Ma50b Complejidad Computacional	**14 de Mayo de 2009**

Pauta Control 2

Prof. Cátedra: M. Kiwi — *Prof. Auxiliar: T. González, I. Fantini*

PROBLEMA 1:

(i).- Para ver que QuadraticEq está en NP basta considerar el certificado $u \in \mathbb{F}_2^n$ y el proceso de verificación que consiste en comprobar que las siguientes igualdades se cumplen:

$$\sum_{i,j=1}^{n} A_{i,j}^{(k)} u_i u_j = b_k\,, \quad k = 1, \ldots, m.$$

Para establecer que QuadraticEq es NP-duro basta probar que 3SAT $\leq_m^P$ QuadraticEq. En efecto, sea $\langle \phi \rangle$ tal que $\phi = \wedge_{i=1}^{m} C_i$ es una formula Boolean en forma 3CNF en las variables $x_1, \ldots, x_n$. Sin pérdida de generalidad asumimos que $C_i = x_{i_1}^{e_{i,1}} \vee x_{i_2}^{e_{i,2}} \vee x_{i_3}^{e_{i,3}}$ donde x^e denota x si $e = 1$, y $\bar{x}$ si $e = 0$. A la clausula C_i le asociamos las siguientes ecuaciones cuadráticas en $\mathbb{F}_2$ en las indeterminadas $x = (x_i : i = 1, \ldots, n)$ e $y = (y_i : i = 1, \ldots, m)$,

$$y_i = (e_{i,1} - x_{i_1})(e_{i,2} - x_{i_2})\,, \tag{1}$$
$$0 = y_i(e_{i,3} - x_{i_3})\,. \tag{2}$$

Notar que, para x dado, existe un y_i tal que x e y_i satisfacen ambas ecuaciones simultáneamente si y solo si $C_i(x) = 1$. Observando que en $\mathbb{F}_2$ se tiene que $z^2 = z$, sigue que podemos asociar a (1) y (1) matrices $A^{(2i-1)}, A^{(2i)} \in \mathbb{F}_2^{n' \times n'}$, $n' = n + 1m$, y $b_{2i-1}, b_{2i} \in \mathbb{F}_2$ tales que

$$\exists x \in \mathbb{F}_2^n\,,\ \forall i \in \{1, \ldots, m\}\,,\ C_i(x) = 1 \quad \Longleftrightarrow \quad \exists u \in \mathbb{F}_2^{n'}\,,\ \forall k \in \{1, \ldots, m\}\,,\ \sum_{i,j=1}^{n'} A_{i,j}^{(k)} u_i u_j = b\,.$$

Luego, $\langle \phi \rangle \in$ 3SAT si y solo si $\langle A^{(1)}, \ldots, A^{(2m)}, b \rangle \in$ QuadraticEq.

Como es fácil asociarle a cada clausula C_i el par de ecuaciones más arriba indicada, y reordenar los términos correspondientes, se pueden generar eficientemente las matrices $A^{(2i-1)}$ y $A^{(2i)}$ y las coordenadas b_{2i-1} y b_{2i}. Sigue que el cálculo de $\langle A_1, \ldots, A_{2m}, b \rangle$ a partir de $\langle \phi \rangle$ se puede hacer en tiempo polinomial. En resumen, 3SAT $\leq_m^P$ QuadraticEq.

(ii).- La pertenecia de FeedbackArc a NP se obtiene considerando el certificado $F \subseteq E$ y verificando que $|F| \leq k$ y que $G \setminus F$ no posee cíclos (esto último se puede hacer eficientemente realizando $|V(G)|$ búsquedas horizontales en $G \setminus F$, cada búsqueda partiendo de un nodo distinto de $G \setminus F$).

Veamos ahora que VertexCover $\leq_m^P$ FeedbackArc. Consideremos una instancia $\langle G, k \rangle$ de VertexCover donde $G = (V, E)$ es un grafo (no-dirigido). Construimos $G' = (V', E')$ digrafo de tal manera que cada nodo de v en V da lugar a dos nodos v_{tail} y v_{head} en V' y un arco $v_{tail} v_{head}$ en E'. Además, cada arco $e = uv$ en E da lugar a dos arcos $u_{head} v_{tail}$ y $v_{head} u_{tail}$ en E'. La Figura 1 ilustra la construcción recién descrita. Dada la característica local de la construcción de G' a partir de G, sigue facilmente que esta se puede realizar eficientemente (en tiempo polinomial).

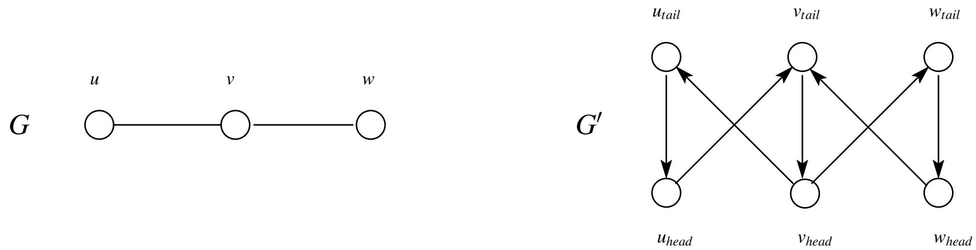

Figura 1: Grafo G y su grafo G' asociado.

Afirmamos que $\langle G,k\rangle \in$ VertexCover si y solo si $\langle G',k\rangle \in$ FeedbackArc.

En efecto, sea S un conjunto de nodos de $G=(V,E)$ que recubre los arcos de G. Sea $F=\{v_{tail}v_{head} : v\in S\}$. Claramente, $|F|=|S|$. Supongamos que existe un circuito $C=v^{(0)}\dots v^{(\ell-1)}$ en $G'\setminus F$ y tomemos el circuito para el que ℓ es mínimo. Por la estructura de G' se debe tener que ℓ es par, digamos $\ell=2\ell'$. Como no hay ciclos de largo 2 en G', necesariamente se tendrá que $\ell\geq 4$. Además, por la forma en que se construyó G', los nodos en C deben irse alternando entre los nodos tipo *tail* y los tipo *head*. Sin pérdida de generalidad podemos suponer que $C=v_{tail}^{(0)}v_{head}^{(0)}v_{tail}^{(1)}v_{head}^{(1)}\dots v_{tail}^{(\ell')}v_{head}^{(\ell')}$. Como $v_{head}^{(0)}v_{tail}^{(1)}$ está en G', el arco $v^{(0)}v^{(1)}$ necesariamente está en G, y por lo tanto $v_{head}^{(1)}v_{tail}^{(0)}$ está en G'. Luego, $v_{tail}^{(0)}v_{head}^{(0)}v_{tail}^{(1)}v_{head}^{(1)}$ es un cíclo en $G'\setminus F$. Sigue que $v^{(0)},v^{(1)}\notin S$ y que $v^{(0)}v^{(1)}\in E$, lo que contradice que S es un recubrimiento de G.

Supongamos ahora que F es tal que $G'\setminus F$ es un digrafo acíclico. Sea S el subconjunto de nodos $v\in V$ tal que $v_{tail}v_{head}$ está en F o para algún $u\in V$ se tiene que $u_{head}v_{tail}$ está en F. Claramente, $|S|\leq|F|$. Si S no fuera un recubrimiento de G, entonces existiría un arco uv de G tal que $u,v\notin S$. Por definición de S, sigue que $u_{head}v_{tail}, v_{tail}v_{head}, v_{head}u_{tail}, u_{tail}u_{head}$ son arcos en $G'\setminus F$, contradiciendo el hecho que $G'\setminus F$ es acíclico.

En resumen, VertexCover $\leq_m^P$ FeedbackArc.

PROBLEMA 2:

(i).- Sea $L\in$ P tal que M es un máquina de Turing a tiempo polinomial $p(n)$ que decide L. Sin pérdida de generalidad podemos asumir que M tiene una sola cinta y que si acepta lo hace con su cinta en blanco y en el estado de aceptación q_{acep}. Sean Σ y Q el alfabeto y conjunto de estados de M respectivamente. Sea s el estado de partida de M.

La idea de la demostración es considerar una tabla T de cálculo de M en una entrada ω, $|\omega|=m$. Dicha tabla la podemos ver como un arreglo de $p(m)\times p(m)$ celdas. Cada fila corresponde a las $p(m)$ primeras celdas de la cinta de M. Si en el instante t la máquina M se encuentra en el estado q y su cabeza lectora está sobre la j-ésima celda de su cinta leyendo α, entonces decimos que el contenido de $T_{t,j}$ es (q,α). Si en el instante t la j-ésima celda de la cinta de M contine α y la cabeza lectora no se encuentra sobre dicha celda, entonces decimos que el contenido de $T_{t,j}$ es α. Construiremos un circuito Booleano C_m en m entradas que le asocia a la celda $T_{i,j}$ los nodos $P_{i,j}^{a}$ con $a\in\Sigma_{\not b}$ y nodos $Q_{i,j}^{q}$ con $q\in Q$. Cada uno de estos nodos corresponderá a una puerta lógica que se calcula mediante un circuito Booleano a partir de las puertas lógicas asociadas a las celdas $T_{i-1,j-1},T_{i-1,j},T_{i-1,j+1}$ de T. Usando la convención que $P_{i,0}^{a}$ y $P_{i,p(n)}^{a}$ denotan 0, construimos C_m de forma que si $1 < i \leq p(m)$ y $1 \leq j \leq p(m)$, entonces

$$
\begin{aligned}
P^{\beta}_{i,j} &= \bigvee_{p,\alpha,q:\delta(p,\alpha)=(q,\beta,D)} \left(Q^{p}_{i-1,j} \wedge P^{\alpha}_{i-1,j}\right) \vee \left(P^{\beta}_{i-1,j} \wedge \bigwedge_{p\in Q} \neg Q^{p}_{i-1,j}\right), \\
Q^{q}_{i,j} &= \bigvee_{p,\alpha,q:\delta(p,\alpha)=(q,\beta,R)} \left(Q^{p}_{i-1,j-1} \wedge P^{\alpha}_{i-1,j-1}\right) \vee \bigvee_{p,\alpha,q:\delta(p,\alpha)=(q,\beta,L)} \left(Q^{p}_{i-1,j+1} \wedge P^{\alpha}_{i-1,j+1}\right).
\end{aligned}
$$

Además, hacemos $Q^{q}_{1,j} = 1$ si $(q,j) = (s,1)$ y 0 en caso contrario, $P^{1}_{1,j} = \omega_j$ y $P^{0}_{1,j} = \overline{\omega_j}$ si $j < m$, y si $j > m$, entonces $P^{\alpha}_{1,j} = 1$ si $\alpha = \not b$ y 0 en caso contrario. Por último tomamos como puerta de salida de C_m al nodo $Q^{q_{acep}}_{p(m),1}$. Se verifica que C_m es un circuito Booleano en las entradas $\omega = \omega_1\omega_2\ldots\omega_m$ que tiene $O(p^2(m))$ puertas lógicas. Además, se tiene que si $T_{i,j} = \alpha \in \Sigma_{\not b}$, entonces $P^{\alpha}_{i,j} = 1$ y $Q^{q}_{i,j} = 0$, y si $T_{i,j} = (q,\alpha)$, entonces $P^{\alpha}_{i,j} = 1$ y $Q^{q}_{i,j} = 1$. Sigue que $\omega \in L$ si y solo si $C_{|\omega|}(\omega) = 1$. Se concluye que si $L \in$ P, entonces existe una familia de circuitos Booleanos de tamaño polinomial que decide L.

(ii).- Supongamos que $L \in$ P/poli. Por definición, existe una máquina de Turing M a tiempo polinomial, un polinomio p y una secuencia $(\alpha_n : n \in \mathbb{N})$, $\alpha_n \in \{0,1\}^*$ donde $|\alpha_n| \leq p(n)$, tal que para todo $\omega \in \{0,1\}^n$ se tiene que $\omega \in L$ si y solo si M acepta $\langle \omega, \alpha_n \rangle$. Observar que $L_M \in$ P, luego por (i) concluimos que existe una familia de circuitos $(D_m : m \geq 1)$ de tamaño polinomial, digamos $q(m)$, que decide L_M. Sin pérdida de generalidad, podemos asumir que q es monótono no-decreciente. Sea $m = |\langle 0^n, \alpha_n \rangle|$ y C_n el circuito que se obtiene a partir de D_m al fijar el valor de α_n en las entradas correspondientes. Es claro que C_n es un circuito Booleano de n entradas de tamaño igual al de D_m. Sigue que $|C_n| = |D_m| \leq q(m) = q(O(n + p(n)))$, luego $(C_n : n \geq 1)$ es una familia de circuitos Booleanos de tamaño polinomial en n. Se verifica facilmente que $(C_n : n \geq 1)$ decide L.

Supongamos ahora que existe una familia de circuitos Booleanos $(C_n : n \geq 1)$ de tamaño polinomial que decide L. Sea p un polinomio tal que $|C_n| \leq p(n)$. Sea $\alpha_n = \langle C_n \rangle$. Dado que cualquier codificación razonable de C_n tendrá un largo polinomialmente relacionado con el tamaño de C_n se tiene que existe un polinomio q tal que $|\alpha_n| \leq q(n)$ (de hecho, basta tomar $q(n) = p^2(n)$). Sea M la máquina de Turing que en la entrada $\langle \omega, \alpha_n \rangle$ con $\omega \in \{0,1\}^n$, evalua en ω el circuito codificado por α_n, acepta si la evaluación da 1, y rechaza en caso contrario. Sigue que M acepta $\langle \omega, \alpha_n \rangle$ si y solo si $C_n(\omega) = 1$. Luego, M decide L. Además, M es a tiempo polinomial en n dado que la evaluación de C_n toma tiempo polinomial en $p(n)$ (el tamaño de C_n). En resumen, $L \in$ P/poli.

(iii).- Lo natural sería considerar un lenguaje NP-completo dado que son los más difíciles en NP. Por lo demás, no es difícil ver que P/poli es cerrado bajo reducciones mucho-a-uno a tiempo polinomial, por lo que si algún lenguaje NP-completo estuviese en P/poli se tendría que NP $\subseteq$ P/poli.

PROBLEMA 3:

(i).- Sea $\langle \phi \rangle$ tal que ϕ es un fórmula Booleana en las variables $x_1, \ldots, x_n$. Notar que existe $a \in \{0,1\}^n$ tal que $\phi(a) = 1$ si y solo si existe $a' \in \{0,1\}^{n-1}$ tal que $\phi(1,a') = 1$ o $\phi(0,a') = 1$. Equivalentemente, $\langle \phi \rangle \in$ SAT si y solo si $\langle \phi|_{x_1=1} \rangle \in$ SAT o $\langle \phi|_{x_1=0} \rangle \in$ SAT. Sigue que para decidir si $\langle \phi \rangle \in$ SAT basta hacer dos consultas sobre la membresía de $\langle \phi|_{x_1=1} \rangle$ y $\langle \phi|_{x_1=0} \rangle$ en SAT. Observar además, que para cualquier codificación natural de fórmulas Booleanas, al instanciar una variable de la fórmula en 0 o 1 se obtienen fórmulas cuya codificación tienen un largo menor. Luego, $|\langle \phi|_{x_1=1} \rangle|, |\langle \phi|_{x_1=0} \rangle| \leq |\langle \phi \rangle|$. En resumen, SAT es autoreducible hacia abajo.

(ii).- Primero probaremos la afirmación que se hace en la indicación. En efecto, si hay $n+1$ nodos en $Izq \cup Cam$ que toman el mismo valor, entonces no todos ellos podrían estar en Cam, puesto que cualquier camino de la raíz a una hoja de T tiene a lo más n nodos internos. Sea $v \in Izq$ tal que su color es compartido por al menos otros n nodos en $Izq \cup Cam$. Por definición de Izq, el nodo v no puede ser antecesor de una hoja de T donde c tome el valor $acept$. Luego, $c(v) \notin c(A)$ como se quería demostar.

El algoritmo solicitado consiste en hacer una búsqueda en profundidad en T, barriendo T hacia la "derecha" partiendo desde su hoja de más a la "izquierda". Para cada nodo interno v visitado, se calcula $c(v)$ y se mantiene un contador con el número de veces que se ha encontrado cada color. En la búsqueda en profundidad, se ignoran todos los descendientes de aquellos nodos cuyo color haya previamente aparecido $n+1$ veces. Cada vez que se visita una hoja h se calcúla el valor que toma c en ella, y en caso de ser igual a $acept$ el algoritmo retorna h y para. Si la búsqueda termina sin que se encuentre una hoja h tal que $c(h) = acept$, entonces el algoritmo retorna que tal hoja no existe.

Por la indicación, sigue que el algoritmo descrito en el párrafo anterior visita a lo más $(n+1)(p(n)+1)$ nodos internos de T y por lo tanto un número menor de hojas de T. Por cada nodo de T visitado, el procedimiento requiere hacer cálculos que toman tiempo polinomial en n. En total, el algoritmo toma tiempo polinomial en n.

(iii).- Sea U un lenguaje unario NP-completo. Se tiene que SAT mucho-a-uno reduce en tiempo polinomial a U. Sea R la reducción, calculable en tiempo polinomial, digamos p, que lleva instancias de SAT en instancias de U. Notar que R le asocia a $\langle\phi\rangle$, donde ϕ es un fórmula Booleana, un elemento de 1^* de largo a lo más $p(|\langle\phi\rangle|)$. Luego, si $n = |\langle\phi\rangle|$ se tiene que $0 \leq |R(\langle\phi\rangle)| \leq p(n)$. Sea T un árbol que se define recursivamente como sigue: la raíz de T es un nodo asociado a ϕ, si un nodo esta asociado a $\phi|_{x_i=a_i, i=1,\ldots,j}$ entonces su hijo izquierdo y derecho estan asociados a $\phi|_{x_i=a_i, i=1,\ldots,j; x_{j+1}=1}$ y $\phi|_{x_i=a_i, i=1,\ldots,j; x_{j+1}=0}$ respectivamente. Notar que la profundidad de T es igual al número de variables de ϕ, que a su vez está acotado por $n = |\langle\phi\rangle|$. Al nodo interno de T asociado a $\phi|_{x_i=a_i, i=1,\ldots,j}$ le asignamos el color $|R(\langle\phi|_{x_i=a_i, i=1,\ldots,j}\rangle)|$. A cada hoja de T asociada a $\phi|_{x_i=a_i, i=1,\ldots}$ le asignamos el valor $acept$ o $rech$ dependiendo de si $\phi(a_1, a_2, \ldots)$ toma el valor 1 o 0 respectivamente. Es fácil verificar que se satisfacen todas las condiciones para aplicar el algoritmo de la parte (ii), y por lo tanto en la entrada $\langle\phi\rangle$ se puede, en tiempo polinomial en $|\langle\phi\rangle|$, ya sea encontrar una asignación de valores de verdad que satisfaga ϕ o determinar que tal asignación no existe. Sigue que SAT $\in$ P lo que a su vez implica que P $=$ NP como se quería demostrar.