MA3705. Algoritmos Combinatoriales 2020.
Profesor: José Soto
Escriba(s): Arturo Lazcano y Vicente Maturana.
Fecha: 18 de Diciembre de 2020.

Cátedra 24

1. Algoritmo de Edmonds Karp

Como se vió en la clase pasada, el algoritmo de Edmonds-Karp toma el flujo f, calcula su red residual N^f y se queda solamente con los arcos que tienen capacidad positiva para luego calcular un $s-t$ camino aumentante con el menor número de arcos (Figura 1).

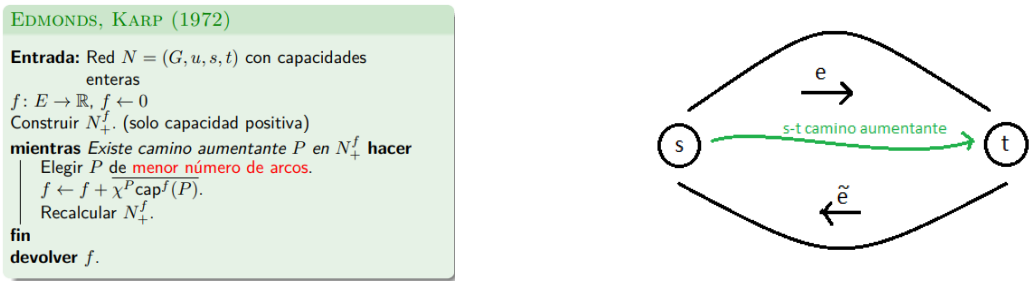

Figura 1

Y queremos demostrar lo siguiente:
Teorema. El algoritmo de Edmonds Karp es fuertemente polinomial.

1.1. Capas: BFS en N_f^+

Habíamos quedado en que teníamos un grafo G_i residual (N^f_+) en la i-ésima iteración. Luego, se construyen capas $L^i_0, L^i_1, ..., L^i_r$ basadas en BFS: $dist_i(s,v) = k \iff v \in L^i_v$. Donde L^k_n representa los nodos a distancia n de s y en el grafo residual de la iteración k.
El algoritmo calculará la capacidad residual del camino P ($cap(P) = \min_{e \in P} U^f_{(e)} > 0$ pues usamos los arcos con capacidad residual positiva) y "empujar" esa cantidad de flujo por P, donde "empujar" significa aumentar $cap(P)$ en arcos $e \in E \cap P$ y disminuir $cap(P)$ en arcos e tal que el reverso de e pertenezca a P ($\overleftarrow{e} \in P$). Como se puede ver en el esquema de la Figura 2, el camino P *respeta las capas*, es decir, va de una a la siguiente.

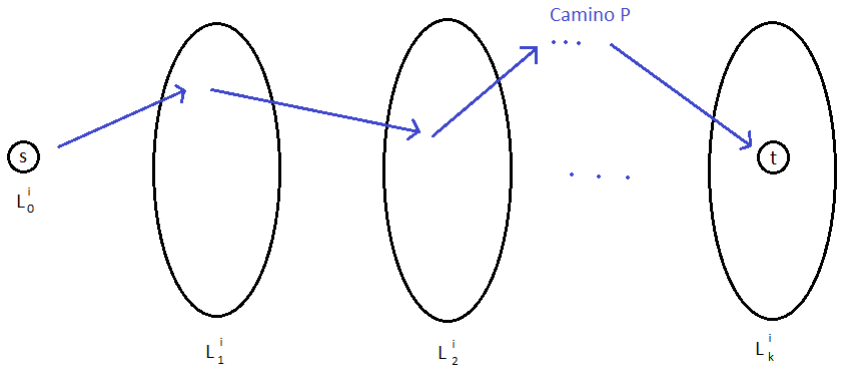

Figura 2

Como se mencionó la clase pasada, tenemos el siguiente lema:
Lema 1.
Arcos que salen: $E(G_i) \setminus E(G_{i+1}) \subseteq P$ (arcos saturados). Además, al menos un arco de P sale.
Arcos que entran: $E(G_{i+1}) \setminus E(G_i) \subseteq$ reversos de P.

Un arco entra en la siguiente iteración cuando:
1) Hay un arco $e \in E$ y originalmente no tenia nada de flujo, entonces al empujar un poco de flujo por ahí, el arco reverso empieza a tener capacidad por lo que este arco reverso podría aparecer en la siguiente iteración.
2) Un arco de P es en verdad un arco reverso ($\overleftarrow{e}$) de un arco e donde e es un arco que se usa para devolver flujo pero no puedo empujar flujo (está saturado). Por lo tanto si empujo un poco por $\overleftarrow{e}$, el arco e empieza a tener capacidad y puede aparecer en la siguiente iteración

Lo anterior trae como consecuencia los siguientes lemas:

Lema 2. $\forall v, i : dist_i(s, v) \leq dist_{i+1}(s, v)$.

Lema 3. $\forall e \in E \cup \overleftarrow{E}$. e sale de $E(G_i)$ para, a lo más, n/2 iteraciones i distintas.

Demostración Lema 2:
Primero, agregaremos los arcos del reverso de P que entran, notando que es imposible que este arco sea usado en el camino más corto de s a v, es decir, $dist(s, v)$ se mantiene. Por último, eliminaremos los arcos directos de P que salen, es decir, $dist(s, v)$ solo puede crecer, pues para un grafo cualquiera G, al quitar una arista (arco), la distancia de un vértice a otro solo puede mantenerse o aumentar.

Demostración Lema 3:
Usaremos el lema anterior. Sea $ab = e \in P$ con a en la capa k y b en la capa $k + 1$. Si e sale en la iteración i, y quiere volver a entrar en otra iteración j, b estará en una capa $\geq k + 1$, por lo tanto, para que e entre, a tiene que estar en una capa $\geq k + 2$, pues un arco $e = ab$ entra solo si e es reverso de P. Por lo que, cada vez que e entra, el nivel de la cola (nivel de a) subió en 2 unidades y como cada vértice puede subir de nivel, a lo más n veces, entonces hay a lo más n/2 iteraciones.

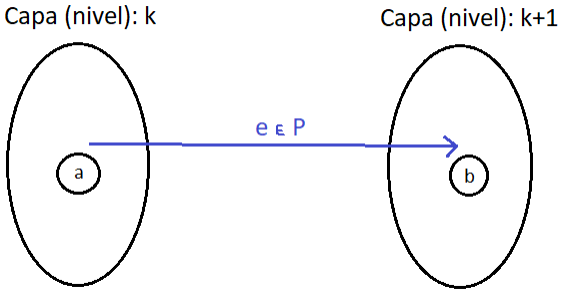

Figura 3

1.2. Complejidad de Edmonds-Karp 2

Con los lemas vistos anteriormente, podemos concluir la complejidad final para este algoritmo de caminos de mayor capacidad:

1. En la iteración i, al menos un arco sale de $E(G_i)$
2. Como cada arco sale a lo más $O(n)$ veces, el número total de iteraciones es $O(nm)$

3. Cada iteración toma $O(n+m) = O(m)$ pensando en que el grafo original satisface, por simplicidad, $n = O(m)$

Por lo tanto tenemos:

Teorema. Edmonds-Karp tiene complejidad $O(nm^2)$

1.3. Resumen Algorítmico para Flujos y Cortes (suponiendo $n = O(m)$)

El algoritmo de Edmonds-Karp II, publicado en 1972 constituye el primer algoritmo descubierto para el cálculo de flujos máximos en tiempo (fuertemente) polinomial ($O(nm^2)$). Además ocurre el mismo fenómeno que encontramos cuando analizamos matchings y cubrimientos, es decir, si conocemos previamente un flujo máximo f, es posible encontrar, en tiempo $O(m)$, un s -t corte mínimo.

Veamos un resumen de lo que se sabe para algoritmos de flujos máximos con valores enteros:

Recordemos que Ford-Fulkerson lo que hace es empujar por *cualquier* camino, es decir, a la hora de elegir un camino aumentante, lo hace al azar, y por lo tanto podría suceder que cada vez aumentase el valor del flujo en una sola unidad, con lo que es posible deducir que su complejidad es $O(m\,\mathrm{OPT})$. La cátedra anterior vimos que Edmons-Karp I, que elije el camino aumentante con más capacidad, tiene complejidad $O(m \log \mathrm{OPT})$ (esto sin considerar que es posible usar el algoritmo de Dijkstra para meodificarlo y hacerlo más rápido) y este es débilmente polinomial. Como ya mencionamos y mostramos antes, Edmons-Karp II tiene complejidad $O(nm^2)$ lo que ya es fuertemente polinomial (pues no depende de cómo escribamos el óptimo).
Durante los 80's, hubo algunos avances mediante modificaciones y el uso de nuevas estructuras de datos, que permitieron generar algoritmos de complejidad $O(n^2m)$, $O(n^3)$, y hasta $O(nm \log n)$. El mejor algoritmo que tenemos hasta ahora para este problema es uno que mezcla resultados de dados Oriln y por King, Rao, Tarjan, y tiene complejidad $O(nm)$

De manera más ordenada, estos son lo algoritmos que tenemos para el cálculo de flujos maximos a valores enteros:
Nota: OPT es el valor del flujo máximo.

Algoritmos	Complejidad
Ford-Fulkerson (1956)	$O(m\,\mathrm{OPT})$
Edmons-Karp I (1972)	$O(m \log \mathrm{OPT})$
Edmons-Karp II (1972)	$O(nm^2)$
Goldberg-Tarjan (1988)	$O(n^2m), O(n^3), O(nm \log n)$
Orlin+King Rao Tarjan (2013)	$O(nm)$

Cabe mencionar que estos algoritmos son para grafos generales, y existen muchas mejoras para grafos con estructuras especiales, (grafos es bipartitos, grafos con capacidades binarias, etc.) y es un problema abierto encontrar mejores algoritmos para el cálculo de flujos máximos a valores enteros.

2. Método del Elipsoide y Programación Lineal

Recordemos; Programación Lineal se refiere a los problemas de optimización de un programa lineal en un poliedro determinado, es decir, un conjunto definido por desigualdades lineales, el cual a priori podría ser vacío, y nos referimos de manera genérica a ellos como PL. De manera formal, escribimos un PL de la siguiente manera:

$$\begin{aligned} \text{máx}\, c^\top x \\ Ax \leq b \\ x \geq 0 \end{aligned}$$

Consideraremos que, en general, los valores son racionales (y mediante escalamiento, enteros) para poder escribirlos en un computador. Veamos, de manera somera, la historia de los algoritmos conocidos para resolver un PL.

- En 1827 Fourier publica su método para determinar factibilidad eliminando variables, muy parecido a el método de Gauss para la resolución de sistemas lineales (en este caso es la resolución de un sistema de desigualdades). Es un método finito, pero demasiado lento. Este es ahora conocido como el método de Fourier-Motzkin.
- En 1939 Kantorovich propone un método para resolver un PL muy similar al Simplex actual, mas siendo parte de lo que era la Unión Soviética en ese tiempo, este algoritmo no fue conocido en occidente.
- En 1947 Dantzig propone el método Simplex (sin saber del trbajo de Kontorovich) y algunas simplificaciones. Este método permite resolver un PL a mano, mediante el uso de tablas. Durante los años se han hecho muchas simplificaciones y actualmente tenemos un algoritmo Simplex simplificado que resulta fácil de programar y de enseñar. Sin embargo, Simplex no es polinomial, pues existen ejemplos donde Simplex puede demorar demasiado debido a cómo elige los vértices por los que pasa (lo cual, en otros casos lo puede hacer muy rápido). Es decir, Simplex no tiene la garantía de terminar rápido, aunque en la práctica generalmente Simplex demora poco.
- En 1947 Von Neumann desarrolla la teoría de la dualidad para PL.
- Durante la década de 1960, Cobham y Edmonds desarrollan la tesis de polinomialidad, es decir, la noción de que un algoritmo es eficiente cuando se puede implementar en tiempo polinomial, lo que es conocido actualmente como la clase de problemas P.
- Durante esta misma década, PL empieza a ser muy importante para distintos ámbitos, como las finanzas, la guerra, en empresas, más allá de la academia, pero se desconocía si existían algoritmos polinomiales para éste.

Como respuesta a la pregunta sobre la existencia de algoritmos polinomiales para PL, se desarrolla el método del elipsoide, por lo que ahondaremos un poco en qué es una elipsoide y cómo funciona el método, al menos de manera heurística.

Definición 1 (Elipsoide)**.** Dado un punto $\overline{x} \in \mathbb{R}^d$ y una matriz $M \in \mathbb{R}^{d\times d}$ semi-definida positiva, la elipsoide de centro $\overline{x}$ definida por M es:

$$E(M, \overline{x}) = \{y \in \mathbb{R}^d : (x - \overline{x})^\top M^{-1}(y - \overline{x}) \leq 1\} \tag{1}$$

Observación La condición de que M sea semidefinida positiva permite que la elipsoide resultante sea de volumen nulo (si estuviésemos en $\mathbb{R}^3$ por ejemplo, podría ser una elpsoide en un plano), mas, pidiendo que la matriz M sea definida positiva nos aseguramos que la elipsoide E tenga volumen.

Ejemplo 1. La elipsoide ortogonal con ejes de largo r_i para coordenada i es

$$E\left(0, \begin{pmatrix} r_1^2 & & \\ & \ddots & \\ & & r_n^2 \end{pmatrix}\right)$$

Podemos calcular volúmenes de elipsoides, por ejemplo, mediante fórmulas que involucran la determinante de la matriz semidefinida positiva asociada. La razón por la cual las elipsoides resultan de interés es que, dada una elipsoide E, al hacer pasar un hiperplano por el centro de esta, se puede obtener una elipsoide mínima E' que es la elipsoide más pequeña que contiene a una mitad de E, y se verifica que

$$\text{Vol}(E') \leq \text{Vol}(E)\exp(-\frac{1}{2d}).$$

Este resultado es conocido como el **Lema de la media elipsoide**. Este lema no solo nos da la desigualdad de arriba, sino que también asegura la existencia de la elipsoide minimal.

Lo siguiente será una aplicación de este lema.

Dado un convexo $K \subseteq R^d$ y un punto x, suponemos que tenemos acceso a un oráculo que responde de la siguiente forma

1. $x \in K$
2. Reporta un hiperplano de separación de dirección c tal que $c^\top x < c^\top y$ para todo $y \in K$.

Es decir, el oráculo me dice si el punto está dentro de K o no, y de ser la respuesta negativa, entrega además un plano que separa al punto y al convexo.
Problema General: Dado un oráculo para K y una elipsoide E_0 que contiene a K, encontrar un punto $x \in K$.

Una idea razonable para atacar el problema es la siguiente: Primero preguntemos si el centro x_0 de la elipsoide E_0 está en el convexo o no. Si está en K terminamos, si no, entonces el oráculo reporta un hiperplano de separación que indica para que lado está el convexo, y que sin pérdida de generalidad, podemos tomar tal que paso por x_0. Por el lema de la media elipsoide, somos capaces de encontrar una segunda elipsoide E_1 que contiene a K (pues sabemos para qué lado del hiperplano está contenido K) y que tiene un volumen más pequeño que la original (en una razón geométrica). Repetimos el mismo procedimiento sobre E_1, y podríamos encontrar E_2, etc. Es decir, podemos armar una secuencia de elipsoides E_i que contienen a K y cuyo volumen decrece de manera geométrica y en algún punto (si el convexo K tiene un volumen asociado) podremos encontrar una elipsoide con menor volumen que K, lo que nos asegurará que su centro está en éste.

Veamos algunas virtudes/problemas de este procedimiento:

- Si $\frac{\text{Vol}\, K}{\text{Vol}\, E_0}$ no es tan pequeño (formalmente, si $d \ln \frac{\text{Vol}\, K}{\text{Vol}\, E_0} \geq C$), entonces encontramos un punto de K en a lo más C iteraciones.
- Si solo nos interesa un punto suficientemente cerca de K es es bueno.
- El problema es cuando $\text{Vol}(K) = 0$, en cuyo caso no podemos podemos aproximar. Otro problema es cómo encontrar la elipsoide inicial, pues está podría ser difícil de encontrar.

Sigamos con nuestra linea de tiempo:

- En 1972, métodos de este estilo (algoritmos iterativos para programas convexos) fueron estudiados por Shor y luego Nemirovski/Yudin y proponen el **método de la elipsoide**, que acabamos de discutir.
- En 1979 Khachiyan propone una versión del método de la elipsoide que sirve para factibilidad de PL's y corre en tiempo polinomial.

Veamos un poco la idea del método de la elipsoide para programas lineales: la idea es aprovechar la dualidad fuerte de los PL. El programa dual del PL en la forma que mencionamos antes está dado por:

$$\begin{aligned} \text{mín}\, b^\top y \\ A^\top y \geq c \\ y \geq 0 \end{aligned}$$

Como la solución del problema dual es igual al del problema original, y como tenemos dualidad débil, podemos definir el siguiente problema:

$$\begin{aligned} c^\top x = b^\top y \\ Ax \leq b \\ x \geq 0 \\ A^\top y \geq c \\ y \geq 0 \end{aligned}$$

Gracias a este nuevo programa, la resolución del PL original se reduce a encontrar un punto factible para este último programa o concluir que el conjunto factible es vacío. Llamaremos a tal poliedro Q, y nos interesará utilizar el método del elipsoide. Un posible problema para esto es que Q podía no tener vértices, por ejemplo si consideramos Q como el subconjunto del espacio que definido como lo que se encuentra entre dos hiperplanos paralelos, y en otros casos podría tener volumen 0.

La ausencia de vértices es un problema pues son, por lo general, lo que devolvemos cuando se resuelve un PL, y el de volumen nulo ya lo discutimos antes. Sin embargo, gracias a técnicas de álgebra lineal como capaces de transformar Q en Q', donde este último tiene vértices y volumen no nulo. Además, podemos caracterizar los vértices como soluciones únicas de subsistemas $Cx = p$ de Q. En general nos interesará verificar que estos x_j no están muy lejos del origen, y esto lo podemos hacer usando la fórmula de Cramer.

Se deduce entonces una cota para los valores de x_j. Los vértices de Q' son puntos de coordenadas racionales cuyo numerador y denominados (que son enteros) están acotados por $(dU)^d$, con U el valor máximo de los datos del problema y d la dimensión del espacio. En conclusión, los vértices de Q' están en $E_0 = B(0, \sqrt{d(dU)^{2d}}$ de volumen a lo más $(2\sqrt{d(dU)^{2d}})^d \leq (2d(dU)^{2d+1})^d$. Adicionalmente, por la forma de construcción de Q' tenemos que o bien es vacío o bien$\text{Vol}(Q')$ es al menos $1/(2d(dU)^{2d+1})^d$. Con todo esto es posible mostrar que el método de la elipsoide termina en a lo más $O(d \log \frac{\text{Vol}(Q')}{Vol(E_0} = O(d^3 \log(dU))$ iteraciones, lo que es tiempo (débilmente) polinomial.

Respecto al oráculo de separación necesario para el método de la elipsoide, como el poliedro es explícito, al probar si $x \in Q'$, basta ver si satisface la descripción de este, y si no, podemos devolver cualquier restricción insatisfecha, la cual actúa como hiperplano de separación. Esto es posible hacerlo en tiempo polinomial en d y en $\log U$

Cada iteración del método (calcular la nueva elipsoide) es polinomial (de hecho lineal) en los datos iniciales, por lo que el método de la elipsoide toma tiempo polinomial en d y en $\log U$.

Observación: Si Q' es vacío, el método lo detecta, pues si no fuese vacío acabaría encontrando un punto factible al cabo de ciertas iteraciones, por lo tanto, si se toma más de ese tiempo, concluimos que Q' es vacío.

Terminemos entonces la linea de tiempo:

- En 1979 Khachiyan muestra que es posible resolver PL en tiempo polinomial (es decir, está en la clase P).
- En 1980 Lovasz, Grötshel y Schrijver prueban que la optimización de PL respecto a Q en tiempo polinomial es equivalente a que el oráculo de separación de Q se tome tiempo polinomial en dar su respuesta. **Consecuencia importante:** Se pueden resolver PL's con cantidades enormes de de restricciones (no polinomiales incluso) si se puede implementar un oráculo de separación.
- En 1984 Karmakar propone su algoritmo proyectivo para PL, conocido como método de punto interior. Este algoritmo es más rápido que el método de la elipsoide.
- En 1989 Vaidya muestra algunas mejoras para los métodos conocidos, concluyendo que se pueden resolver PL's en esencialmente $O(n^{2.5} \log U)$
- En 2019, Cohen, Lee y Song logran un algoritmo que es casi tan rápido como multiplicar matrices, más precisamente es básicamente $O(n^{\text{máx}\{\omega, 2+1/6\} \log U})$, donde ω es la constante de multiplicación de matrices (esto toma tiempo $O(n^\omega)$)

Queda entonces un problema abierto: ¿Existen algoritmos fuertemente polinomiales para PL?

Observación: Estos últimos algoritmos mencionados suelen ser muy difícil de implementar, y por lo tanto, se suele optar por usar Simplex, que si bien no es polinomial, en la práctica funciona razonablemente bien.