MA4701. Optimización Combinatorial. 2013.
Profesor: José Soto
Escriba(s): Juan Granier, Juan Pablo Contreras y José Coronado
Fecha: 29 de Noviembre de 2013.

Cátedra 25

Reducción de Problemas (Lenguajes)

Dados dos lenguajes $L_1,\ L_2 \subseteq \Sigma^*$ denotamos por $L_1 \leq_p L_2$ (y leemos "L_1 se reduce a L_2") siempre que exista un algoritmo polinomial A que calcula $f : \Sigma^* \to \Sigma^*$ que satisface

$$x \in L_1 \Leftrightarrow f(x) \in L_2.$$

Intuitivamente que un lenguaje L_1 pueda reducirse a un lenguaje L_2 quiere decir que L_1 es una problema más fácil que L_2. Si sabemos resolver L_2 entonces podemos resolver L_1 pues podemos ver la entrada de L_1 a través de la función f obteniendo una entrada para L_2.
En resumen si existe algoritmo polinomial para decidir L_2 entonces tenemos algoritmo polinomial para decidir L_1. A continuación veremos un ejemplo de lenguajes que pueden reducirse en otros

Ejemplo 1. Veremos una relación entre los problemas HAM-PATH y HAM-CYCLE que corresponden a decidir si existe un camino Hamiltoniano y de un ciclo Hamiltoniano en un grafo respectivamente. Definamos

$$\text{HAM-PATH} = \{\langle G\rangle \ :\ \text{G contiene un camino Hamiltoniano}\}$$

$$\text{HAM-CYCLE} = \{\langle G\rangle \ :\ \text{G contiene un ciclo Hamiltoniano}\}$$

Recordemos que un camino o ciclo Hamiltoniano es un camino o ciclo que pasa por todos los vértices una vez. A continuación demostraremos que HAM-CYCLE $\leq_p$ HAM-PATH sin embargo la desigualdad también vale en el sentido opuesto.

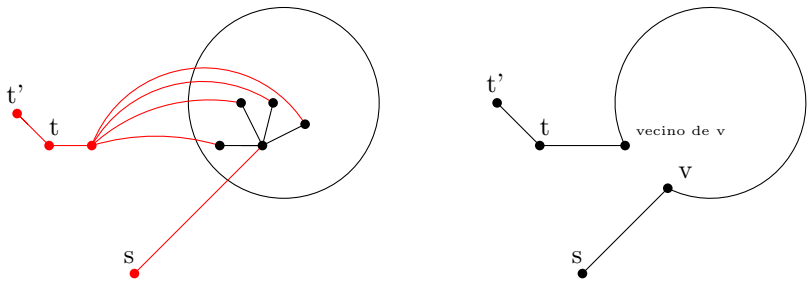

Figura 1: Conversión de grafos

Dado $G = (V, E)$ grafo construimos un nuevo grafo que llamaremos H_G de la siguiente manera: En el grafo G agregamos 3 vértices nuevos $\{s, t, t'\}$. Elegimos un vértice arbitrario $v \in V$, conectamos s con v y a cada $u \in N_G(v)$ lo conectamos con t. Finalmente conectamos t' con t.

Como $d(s) = d(t') = 1$ si existe camino Hamiltoniano en H_G este debe tener como extremos a s y t' y si ut es arista del camino entonces, como u es vecino de v, podemos encontrar un ciclo Hamiltoniano en G que parte en v usa las aristas del camino, llega a u y vuelve a v. Con esto es fácil deducir que H_G tiene camino Hamiltoniano si y sólo si G tiene ciclo Hamiltoniano.

Definición 1. Un lenguaje L es NP-difícil si y solo si $\forall\ L' \in \text{NP}\ \ L' \leq_p L$.

En general es sencillo crear problemas que sean NP-difíciles, lo que resulta más interesante es otra categoría que definimos a continuación

Definición 2. Un lenguaje L es NP-completo si y solo si $L \in \text{NP}$ y L es NP-difícil.

L es NP-completo si no se puede resolver en tiempo polinomial pero sus soluciones si se pueden verificar en tiempo polinomial. Con el fin de darle sentido a esta última definición enunciaremos, sin demostración, un teorema que nos asegura que existen problemas NP-completos

Teorema 1. *Cook-Levin*
SAT (satisfacción) es NP-Completo

Aunque no demostraremos el teorema anterior debido a su complejidad, veremos lo que es SAT.
Podemos pensar una sentencia lógica dependiente de variables booleanas $x_1,\ x_2, \cdots,\ x_k$ como una función $\phi : \{0,1\}^k \to \{0,1\}$ que recibe el nombre de función booleana. Definimos SAT como

$$\text{SAT} = \{\langle\phi\rangle :\ \phi \text{ es booleana y existe } x = (x_1,\ x_2, \cdots,\ x_k) \in \{0,1\}^k \text{ con } \phi(x) = 1\}$$

Ejemplo 2.

$$\langle (x_1 \wedge x_2) \vee (\ x_3 \wedge x_1)\rangle\ \in \text{SAT}$$

En efecto, la asignación $x_1 = x_2 = 1$ y $x_3 = 0$ hace que la sentencia sea valida. Por otro lado

$$\langle x_1 \wedge\ x_1\rangle\ \notin \text{SAT}$$

Dada una sentencia lógica o función booleana es muy difícil decidir si está o no en SAT, sin embargo si nos dan una asignación $(x_1,\ x_2, \cdots,\ x_k)$ es fácil comprobar la validez de la sentencia.

Observación 1. Si SAT $\leq_p L$ entonces L es NP-difícil por transitividad de la relación $\leq_p$.

Observación 2.

$$\text{SAT} \leq_p \begin{cases} \textsc{Ham-Cycle} \\ \textsc{Ham-Path} \\ \textsc{Knapsack} \\ \textsc{Vertex-Cover} \\ \textit{etc.} \end{cases}$$

Son muchos los problemas que se cree que no se pueden resolver en tiempo polinomial.
Con vista a la optimización veamos que todo problema del estilo

$$\text{Max } w(S)\ :\ S \in OBJ$$

donde OBJ es una familia de objetos dependientes de G se transforma a problemas de decisión

$$L = \{\langle G, k\rangle\ :\ \exists S \in OBJ,\ w(S) \geq k\}$$

Ejemplo 3. (Corte máximo)
Definimos $w(S) = |\delta(S)|$ y $L = \{\langle G, k\rangle\ :\ G$ tiene corte con más de k aristas $\}$.
Si podemos encontrar un algoritmo polinomial que decida L, podemos variar o bien hacer búsqueda binaria en k para obtener un algoritmo que resuelva en problema del corte máximo. Notemos además que la reciproca tambien se tiene, es decir, si se conoce un algoritmo para corte máximo este se puede usar para decidir L

Los problemas de optimización asociados a problemas de decisión NP-completos son difíciles de resolver a priori, sin embargo, debido a la gran cantidad y a su importancia es necesaria una forma de abordarlos. Este problemática motiva el estudio de algoritmos de aproximación

Definición 3. Un Algoritmo de aproximación polinomial para el problema

$$\max\{w(S)\ :\ S \in OBJ\}$$

es un algoritmo polinomial que encuentra un objeto S tal que $w(S) \geq \frac{w(OPT)}{f(n)}$ donde OPT es un objeto óptimo del problema.
La función $f(n)$ se conoce como factor de aproximación.

Analogamente para minimización queremos encontrar $S \in OBJ$ tal que $w(S) \leq f(n)w(OPT)$. En ambos casos minimización y maximización $f(n) \geq 1$, si $f(n) = 1$ el algoritmo es exacto. La idea es determinar un algoritmo con $f(n)$ lo más cercano a 1 posible.

Ejemplo 4. VERTEX-COVER (versión optimización)
Dado G grafo encontrar $U \subseteq V(G)$ que *toca* todas las aristas y que sea de cardinalidad mínima.

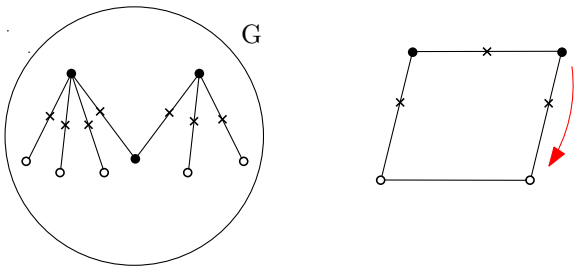

Figura 2: Nodos que aparecen en negro representan al cubrimiento de vértices.

Recordemos que si G es bipartito el problema puede resolverse buscando un Matching de cardinalidad máxima en tiempo polinomial, pero en general el problema es NP-difícil (NP-completo).
A continuación mostramos unas ideas para formar un algoritmo de aproximación y veremos porque algunas pueden resultar intuitivamente mejores que otras

- (Glotón) Elegir en cada iteración al vértice que cubre la mayor cantidad de aristas no cubiertas y agregarlo al cubrimiento. *Ejercicio*: Calcule el factor de aproximación en este caso
- Construir un emparejamiento (matching) M maximal para inclusión. Sean S los extremos de M entonces S es cubrimiento por vértices (si existiese $e \in E$ no cubierto entonces $M + e$ es matching $\longrightarrow\!\!\!\!\!\!\leftarrow$).
  Si llamamos OPT al mínimo cubrimiento por vértices se tiene

  $$|S| \leq 2|M| \leq 2|OPT|$$

  pues recordemos que cualquier matching tiene cardinal menor o igual que cualquier cubrimiento por vértices. Esta última desigualdad nos dice que el algoritmo anterior es una 2 aproximación para VERTEX-COVER

Una modificación a este problema consiste colocar pesos a los vértices y buscar cubrimiento de por vértices de peso mínimo. En este caso no ayuda un algoritmo como glotón ya que podrían existir vértices de poco peso pero con poca aristas incidentes.

Ejemplo 5. VERTEX-COVER de peso mínimo.
Dado G grafo y $w : V(G) \to [0, \infty)$ encontrar $S \subseteq V(G)$ cubrimiento por vértices de peso mínimo. Podemos ver este problema con un programa entero

$$\begin{array}{lll} \text{Min} & w^T x & \\ s.a. & x_v + x_u \geq 1 & \forall e = uv \in E \\ & x_v \in \{0, 1\} & \end{array}$$

Esto no resuelve el problema ya que programación entera es NP-difícil. (Casi todos los problemas pueden escribirse como programación entera). Es bueno, sin embargo, ver que pasa con la relajación del problema obteniendo un problema de programación lineal

$$\begin{array}{lll} \text{Min} & w^T x & \\ s.a. & x_v + x_u \geq 1 & \forall e = uv \in E \\ & 0 \leq x_v \leq 1 & \forall v \in V \end{array}$$

El algoritmo del elipsoide nos asegura que este programa lineal puede resolverse en tiempo polinomial y más aún se puede obtener una solución optima que sea un punto extremo. Designemos por x^* al punto extremo que es optimo.

Teorema 2. *Si x^* es optimo extremo entonces $x_v^* \in \{0, \frac{1}{2}, 1\}$ para todo $v \in V$.*

En efecto, procedamos por contradicción. Si no fuese así definamos 2 conjuntos

$$V_-(x^*) = \{v \in V : \ x_v^* \in (0, \frac{1}{2})\} \qquad V_+(x^*) = \{v \in V : \ x_v^* \in (\frac{1}{2}, 1)\}$$

de la suposición se tiene $V_-(x^*) \cup V_+(x^*) \neq \emptyset$.
Definamos

$$\epsilon_1 = \text{mín}\{x_v^* : v \in V_-(x^*)\} \qquad \epsilon_2 = \text{mín}\{1 - x_v^* : v \in V_+(x^*)\}$$

$$\epsilon_3 = \text{mín}\{x_v^* - \frac{1}{2} : v \in V_+(x^*)\} \qquad \epsilon_4 = \text{mín}\{\frac{1}{2} - x_v^* : v \in V_-(x^*)\}$$

$$\epsilon = \text{mín}\{\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\} > 0$$

Sea además

$$y_v^{\pm} = \begin{cases} x_v^* & \text{si} \quad v \notin V_-(x^*) \cup V_+(x^*) \\ x_v^* \pm \epsilon & \text{si} \quad v \in V_+(x^*) \\ x_v^* \mp \epsilon & \text{si} \quad v \in V_-(x^*) \end{cases}$$

Entonces se tiene

$$x^* = \frac{y^+ + y^-}{2}.$$

Basta mostrar que y^- e y^+ son factibles, contradiciendo el hecho de que x^* es extremo.
Sea e una arista del grafo con alguno de sus extremos, digamos v, en $V_-(x^*) \cup V_+(x^*)$. Si $v \in V_-(x^*)$ entonces por la forma en la que definimos ϵ se tendrá $v \in V_-(y^-)$ y $v \in V_-(y^+)$. De igual forma si $v \in V_+(x^*)$ se tiene $v \in V_+(y^-)$ y $v \in V_+(y^+)$. Además si una arista no tiene extremos en $V_-(x^*)$ entonces está cubierta por y^+ y por y^-, mientras que si uno de sus extremos está en $V_-(x^*)$ el otro debe estar en $V_+(x^*)$ (pues x^* es factible) y en este caso ambas soluciones, y^+ e y^- mantienen constante la suma del valor de ambos extremos ya que se suma y resta ϵ.

Podemos obtener un cubrimiento mínimo entregando $S = \{v : x_v^* \geq \frac{1}{2}\}$. Probaremos ahora que este algoritmo es una 2 aproximación
Sean S el conjunto obtenido por este procedimiento, OPT el mejor cubrimiento por vértices y x^* la solución fraccional encontrada entonces

- S es factible
- Es válida la siguiente cadena de desigualdades

  $$w(S) \leq 2w^T x^* \leq 2w(OPT),$$

  esto pues

  $$\sum_{v \in V} w_v \leq \sum_{v \in V} w_v x_v^*$$

  y

  $$w^T x^* \leq w(OPT).$$

  Note que el primero es el óptimo del programa lineal (problema relajado) y el segundo el óptimo del programa entero
  Así

  $$w(S) \leq 2w(OPT).$$