
Como se puede apreciar a continuación, el tiempo crece de forma cuasi lineal en el número de aristas de nuestro problema. Como se ha explicado antes, se ve un salto sustancial en la gráfica en el momento en el que se pasa a analizar los ficheros del segundo grupo, pues aumentan en gran medida la cantidad de aristas. Cabe destacar que en todo el desarrollo se ven varios picos tanto para el tiempo máximo como el medio dada la cualidad probabilista del algoritmo.

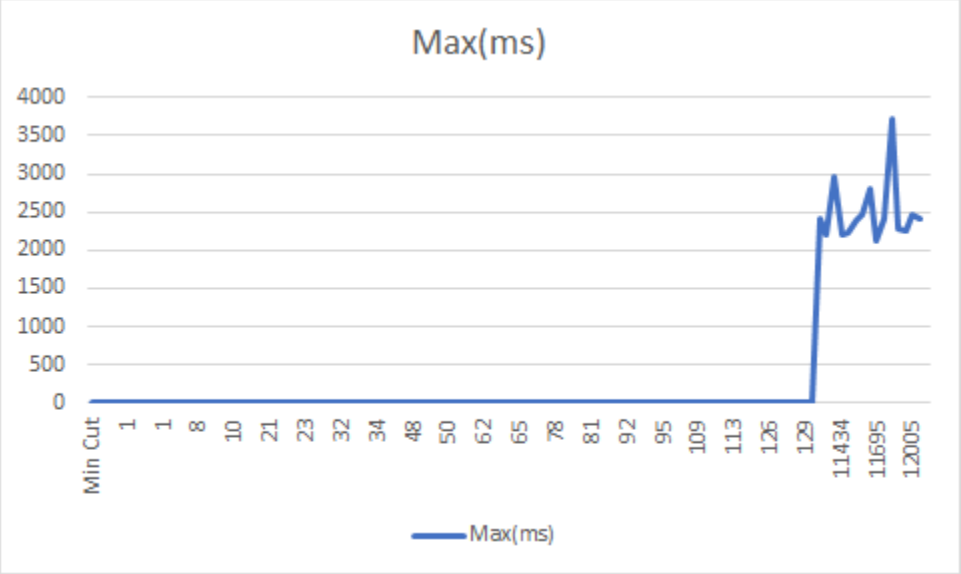




TIEMPO EN FUNCIÓN DEL CORTE MÍNIMO:

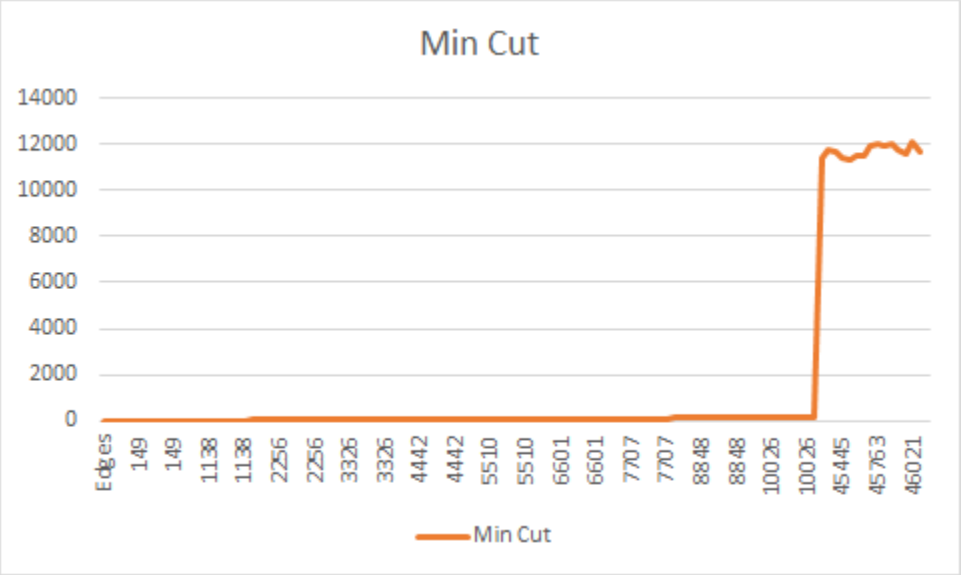
De manera incluso más evidente que antes, se encuentra un crecimiento prácticamente lineal en los costes en tiempo según el corte mínimo. Esto se debe a la capacidad de Karger-Stein para encontrar los óptimos sin gastar muchos recursos en todos los casos. Además, cabe destacar la mayor variabilidad de los máximos respecto de la media, que se debe al buen funcionamiento de la condición de Stein sobre el algoritmo de Karger básico.





CORTE MÍNIMO RESPECTO DEL TOTAL DE ARISTAS:

Finalmente, podemos apreciar como el corte mínimo crece de manera lineal pero ligeramente escalonada respecto del número de aristas.



Además, se han realizado otra serie de pruebas con sets de datos mayores y menores, encontrando siempre que el algoritmo es increíblemente eficiente hasta una "barrera" que hemos encontrado en matrices 1-0 de aproximadamente 300 elementos con una densidad del 90%, o con matrices más grandes (400, 500...) en densidades menores (20-30%).

El algoritmo deja de ser funcional a partir de los 550 elementos, donde no es capaz de resolver los casos más básicos en un tiempo razonable (menos de 1-2 minutos).