Edge Bandwidth

Luis Alberto Ballado Aradias
Cinvestav Unidad Tamaulipas
luis.ballado@cinvestav.mx

1 de mayo de 2023

I. Introducción

El cálculo del Edge Bandwidth es un problema importante en redes de comunicaciones para determinar la capacidad de transferencia de datos entre dos nodos en la red.

El max bandwidth se refiere a la cantidad máxima de datos que pueden transferirse por segundo a través de una conexión entre dos nodos de la red. El cálculo puede ser de gran utilidad en diferentes situaciones como identificar cuellos de botella en la red que puedan estar afectando el rendimiento, o para planificar la expansión de una infraestructura de res. Además, es de gran utilidad en aplicaciones de transmisión de video en tiempo real, la transferencia de archivos de gran tamaño, solo por mencionar algunas aplicaciones a este problema.

El cálculo del **edge-bandwidth** de un grafo es el mínimo entre todos los posibles costos de aristas de la máxima diferencia entre dos aritas adyacentes.

$$B_f(G) = max|f(u) - f(v)| : uv \in E$$

i. Objetivo del proyecto

Partiendo de ellos, se busca crear una estructura de datos que nos ayude a realizar un cálculo que, tal vez no sea el óptimo, buscar reducir los tiempos de ejecución haciendo uso de estructuras de datos capaces de poder tomar ventaja de los ciclos necesarios para poder trabajar en el problema

ii. Comparativa de eficiencia de algoritmos

II. Pseudocódigos

```
for para i = 0 en lista de aristasAdyacentes do
    maxDif ← 0 difAbs = abs(solucion[primer elemento respecto a la lista de
     aristasAdyacentes] - solucion[segundo elemento respecto a la lista de
     aristasAdyacentes]);
    if difAbs >maxDif then
        maxDif ← difAbs ;                          /* ir guardando el máximo */
    end
    return maxDif;
end
```

Algorithm 1: Evaluación Secuencial

```
for para i = 0 en lista de aristasAdyacentes[u].vecinos.size() do
    maxDif1 ← 0 difAbs = abs(solucion[aristasAdyacentes[aristas_v[u].positions[i]].first] -
     solucion[aristasAdyacentes[aristas_v[u].positions[i]].second]); ;      /* se itera
     respecto a los vecinos que tenga u */
    if difAbs >maxDif then
        maxDif1 ← difAbs ;                         /* ir guardando el máximo */
    end
end
for para i = 0 en lista de aristasAdyacentes[v].vecinos.size() do
    maxDif2 ← 0 difAbs = abs(solucion[aristasAdyacentes[aristas_v[u].positions[i]].first] -
     solucion[aristasAdyacentes[aristas_v[u].positions[i]].second]); ;      /* se itera
     respecto a los vecinos que tenga v */
    if difAbs >maxDif then
        maxDif2 ← difAbs ;                         /* ir guardando el máximo */
    end
end
return max(maxDif1,maxDif2);
```

Algorithm 2: Evaluación Incremental

III. Análisis matemático

Nuestro primer algoritmo de evaluación secuencial tiene que recorrer los n elementos que contenga el la lista de aristas adyacentes, siendo una complejidad lineal del orden $O(n)$. Nuestro segundo algoritmo de evaluación incremental al conocer el indice, se itera respecto a los vecinos que pueda tener. Se realiza tanto para u y v, con una complejidad constante respecto a la cardinalidad de ambos $O(u + v)$. Pero depende de la cardinalidad que pueda tener y variando de grafo en grafo.

IV. Experimentación y Resultados

La evaluación secuencial se hace respecto a la cardinalidad del vector de aristas adyacentes.

Para evitar el barrido en los cambios de etiquetado (swap(u,v)) se propone aprovechar la primera corrida para crear un vector que almacenará un objeto de tipo EdgeInfo que contiene el índice (consecutivo usado para hacer referecia a él). De esta forma se logra evitar recorrer nuevamente para el cálculo de un nuevo **Edge Bandwidth**, reduciendo el cálculo a la cardinalidad del vector de aristas vecinas de los elementos a intercambiar en el swap(u,v) w parejas de u; p parejas de v.

De esta manera el vector se pobla al momento de ir formando la lista de adyacencia, y se agregarán los indices vecinos al iterar la lista de adyacencia formando los pares.

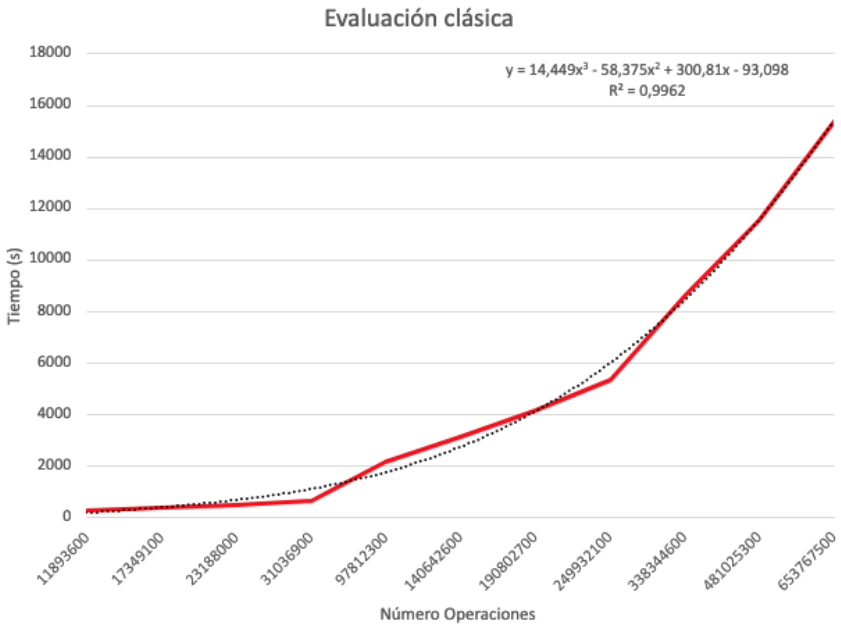

Figura 1: *Gráfico de resultados Evaluación Clásica*

Cuadro 1: *Tabla de resultados con la evaluación clásica*

Problema	Evaluación	Llamadas	Tiempo (ms)
n100 p0,5	Clásica	11893600	269,86
n100 p0,6	Clásica	17349100	381,1
n100 p0,7	Clásica	23188000	502,8
n100 p0,8	Clásica	31036900	685,73
n200 p0,5	Clásica	97812300	2187,36
n200 p0,6	Clásica	140642600	3125,26
n200 p0,7	Clásica	190802700	4181,23
n200 p0,8	Clásica	249932100	5344,96
n300 p0,5	Clásica	338344600	8573,06
n300 p0,6	Clásica	481025300	11572,93
n300 p0,7	Clásica	653767500	15408,46

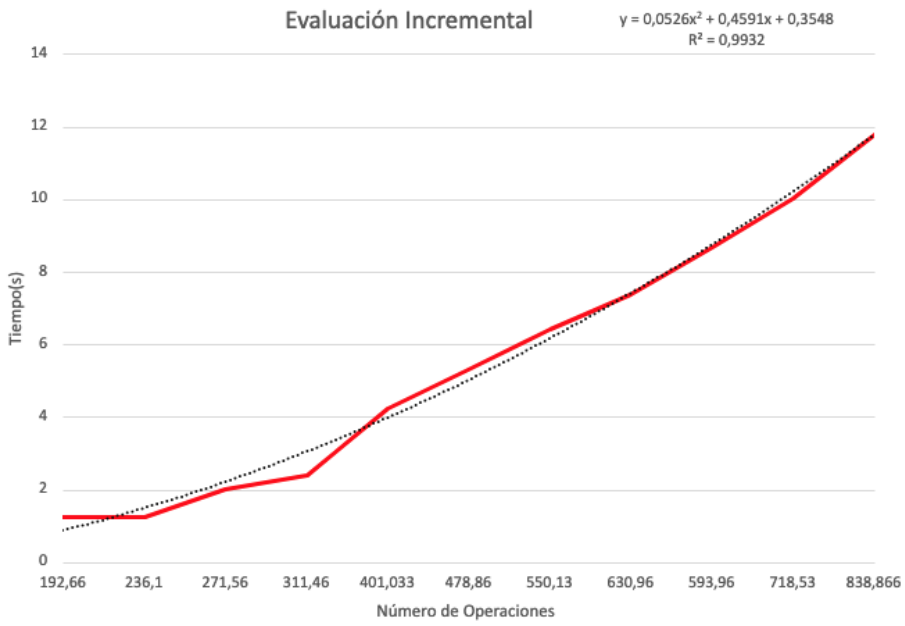

Figura 2: *Gráfico de resultados Evaluación Incremental*

Cuadro 2: *Tabla de resultados con la evaluación incremental*

Problema	Evaluación	Llamadas	Tiempo (ms)
n100 p0,5	Incremental	192,66	1,26
n100 p0,6	Incremental	236,1	1,26
n100 p0,7	Incremental	271,56	2,03
n100 p0,8	Incremental	311,46	2,4
n200 p0,5	Incremental	401,033	4,23
n200 p0,6	Incremental	478,86	5,3
n200 p0,7	Incremental	550,13	6,43
n200 p0,8	Incremental	630,96	7,4
n300 p0,5	Incremental	593,96	8,7
n300 p0,6	Incremental	718,53	10,03
n300 p0,7	Incremental	838,866	11,8

Considerando los resultados de las graficas déspues de experimentar la corrida con grafos bipartitos completos, el comportamiento es del orden polinomial. Esto es debido al aumento del problema en cada ejecución.

En la segunda grafica se observa la tendencia lineal de la evaluación incremental, no obstante no podemos considerar que una es mejor que la otra. Cabe señalar que no atacamos el problema **max bandwidth** de manera de busca el óptimo y sólo realizamos el intercambio de posiciones evitando recalcular toda la lista de pares de aristas adyacentes. Pero puede ser piedra angular para aplicarlo a una metaheurística de manera de ir encontrando una mejor solución respecto a los objetivos del **max bandwidth**

V. Aprendizajes

Al cursar el curso de Análisis y diseño de algortimos, logré acercarme al análisis de grafos más allá de lo visto en matemáticas discretas. El poder manipularlo, explorarlo y lograr obtener una herramienta poderosa como lo es una estructura y representación de un grafo. Es bien sabido que varios problemas en nuestra vida cotidiana se pueden llegar a representar con grafos. Tal es así que traté de llevarlo a mis otras materias, así como el empujón que siempre quise tener para comenzar a programar en C++. Salirme de mi zona de confort, y ampliar mis conocimientos en lenguajes que no pasan de moda y altamente portable.

Aunque es un hecho que no repartí mi tiempo de forma equitativa entre todas las materias, logré llevar los conocimientos de clase a los demás proyectos como en computo paralelo, desarrollando el proyecto en C++ y cambiando mi paradigma a utilizar algoritmos vistos en clase.

Se muy bien que mi desempeño pudó ser mejor y a pesar de los altibajos logré con ayuda del Dr. Eduardo Tello a quién agradezco por sus palabras de aliento cuando imaginaba que mi proyecto de estudiar una Maestria en Ciencias se venía abajo. Eso ya no es un problema para mí, ya que considero que los conocimientos adquiridos este cuatrimestre serán de gran ayuda para el desarrollo de códigos adaptables y modulares cuando me toque regresar a la vida laboral.

No espero obtener una buena calificación, pero reconozco el esfuerzo que dedique en la materia con la esperanza de obtener una nota mínima aprobatoria.

VI. Conclusiones

Como se mencionó en clases, la algorítmica es reconocida como la pieda angular de las ciencias computacionales. El saber de ellas, y aplicarlas es de suma importancia para el desarrollo de códigos limpios.

El manejo de abstracciones de clases, las estructuras de datos para generar algoritmos eficientes es de suma importancia. Un ejemplo de ello fué en el presente trabajo donde se logró reducir considerablemente el tiempo de ejecución con simples estructuras aprovechando los ciclos donde se forma la información, para así hacer uso de ellos en pasos posteriores, aunque pueda afectar en el crecimiento espacial, es algo con el que podemos llegar a lidiar. Quitandonos así de tiempos de ejecuciones altos, logrando eficienciar nuestro código.

Referencias

[1] New results on edge-bandwidth, Theoretical Computer Science, Tiziana Calamoneri(a), Annalisa Massini(a), Imrich Vrto(b), (a) Computer Science Department, University of Rome, (b) Institute of Mathematics, Slovak Academy of Sciences, Department of Informatics, Bratislava, 2003

[2] On the edge-bandwidth of graph products, Theoretical Computer Science, József Balog(a), Dhruv Mubayi(b), András Pluhár(c), (a) Department of Mathematical Sciences, The Ohio State University, Columbus OH, USA, (b) Department of Mathematics, Statistics, and Computer Science, University of Illinois, Chicago IL, USA, (c) Department of Computer Science, University of Szeged, Hungary, 2006