MA3705. Algoritmos Combinatoriales. 2014.
Profesor: José Soto
Escriba(s): Juan Granier, Ian Letter, Alberto Rojas y José Soto.
Fecha: 21 de Noviembre 2014 .

Cátedra 28

1. Técnicas para diseñar/analizar algoritmos de aproximación

En esta sección se verá la resolución de algunos problemas de alta complejidad utilizando algoritmos que permiten aproximar por una constante. Antes de ello se enuncian algunos ejemplos.

Ejemplo 1. *TSP: Problema del vendedor viajero*
En este caso, utilizando árboles generadores y matchings se puede obtener una $\frac{3}{2}$ *aproximación.*

Observación 1. *No es trivial obtener aproximaciones en base a constantes para problemas NP-Completos. Muchas veces queda en base a los datos del problema.*

Más ejemplos:

- Glotón (K-aproximación para intersección de K matroides)
- Programación Dinámica (Problema de la mochila)

1.1. Problema 1: Programación Lineal

Vertex-cover (cubrimiento por vértices de peso mínimo)

Dado $G = (V, E), w : V \longrightarrow \mathbb{R}_+$.
Encontrar $C \subseteq V$ cubrimiento de peso mínimo.

Observación 2. *Vertex-cover es NP-Completo, para grafos generales. (En el caso bipartito es polinomial; como ya fue visto).*

Solución. En primer lugar lo modelamos como un problema entero.

Programa Entero:

$$\begin{aligned} \text{mín} \quad & \sum_{v \in V} x_v w_v \\ \text{s.a.} \quad & x_i + x_j \geq 1 \quad \forall e = ij \in V \\ & x_v \in \{0, 1\} \quad \forall v \in V \end{aligned}$$

Se relaja el programa entero, obteniendo un **programa lineal**:

$$\begin{aligned} \text{mín} \quad & \sum_{v \in V} x_v w_v \\ \text{s.a.} \quad & x_i + x_j \geq 1 \quad \forall e = ij \in E \\ & x_v \geq 0 \quad \forall v \in E \end{aligned}$$

Observación 3. *Se puede eliminar la restricción de* ≤ 1 *ya que la función de peso está definida en* $\mathbb{R}_+$*. Luego, si una variable es mayor que 1, el* bajarla *a 1 mejora la solución.*
Notar que es solo valido para funcion de peso no negativo.

Sea x^* una solución óptima del PL, que en general **no** es integral.

Algoritmo 1: Redondeo determinista
return $C = \{v : x^*_v \geq \frac{1}{2}\}$

Probaremos algunos resultados respecto al conjunto obtenido.

Lema 1. *El conjunto devuelto C es cubrimiento.*

Demostración. Sea $e = ij$ en E. Sabemos que $x_i^* + x_j^* \geq 1$. Luego $\exists k \in \{i, j\} \ : x_k^* \geq \frac{1}{2}$. Por lo tanto, $k \in C$, y entonces e está cubierto.

Lema 2. *El peso de C satisface:*

$$w(C) = \sum_{v \epsilon C} w_v = \sum_{v \epsilon V} w_v 1_{\{x_v^* \geq \frac{1}{2}\}} \leq \sum_{v \epsilon V} w_v x_v^* 1_{\{x_v^* \geq \frac{1}{2}\}} = 2w(PL) \leq 2w(OPT)$$

Antes de demostrar este lema, se enunciará un lema que nos resultará útil, además de servir en contextos más generales.

Lema 3. *Si x^* es punto extremo del polítopo que define el PL, entonces: $x^* \in \{0, \frac{1}{2}, 1\}$*

Demostración. Se definen los conjuntos:

$$C^+ = \{v \in V : x_v^* \in (0, 1/2)\}$$
$$C^- = \{v \in V : x_v^* \in (1/2, 1)\}$$

Se probará que $C^+ = C^- = \emptyset$ concluyendo el resultado buscado. Para ello, como sabemos que $C^+ \cap C^- = \emptyset$, supongamos que $C^+ \cup C^- \neq \emptyset$, se define:

$$\varepsilon = \left\{ \begin{array}{l} x_v^* : \ v \in C^- \\ \frac{1}{2} - x_v^* : \ v \in C^- \\ x_v^* - \frac{1}{2} : \ v \in C^+ \\ 1 - x_v^* : \ v \in C^+ \end{array} \right\}$$

Luego, se toman los siguientes elementos en V:

$$y_v^+ = \begin{cases} x_v^* & \text{si } v \in V \setminus (C^+ \cup C^-) \\ x_v^* + \varepsilon & \text{si } v \in C^- \\ x_v^* - \varepsilon & \text{si } v \in C^+ \end{cases}$$

$$y_v^- = \begin{cases} x_v^* & \text{si } v \in V \setminus (C^+ \cup C^-) \\ x_v^* - \varepsilon & \text{si } v \in C^- \\ x_v^* + \varepsilon & \text{si } v \in C^+ \end{cases}$$

Notamos que y^-+, y^- son puntos factibles. Sea $e = ij \in E$. Si i o j están en C^-, luego, usando que $x_i^* + y_j^* \geq 1$, el otro está en C^+ o bien tiene valor 1. Por otro lado, si ambos índices están en C^+, eso implica que

$$y_i^+ + y_j^+ \geq \underbrace{(x_i^* + \varepsilon)}_{\geq \frac{1}{2}} + \underbrace{(x_j^* - \varepsilon)}_{\geq \frac{1}{2}} \geq 1$$

Esto nos permite deducir en ambos casos que $y_i^+ + y_j^- \geq x_i^* + x_j^* \geq 1$.

De lo anterior vemos que $x^* = \frac{y^+ + y^-}{2}$ *no* es extremo, lo que es una contradicción. Por lo tanto probamos que $C^+ = C^- = \emptyset$.

En conclusión, el algoritmo devuelve un cubrimiento de peso a lo más 2 veces el del óptimo.

Observación 4. *Esto funciona también para restricciones del tipo $x_i + x_j + ... + x_l \geq 1$, la cual da una n-aproximación, donde n es el número de sumandos de la restricción.*

1.2. Problema 2: Set-cover

Sean $\mathcal{C} = (c_1,...,c_m)$ una familia de conjuntos (subconjuntos de V)
Sea además $w : \mathcal{C} \longmapsto R_+$ (función de peso para cada subconjunto)
Encontrar una subfamilia de $\mathcal{C}$, digamos $\mathcal{A}$, tal que su unión cubra todo V, y que además cumpla con ser de peso mínimo.

Observación 5. *Esto generaliza el problema de Vertex-cover.*

Solución. Repitiendo la idea anterior primero lo escribimos como un Programa Entero:

$$\begin{aligned} \text{mín} \quad & \sum_{c\in\mathcal{C}} x_c w_c \\ \text{s.a.} \quad & \sum_{c:\ v\in\mathcal{C}} x_c \geq 1 \quad \forall v \in V \\ & x_c \in \{0,1\} \quad \forall c \in \mathcal{C} \end{aligned}$$

En primer lugar intentamos utilizando la técnica del **Redondeo Aleatorio**:

Algoritmo 2: Redondeo aleatorio

$x^* \leftarrow$ solución óptima del PL obtenido de relajar el Programa Entero. ($x_c \in [0,1]\ \forall c$)
$A = \emptyset$
for $i \in [k]$
  for $c \in \mathcal{C}$
    Agregar c a A con probabilidad x_c^*
return A

Ahora sólo falta analizar este algoritmo, tanto en factibilidad como optimalidad. Haremos lo segundo preguntándonos; ¿Cuál es el peso esperado de A?
Sea $w(\mathcal{A}) = \sum_{c\in\mathcal{A}} w(c)$ variable aleatoria, entonces calculamos su esperanza:

$$\begin{aligned} \mathbb{E}(w(\mathcal{A})) &= \sum_{c\in\mathcal{C}} w(c)\cdot\mathbb{P}(c\in\mathcal{A}) \\ &\leq k\sum_{c\in\mathcal{C}} w(c)x_c^* \\ &\leq k\cdot w(OPT) \end{aligned}$$

Es decir; podemos controlar el peso óptimo de nuestro algoritmo si cambiamos k. Si bien $\mathcal{A}$ es aleatorio, falta ver que con *alta probabilidad* $\mathcal{A}$ es factible para k adecuado:

Sea $v \in V$: ¿Probabilidad de que v esté cubierto por $\mathcal{A}$?

Sean c_i tal que $v \in c_i$. Como x^*, $\sum x_{c_i}^* \geq 1$, se tiene lo siguiente:

$$\mathbb{P}(v \text{ no está cubierto }) = \mathbb{P}(c_i \text{ no es elegido } \forall i)$$

Luego:

$$\begin{aligned}
\mathbb{P}(v \text{ está en } \mathcal{A}) &= 1 - \mathbb{P}(v \text{ no está en } \mathcal{A}) \\
&= 1 - \prod_{v \in c_i} \mathbb{P}(c_i \text{ no es elegido}) \\
&= 1 - \prod_{v \in c_i} (1 - x^*_{c_i})^k \\
&\geq 1 - \prod_{v \in c_i} \exp\left(-x^*_{c_i} k\right) \\
&= 1 - \exp\left(-k \sum_{v \in c_i} x^*_{c_i}\right) \\
&\geq 1 - \exp\left(-k\right)
\end{aligned}$$

Donde hemos usado que $1 - x \leq e^{-x}$ y que $\sum x^*_{c_i} \geq 1$.

Ahora, ¿Probabilidad de todos que los vértices estén cubiertos?

$$\begin{aligned}
\mathbb{P}(A \text{ factible}) &= \mathbb{P}(v \in \bigcup_{A \in \mathcal{A}} A, \, \forall v) \\
&\geq 1 - \mathbb{P}(\exists v \text{ no cubierto}) \\
&\geq 1 - \sum_{v \in V} \mathbb{P}(v \text{ no cubierto}) \\
&\geq 1 - |V| \exp\left(-k\right)
\end{aligned}$$

Ahora podemos elegir k de buena manera; por ejemplo $k \geq 2log(|V|)$ y asi:

$$1 - |V| \exp(-k) \geq \left(1 - \frac{1}{|V|}\right)$$

Observación 6. *Podemos elegir k para obtener aproximaciones de tipo $(1 - \frac{1}{|V|^l})$ con l a eleccion; eligiendo $k \geq (l + 1)\log(|V|)$. Con esto el peso esperado es $\leq (l+1)\log(|V|) \cdot OPT$. Luego, el factor esperado de aproximación es $O(\log |V|)$, factor que, salvo casos particulares, es lo mejor que se conoce para set-cover.*

1.3. Problema 3: Corte de Peso Máximo

Dado $G = (V, E), w : V \longmapsto \mathbb{R}_+$
Encontrar $U \subseteq V$ tal que $w(\delta(U))$ sea máximo.

Observación 7. *Este problema es NP-Difícil.*

Solución. De manera similar al problema anterior, se plantea un algoritmo utilizando redondeo aleatorio. Se verá que éste algoritmo se puede mejorar trabajando de forma iterativa.

Algoritmo 3:
Elegir U del siguiente modo: $\forall e = ij$, $\mathbb{P}(e \text{ esté cortada}) = \frac{1}{2}$ **return** U

Al igual que en el caso anterior, el corte obtenido es *probablemente* una buena aproximación, para probarlo se calcula la esperanza de la variable aleatoria:

$$\begin{aligned}\mathbb{E}[w(\delta(U))] &= \sum_{e\in E} w_e \mathbb{P}(e \in \delta(U)) \\ &= \frac{1}{2} w(E) \\ &\geq \frac{1}{2} w(OPT)\end{aligned}$$

Por lo tanto este algoritmo devuelve un corte cuyo peso esperado es mayor o igual a $\frac{1}{2}w(OPT)$

Método de esperanzas condicionales

Este método *desaleatoriza* la elección del conjunto U, proponiendo un método de aproximación iterativo para el problema.

Dado un conjunto de vértices $\{v_1, v_2, \ldots, v_n\}$ se define la siguiente función:

$$f(x_1, \ldots, x_n) = w[\delta(U)]$$

donde $U = \{i : x_i = 1\}$.

Se sigue el siguiente procedimiento:

Algoritmo 4:

for $i \in [n]$
Elegir $s_i \in \{0, 1\}$ que maximice $\mathbb{E}[f(s_1, \ldots, s_{i-1}, s_i, x_{i+1}, \ldots, x_n)] = \mathbb{E}[f(x)|x_1 = s_1, \ldots, x_i = s_i)]$.
return s

Notar que es posible usando la misma idea, resolver el problema cuando algunos de los vértices ya están fijados de antemano.