FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 3 — Redes de Flujo

[75.29/95.06] Teoria de Algoritmos I
Primer cuatrimestre de 2022

Alumno	Padron	Email
BENITO, Agustin	108100	abenito@fi.uba.ar
BLÁZQUEZ, Sebastián	99673	sblazquez@fi.uba.ar
DEALBERA, Pablo Andres	106585	pdealbera@fi.uba.ar
DUARTE, Luciano	105604	lduarte@fi.uba.ar
PICCO, Martín	99289	mpicco@fi.uba.ar

Entrega:	Primera
Fecha:	Miercoles 18 de Mayo del 2022

Índice

1. Parte 1: El viaje a Qatar

1.1. Problema enunciado

Se nos presenta con el siguiente problema: Dado un grupo de ciudades, que incluye una ciudad de origen O y una ciudad destino D, y una serie de caminos que conectan dichas ciudades que tienen de atributo una capacidad máxima de personas que pueden circular y además un costo por persona, encontrar cual es la máxima cantidad de personas que pueden ir desde O hasta D al menor costo posible. Podemos asumir que la capacidad de personas es siempre positiva y entera, al igual que los costos por persona.

1.2. Transformacion al problema de flujo máximo con costo mínimo

El problema enunciado anteriormente se puede representar mediante una red de flujo donde la ciudad de origen es la fuente, el destino es el sumidero, y donde ademas, las ciudades serían los vértices, los caminos las aristas y las personas que van a viajar, el flujo. A diferencia de las redes de flujo con las que veníamos trabajando, las aristas en este caso van a tener otro atributo ademas de la capacidad. Tambien, por cada arista vamos a tener un costo por unidad de flujo. En el modelo presentado, la capacidad representa a la cantidad maxima de personas que pueden ir de una ciudad a otra. Luego, el costo por unidad de flujo es el costo por persona que utilice la ruta que conecte 2 ciudades. Donde el costo de un flujo cualquiera resulta la sumatoria del producto entre el flujo resultante en cada arista y el costo por unidad de de cada arista. Entonces, el problema presentado en el cual debemos encontrar las rutas para transportar la máxima cantidad de personas al costo mínimo se traduce en un problema de encontrar el flujo máximo con costo mínimo.

1.3. Resolucion del Flujo maximo con costo mínimo

Para resolver este problema vamos a utilizar un algoritmo conocido como el "Cycle cancelling algorithm" que fue originalmente planteado por Klein en 1967. [**Klein**] Lo que se propone en este algoritmo se divide en dos grandes partes, encontrar el flujo maximo y luego minimizar el costo.

1.3.1. Ford-Fulkerson

Primero hay que encontrar el flujo máximo. Esto podemos hacerlo mediante Ford-Fulkerson. Para explicar este método debemos primero explicar los conceptos de camino de aumento y grafo residual. Dado una red de flujo $G = (V, E)$ y un flujo f, el grafo residual G_f es un grafo dirigido que indica como podemos cambiar el flujo de la red G. Para crear el grafo residual, primero obtenemos las capacidades residuales siguiendo[**CLRS**]:

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{si}(u,v) \in E \\ f(v,u) & \text{si}(v,u) \in E \\ 0 & \text{si no} \end{cases}$$

Es decir que la capacidad residual de una arista (u, v) va a ser su capacidad menos el flujo que pasa por esa arista y la de una arista (v, u) va a ser directamente el flujo. Entonces, utilizando los mismos vértices de la red solo que por cada arista con flujo en la red, el grafo residual va a tener dos aristas.

Luego, un camino de aumento es simplemente un camino que vaya de la fuente al sumidero mediante el grafo residual.[**CLRS**]

Entonces, sabiendo estas dos cosas podemos plantear el algoritmo. Primero, inicializamos el flujo total en 0, es decir ponemos en 0 el flujo por cada arista. Luego, mientras haya un camino de aumento, esto puede hallarse por DFS o BFS por ejemplo, sabemos que el cuello de botella va a ser la mínima capacidad en este camino, pues no va a poder pasar más flujo que el cuello de botella. Luego, aumentamos el flujo en este camino en el grafo residual, saturando la arista del camino cuya capacidad era la mínima. Esto se hace con que, por cada arista en el camino, si esta

arista pertenece a la red de flujo entonces el nuevo flujo va a ser el actual más el cuello de botella. Si no, significa que la arista es una arista que vuelve en el grafo residual y debemos restarle el cuello de botella.[**CLRS**]

1. Pseudocódigo

```
Ford_fulkerson(G,s,t)
    por cada arista (u,v) en G
        inicializar flujo de (u,v) en 0

    mientras haya un camino de aumento P
        Sea c_m la menor capacidad de P
        por cada arista (u,v) en P
            si (u,v) pertenece a G
                (u,v).flujo = (u,v).flujo + c_m
            si no
                (v,u).flujo = (v,u).flujo - c_m
```

2. Complejidad En primer lugar, se asigna un flujo 0 a cada arista. Esta operación cuesta $O(E)$. Luego, tenemos un ciclo que itera mientras haya un camino de aumento. Analizar este ciclo es un tanto más complejo.

   Primero, como todas las capacidades son enteras positivas, todas las capacidades residuales del grafo G_f van a ser enteras positivas. Luego, sea G una red de flujo con un flujo f y sea \$P % un camino de aumento. Entonces, por lo mencionado en la explicación del algoritmo, el flujo nuevo es el actual más el cuello de botella, que es positivo y entero. Esto implica que el flujo nuevo es siempre mayor al anterior por al menos una unidad. Por otro lado, para buscar una cota máxima de flujo, suponemos que todas las aristas salientes de la fuente van a estar saturadas. Con esta suposición, el flujo máximo es la sumatoria de la capacidad de este subconjunto de aristas. Llamando a este número C, sabemos que c es la una cota superior del flujo máximo[**Tardos**]. Esto implica que como mucho, el C iteraciones. En cada iteración, se visita cada arista del camino de aumento. Por lo tanto, la complejidad final de ese ciclo es $O(C.E)$

   Entonces, la complejidad final de Ford-Fulkerson resulta $O(C.E)$.

3. Optimalidad Como se mencionó en la explicación del método, si hay un camino de aumento, podemos aumentar el flujo donde el flujo nuevo va a ser mayor al flujo anterior. Entonces, cuando no haya más caminos de aumento, el flujo va a ser máximo. Por lo presentado en la sección de complejidad, sabemos que eventualmente no va a haber más caminos de aumento. Entonces, la solución debe ser optima.

1.3.2. Ciclos negativos

Luego, la segunda parte del algoritmo se trata de minimizar el costo del flujo máximo calculado. Para ello, es importante entender como funciona el costo y como este puede ser representado en el grafo residual. Sea f el flujo sobre una red, E el conjunto de las aristas y c(x) una función costo, tenemos que el costo del flujo es:

$$c(f) = \sum_{e \in E} f(e)\ c(e)$$

El hecho de que tengamos un costo en las aristas va a impactar en el grafo residual de la forma que el par del grafo residual de una arista en la red de flujo va a tener el mismo costo pero negativo[**Kelin**] . Es decir, notando a una arista e como los vértices u,v que conecta, en orden de la dirección, esto lo representamos como:

$$c(u, v) = -c(v, u)$$

Luego, un teorema [**Busacker**] para redes de grafos indica que un flujo f es el de costo mínimo si y solo si no hay ciclos negativos en los costos del grafo residual. Esto es porque [**Erickson**]:

Sea *gamma* un ciclo de costo negativo en el grafo residual G_f y llamando C_m a la capacidad residual mínima presente en *gamma*. Podemos aumentar el flujo sobre este ciclo de forma que las aristas que no pertenezcan al ciclo se vean inafectadas y las que si pertenezcan al ciclo se modifiquen así:

$$f_{nuevo}(u,v) = \begin{cases} f(u,v) + C_m & \text{cuando}(u,v) \in \gamma \\ f(v,u) - C_m & \text{cuando}(u,v) \in \gamma \end{cases}$$

Entonces, el costo del nuevo flujo se obtiene de:

$$c(f_{nuevo}) = c(f) + c_m \; c(\gamma)$$

De acá, es evidente que mientras $c(\gamma)$ sea negativo, es decir que existan ciclos de costos negativos va a existir un flujo más barato que el actual. [**Erickson**]

Entonces, ahora entendemos que es necesario que el grafo residual final no tenga ciclos negativos de costo. Para encontrar los ciclos negativos podemos utilizar Bellman-Ford.

Teniendo esto en consideración, el algoritmo para calcular el flujo máximo de costo mínimo, que fue originalmente propuesto por Morton Klein en 1967, propone 5 pasos para resolver el problema [**Klein**]:

1. Obtener el flujo máximo de la red sin considerar el costo.
2. Actualizar el grafo residual G_f con el costo por unidad de flujo negativo, como se mencionó previamente.
3. Probar si hay ciclos negativos dirigidos en el grafo G_f. En caso de no haber, se terminó el problema.
4. Se redistribuye el flujo de manera que se satura una de las aristas del ciclo negativo.
5. Repetir desde el punto 2.

Entonces, la idea es primero obtener el flujo máximo da la red dada, lo hacemos por Ford-Fulkerson. Luego, por lo presentado anteriormente, mientras existan ciclos de costo negativo sabemos que se puede reducir el costo saturando una de las aristas del ciclo negativo. Como sucede en Ford-Fulkerson, cada vez que aumentamos el flujo reducimos el costo total del flujo por al menos 1. Esto último se cumple porque los costos de cada arista son enteros positivos, pues el precio de los pasajes son números enteros. De esta manera, tenemos que si comenzamos con un costo inicial $C_i(f)$ vamos a ir reduciendo el costo de a por lo menos 1 hasta llegar al costo final $C_f(f)$. Entonces, el algoritmo eventualmente va a finalizar en $C_f(f) - C_i(f)$ iteraciones y obtendremos la solución óptima.

1.3.3. Pseudocodigo

```
Cycle-canceling():
    Sea f el flujo maximo
    Sea G(f) el grafo residual
    Obtener f y G(f) mediante Ford-Fulkerson
    costo_min = 0
    Mientras G(f) tenga un ciclo negativo C:
        Obtener el C mediante Bellman-Ford
        Sea c_m la menor capacidad residual de C
        Por cada arista (u,v) en C
            si (u,v) pertenece a C
                (u,v).flujo = (u,v).flujo + c_m
            si no
                (v,u).flujo = (v,u).flujo - c_m

    Por cada arista (u,v) en G
        costo_min += (u,v).flujo * (u,v).costo


    Retornar f y costo_min
```

1.3.4. Análisis temporal y espacial

Primero aplicamos Ford-Fulkerson para obtener el flujo máximo cuya complejidad temporal es $O(C\ E)$, donde C es la capacidad del flujo y E la cantidad de aristas que tiene la red. Luego, se aplica el algoritmo de Bellman-Ford, cuya complejidad es $O(V\ E)$ hasta que no haya mas ciclos negativos. Siendo que, como mencionamos antes, cada iteración reduce como mínimo en 1 el costo, se va a llamar a Bellman-Ford un máximo de $C_i(f) - C_f(f)$ veces. Ahora, el costo inicial depende del algoritmo por lo tanto es mejor establecer una cota superior al costo inicial. Llevándolo a un extremo, sea C_{max} la capacidad más alta de una arista, a K_{max} como el costo más alto de una arista, entonces, el costo inicial tiene una cota superior tal que $C_i(f) \leq K_{max}\ C_{max}\ E$. Pensando que C_f tiene que ser positivo, podríamos maximizar a la resta como $C_i(f) - C_f(f) \leq K_{max}\ C_{max}\ E$. Por lo tanto, el algoritmo de Bellman-Ford va a ser llamado un máximo de $K_{max}\ C_{max}\ E$ veces. Entonces, la complejidad temporal del algoritmo resulta $O(V\ E^2\ K_{max}\ C_{max})$.

1.4. Detalles de implementación

El algoritmo fue implementado en Python y probado con la versión `3.10.4`.

Para la ejecución del algoritmo normal no hay dependencia, para exportar el grafo a imagen, se necesita como dependencia `graphviz` que se puede instalar con:

```
pip install graphviz
```

1.4.1. Ejecución del programa

El programa contiene un `shebang` para ser ejecutado en una terminal de la siguiente forma:

```
./src/parte_1.py <filename>
```

El comprimido entregado incluye un carpeta en `assets/` con grafos ejemplos, por ejemplo:

```
./src/parte_1.py ./assets/grafo-qatar.csv
```

```
La cantidad maxima de personas que pueden viajar es:  6
El costo de todos los viajes es:  14
```

1. Exportador de Grafo a Imagen

   Aparte de esto, esta incluido un exportador que genera un imagen en formato *SVG* de los grafos y se puede generar con el siguiente comando:

   ```
   ./src/export.py ./assets/grafo-qatar.csv
   ```

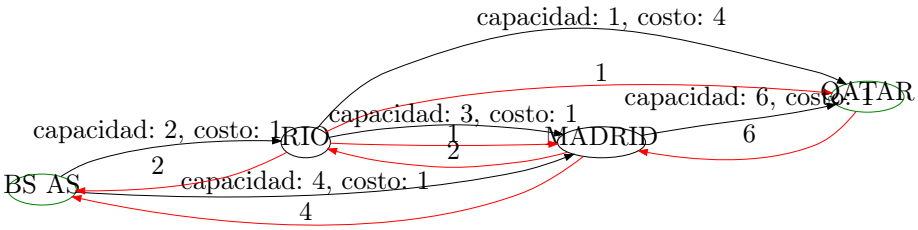

Figura 1: Hospital con un entrenador cargado.

2. Parte 2: Un reality único

3. Fuentes