MA4702. Algoritmos Combinatoriales. 2013.
Profesor: José Soto
Escriba(s): David Hasson , Martín Rios y Cristóbal Rojas.
Fecha: Lunes 03 de Noviembre, 2014.

Cátedra 23

Recuerdo

En la clase anterior vimos el algoritmo de Queyranne, que nos permite encontrar un conjunto S (con $\emptyset \subsetneq S \subsetneq V$) tal que minimice $f(S)$ con f submodular y simétrica.

Algoritmo 1 Queyranne

```
cand ← ∅
f' ← f
V' ← V.
while |V'| ≥ 2 do
    Calcular (s, t) par colgante del sistema (f', V').
    Sea S el conjunto obtenido de {s} expandiendo todos los elementos fusionados.
    cand ← cand ∪ {S}.
    (f', V') ← sistema donde s y t se fusionan.
end while
return argmin{f(X) : X ∈ cand} // el objeto S ∈ cand de menor f(S).
```

Veamos un caso particular: Sea $f : 2^V \to \mathbb{R}, f(S) = w(\delta(S))$ donde (V, E) es un grafo y $w : E \to \mathbb{R}_+$ función de pesos. Fusionar los elementos (vértices) es simple:

Algoritmo 2 $f \to f'$, fusión s y t

```
G' = (V', E')
V' = V − s − t + st
E' = {xy ∈ E : x, y ∈ V − s − t} ∪ {(st)v : sv ∈ E o tv ∈ E}
w' : E' → ℝ+
if e = xy ∈ E(V − s − t) then
    w'(e) = w(e)
else if e = (st)v then
    w'(e) = w(sv) + w(tv) // abusando de la notación: Si sv, tv ∉ E, w(sv) = w(tv) = 0
end if
```

Esto se puede lograr en un tiempo $O(n + m)$.

A continuación presentamos un algoritmo para encontrar un par colgante.

Algoritmo 3 Orden de máxima adyacencia para G no dirigido.

```
Require: Par colgante (G, w).
  v_i ← v ∈ V(G) arbitrario.
  for i = 2, ..., |V(G)| do
    v_i ← argmax{d({v_1, ··· , v_{i−1}}, x) : x ∈ V − {v_1, ··· , v_{i−1}}}.
  end for
  return (v_n, v_{n−1}).
```

Algoritmo 4 Modificación del anterior.

$v_i \leftarrow v \in |V(G)|$ arbitrario.
$d(x) = 0 \ \forall x \in V(G) - v_1$
for $j = 2, \ldots, |V(G)|$ **do**
  for $x \in V \setminus \{v_1, \ldots, v_{j-1}\}$ **do**
    $d(x) \leftarrow d(x) + w(xv_{j-1})$.
  end for
  $v_j \leftarrow \text{argmax}\{d(x) : x \in V - \{v_1, \cdots, v_{i-1}\}\}$.
end for
return (v_n, v_{n-1}).

En la primera versión calcular el máximo toma $O(n)$ y como se realiza sobre todos los vértices toma tiempo $O(n)$. En la modificación la primera asignación, el ciclo anidado y calcular el el mínimo toman tiempo $O(n)$ luego el orden del algoritmo es $O(n^2)$.

Complejidad del algoritmo de Queyranne para el caso de corte global mínimo

1. Encontrar un par colgante de G toma tiempo $O(n^2)$
2. Recalcular fusiones toma tiempo $O(n + m)$

Así el algoritmo de Queyranne toma en total $\#\text{iteraciones} \cdot O(n^2 + m) = O(n^3)$

Observación. *Podemos calcular un corte global mínimo en $O(n^3)$ (mucho mejor que el $O(n \cdot nm^2)$ correspondiente a resolver el problema con flujo máximo). Este tiempo se puede mejorar utilizando colas de prioridad:*
Heaps Binarios: Encontrar par colgante toma tiempo $O(m \log n)$.
Heaps Fibonacci: Encontrar par colgante toma tiempo $O(m + n \log n)$.
Así, se podrá encontrar un corte global mínimo en tiempo $O(n(m + n \log n))$.

Programación lineal

Recordemos que si queremos calcular un flujo máximo f^* de una red una opción habría sido resolver

$$\begin{array}{lll} (P) & \text{máx} & f(\delta^+(s)) \\ & \text{s.a.} & f(\delta^-(v)) = f(\delta^+(v)), \quad \forall v \in V - s - t \\ & & 0 \leq f \leq U. \end{array}$$

Normalmente no es buena idea resolver el problema lineal cuando existen algoritmos combinatoriales disponibles. Pero a veces la única opción es resolver este problema.

Se suele usar el método SIMPLEX para resolver estos problemas dado que es rápido en la práctica, pero en general hay instancias donde SIMPLEX demora tiempo exponencial. Es por esto que a continuación veremos un algoritmo polinomial para resolver PL, en general este algoritmo sirve para programas convexos con ciertas propiedades.

Método de la Elipsoide

Dado K convexo y cerrado en $\mathbb{R}^d$ (de una manera implícita que se verá más adelante), con K satisfaciendo ciertas hipótesis mínimas, este método puede o bien certificar que K es vacío o encontrará $x^* \in K$.

Para describir este método necesitaremos introducir la siguiente notación:

Definición 1. Un Oráculo de Separación para un convexo $K \subseteq \mathbb{R}^d$ es un algoritmo que puede describir K de la siguiente manera:
Dado $x \in \mathbb{R}^d$, el oráculo dirá si $x \in K$ o si no está, en cuyo caso entregará un hiperplano separador $c \in \mathbb{R}^d$, es decir, $c^t x < C_0$ y $c^t y \geq C_0, \forall y \in K$.
Diremos que el oráculo es polinomial cuando ocurre en tiempo polinomial en $(d, \langle K \rangle, N)$, donde

- $N = \#$ bits necesarios para codificar nuestro problema.,
- $d =$ dimensión del espacio.,
- $\langle K \rangle = \#$bits necesarios para codificar el convexo.

A continuación se presentan algunos ejemplos de oráculos.

Ejemplo 1.

1. Encontrar un oráculo polinomial para el convexo $K = \{x \in \mathbb{R}^d : Ax \leq b \text{ y } x \geq 0\}$ con $A \in \mathbb{Q}^{m \times d}, b \in \mathbb{Q}^m$ dados.

Algoritmo 5 ALG(X)

Se denota a_i a la fila i-ésima de A.
if $\forall i \, a_i^T z \leq b \wedge x \geq 0$ **then**
  return $z \in K$.
else
  Devolver la restricción $a_i^t x \leq b^i$ (o $x_i \geq 0$) donde falla.
end if

Observación. *Esto es polinomial en la codificación de A y b (y es polinomial en d).*

2. Dado $G = (V, E)$, $n = |V|, m = |E|$ encontrar oráculo de separación para el convexo $K = \{x \in \mathbb{R}^E : x(\delta(S)) \geq 1, x \geq 0, \forall S \subseteq V; \varnothing \subsetneq S \subsetneq V\}$ en función de n y m

   Si queremos un oráculo polinomial, no podemos revisar todas las desigualdades, pues hay una cantidad exponencial de ellas. Sin embargo, podemos hacer lo siguiente:

Algoritmo 6 ALG(X)

Dado x, revisar si $x \geq 0$.
Utilizar x como función de peso y calcular el corte minimo global S^* del grafo.
if $x(\delta(S)) \geq 1$ **then**
  return true
else
  devolver el hiperplano asociado a S^*, es decir, $\chi^{S^*} = \begin{cases} 1 & \text{si } e \in \delta(S^*) \\ 0 & \text{si no} \end{cases}$
end if

Definición 2. Una elipsoide $E \subseteq \mathbb{R}^d$ es una transformación lineal afín invertible de $B(0,1)$.
En otras palabras: $T : \mathbb{R}^d \to \mathbb{R}^d$; $T(x) = Ax + b$ con A invertible, $b \in \mathbb{R}^d$.

La elipsoide asociada a T es

$$\begin{aligned} T(B(0,1)) &= \{T(x) \in \mathbb{R}^d : x^t x \leq 1\} \\ &= \{y \in \mathbb{R}^d : \left[A^{-1}(y-b)\right]^t \left[A^{-1}(y-b)\right] \leq 1\} \\ &= \{y \in \mathbb{R}^d : (y-b)^t \underbrace{(A^t)^{-1} A^{-1}}_{M^{-1}} (y-b) \leq 1\} \end{aligned}$$

Normalmente llamamos $M = AA^t$ y $E(b, M) = \{y \in \mathbb{R}^d : (y-b)^t M^{-1}(y-b) \leq 1\}$ la elipsoide de centro b y matriz M. Además la elipsoide está bien definida cuando M es semidefinida positiva.

Observación.

- $B(0,1) = E(0, I)$
- $B(0,r) = E(0, r^2 I)$
- *En general la elipsoide con ejes de largo* r_i *y centro* 0 *es* $E\left(0, \begin{pmatrix} r_1^2 & & \\ & \ddots & \\ & & r_d^2 \end{pmatrix}\right)$
- $Vol(E(0; r_1, r_2, \cdots, r_d)) = Vol(B(0,1)) \prod r_i$. *En general* $Vol(E(b, A)) = Vol(B(0,1))\sqrt{\det A} = Vol(B(0,1)) \prod \sqrt{\lambda_i}$, *donde* λ_i *son los valores propios de* A.

Método de Elipsoide

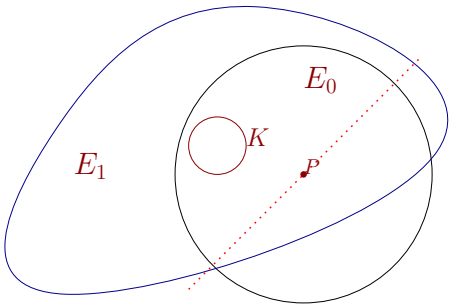
INPUT:

- K convexo cerrado con oráculo de separación.
- Se supondrá que se tienen p, R tales que $K \subseteq B(p, R)$
- Si $K \neq \varnothing$, entonces existe $c \in K$ tal que $B(c, r) \subseteq K$.

donde p, R, r y el oráculo de separación son datos.
OUTPUT: ¿ K es vacio o no ? si no es vacio retornar $x \in K$

Algoritmo 7 Método del Elipsoide

```
E ← B(p, R)
q ← p
while Vol(E) ≥ Vol(B(0, r)) do
  if q ∈ K (preguntar al oráculo) then
    return q
  else
    El oráculo devuelve un hiperplano c tal que c^t q = 0 y c^t y < 0, ∀y ∈ K
    E ← el elipsoide de volumen mínimo que contiene E ∩ {y : c^T y ≤ c^T q}
  end if
end while
return K = ∅
```



Lema 1 (de la media elipsoide)**.** *$E(q,A) \cap \{x : c^T x \leq c^T a\}$ está contenida en $E' = E(q', A')$ con $q' = q - \frac{b}{d+1}$, $A' = \frac{d^2}{d^2-1}\left(A - \frac{2}{d+1}bb^T\right)$ con $b = \frac{Ac}{\sqrt{c^T Ac}}$.*
Además, $\frac{Vol(E')}{Vol(E)} \leq e^{\frac{-1}{2d}}$.

La próxima clase vamos a ver que por este lema número de iteraciones no puede ser muy grande.