$G(\phi)$ formado por m triángulos etiquetados con los literales de cada cláusula, y unimos cada vértice etiquetado con x con los etiquetados con $\neg x$.

- $G(\phi)$ contiene un conjunto de vértices independientes de tamaño m sii ϕ es satisfactible: el conjunto está formado por un literal cierto de cada cláusula de ϕ bajo σ que la hace cierta.

- El problema sigue siendo NP-completo restringido a grafos de grado 4, que son aquéllos en los que en cada nodo inciden a lo sumo 4 arcos

Dem: Consideramos (3 SAT, 3, 2) y repetimos la construcción anterior generando aristas, en lugar de triángulos, para las cláusulas con 2 literales.

Cliqués y cubrimientos con nodos

- Cliqué subgrafo completo con k nodos
- Cubrimiento: conjunto de nodos que contiene al menos un vértice de cada arco.
- CLIQUÉ es NP-completo

  Demost: $G=(V,E)$ tiene un cliqué de tamaño k sii $\bar{G}=(V, V^2-E)$ tiene un conjunto independiente con k nodos.

- CUBRIMIENTO es NP-completo

  Demost: I es independiente en G sii $V-I$ es cubrimiento de G.

Cortes bipartitos de un grafo

- Corte de G es toda partición de V en dos subconjuntos (no vacíos) S y $V-S$.
- Tamaño de un corte = nº de aristas que unen los dos lados del corte
- MIN CUT: corte mínimal de G. Polinomial utilizando MAXIMO FLUJO.
- MAX CUT: ¿existe un corte de tamaño k o mayor?

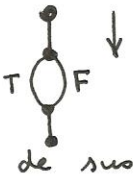
— MAX CUT es NP-completo: reduciendo NAESAT a él.

- En realidad consideraremos el problema sobre multigrafos, que pueden tener arcos repetidos, contando cada uno por separado en los cortes.
- Si $Var(\phi) = \{x_1, \ldots, x_m\}$ $G(\phi)$ contiene $2m$ nodos correspondientes a los literales afirmados y negados.
- En un triángulo el máximo corte tiene tamaño 2 y se obtiene para cualquier corte posible del mismo
- Para cada $C_i = \alpha \vee \beta \vee \gamma$ de ϕ introducimos en $G(\phi)$ el triángulo correspondiente, excepto los posibles lados entre literales idénticos. Además añadimos tantos arcos entre x_i y $\neg x_i$ como veces aparecen en total dichos literales en ϕ.

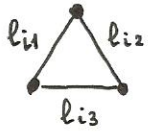
- $G(\phi)$ tiene un corte de tamaño $5m$, siendo $\phi = \bigwedge_{i=1}^{m} C_i$ sii ϕ es NAESAT - satisfactible.
  - En un corte maximal nunca están x_i y $\neg x_i$ en el mismo lado
  - Sea $(S, V-S)$ un corte de tamaño mayor o igual a $5m$. De entre ellos $3m$ corresponden a los multiarcos $(x_i, \neg x_i)$. Los otros $2m$ arcos provienen de los triángulos, por lo que todos han de tener al menos un vértice a cada lado. Tomamos ciertos los literales a un lado y falsos los otros.
  - El recíproco es análogo.
- MAX BISECTION sigue siendo NP-completo: imponemos $|S| = |V-S|$.
  - Añadimos otros $|V|$ vértices sueltos a G, ¡que pueden mandarse al lado que queramos del corte S sin alterar su tamaño!
  - ¡En realidad ya lo teníamos probado antes pues el corte maximal de $G(\phi)$ era una bisección!
- MIN BISECTION, ¡pese a ser NP-completo!
  - Si $|V| = 2n$, G tiene una bisección de tamaño k o menor sii $\bar{G}$ la tiene de tamaño $n^2 - k$ o mayor.
  - ¡Típico ejemplo de aplicación de la dualidad!

Caminos hamiltonianos (en grafos no-dirigidos)

- HAMILTON PATH es NP-completo, reduciendo a él directamente 3 SAT.
  - $\phi = \bigwedge_{i=1}^{m} C_m$ con $Var(\phi) = \{x_1, \dots, x_m\}$
  - $G(\phi)$ está formado por una serie de componentes con vértices externos ● e internos ○. Las componentes se ensamblan compartiendo los primeros
  - Componente de elección para valorar una variable

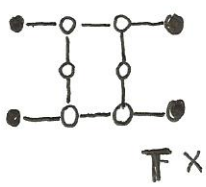


  - Triángulos para representar las cláusulas por medio de sus lados



  - Componentes para valorar los literales de forma consistente

$l_{ij} = x_2$ Establece el OR exclusivo entre ambas aristas superior e inferior: dos posibles recorridos



- En el recorrido tendremos que recorrer los literales falsos de cada triángulo, lo que anula el carácter hamiltoniano del camino si en una cláusula todos son falsos
- Ensamblamos las piezas comenzando por la valoración secuencial de todas las variables. Luego construimos un dique con el vertice final de la valoración, todos los vértices de los triángulos y un nodo auxiliar más al que se le pega el vértice final del recorrido
- Cuando un literal aparece repetido la componente correspondiente a la valoración de la correspondiente variable se hace más larga para empalmar en ella los segmentos correspondientes a las repetidas ocurrencias del literal

Prop: ϕ satisfactible sii $G(\phi)$ contiene un camino hamiltoniano

Demost: Hemos de pasar por los nodos introducidos en cada arista del triangulo para garantizar la consistencia lo que se puede hacer para aquellas ciertas (se hace al establecer la valoración). Para las aristas falsas sería preceptivo pasar por sus extremos lo que nos obliga a pasar dos veces por uno de ellos si todos fuesen falsos.

- Problema del viajante es NP-completo

Reducimos HAMILTON PATH a él: dado $G = (V, E)$ definimos $G_v = (V, d)$ con $d(v_i, v_j) = 1$ sii $(v_i, v_j) \in E$, $d(v_i, v_j) = 2$ en otro caso

El coste de un camino hamiltoniano es $n-1$, y podría ser cerrado con coste 2 a lo sumo, totalizando $n+1$

Un circuito con coste $n+1$ sólo puede tener un arco de coste 2, que suprimido nos suministra un camino hamiltoniano de G.

- Coloreado de un grafo (con tres colores)

  Se trata de pintar los vertices de distintos colores, sin que dos adyacentes estén pintados del mismo color.

  • El coloreado de grafos con 3 colores es NP-completo.

  Reducimos NAESAT a 3-COLOREADO: Siendo $Var(\phi) = \{x_1, \dots, x_n\}$ introducimos n triángulos con un vértice común con todos ellos, que pintaremos siempre con el color 2; los otros dos vértices libres que corresponden a x_i y $\neg x_i$ se pintarían con 0 o 1.
  Cada cláusula de ϕ da lugar a un triángulo adicional, cuyos vértices se etiquetan con sus literales, conectándose cada uno de ellos con su negación en los primeros triángulos.
  Al menos (y a lo más) un vértice de cada cláusula deberá ser pintado con 0 y otro con 1; éstos representan los literales falso y cierto que hemos de tener en la cláusula, el color 2 del tercero representa que puede ser cierto o falso arbitrariamente.
  La coherencia de la valoración se establece a través de los arcos que conectan los triángulos.

ALGUNOS PROBLEMAS SOBRE CONJUNTOS FINITOS Y NUMEROS NATURALES

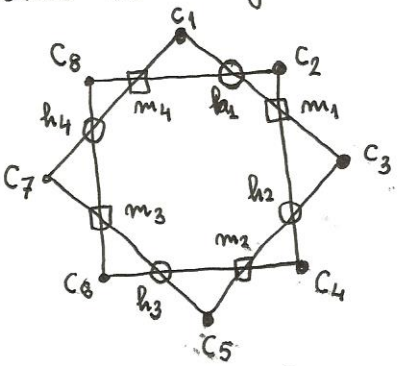
Matching tripartito

Sean A_1, A_2, A_3 con $|A_i| = n$ y una relación $R \subseteq A_1 \times A_2 \times A_3$ se trata de saber si R contiene n tuplas disjuntas, o sea podemos tomar $A_i = \{a_{i1}, \dots, a_{in}\}$ con $R(a_{1j}, a_{2j}, a_{3j})$ $\forall j \in 1..n$.

• MATCHING TRIPARTITO es NP-completo

Reducimos 3 SAT a MATCHING TRIPARTITO. Dado que la construcción trata de manera distinta a los tres conjuntos, tomaremos para aumentar la claridad $A_1 = H$, $A_2 = M$, $A_3 = C$.

• La construcción se basa en el manejo de una serie de polígonos estrellados, uno por cada $x \in Var(\phi)$, cada uno de los cuales tiene k_x lados, siendo k_x el mayor entre el número de ocurrencias de x y el de $\neg x$ en ϕ. Así cada ocurrencia quedaría representada por una c_i distinta, con las impares correspondiendo a x y las pares a $\neg x$.
Los triángulos de la figura definen las ternas de R en que participan los h's y m's que aparecen en ella



- Sólo hay dos formas de agrupar dichas h's y m's de forma disjunta, en una de ellas se ligan los c_i's impares y en la otra los pares. Ello corresponderá con los valores de verdad que asignaremos a la variable $x : T$ en el primer caso y F en el segundo.
- Cada cláusula $C_i = l_{i1} \vee l_{i2} \vee l_{i3}$ de $\emptyset$ introduce dos elementos más, una de H y otro de M : h^i y m^i, respectivamente con $R(h^i, m^i, c^{i1})$, $R(h^i, m^i, c^{i2})$, $R(h^i, m^i, c^{i3})$, siendo los c^{ij} los correspondientes a las ocurrencias de los literales de que se trate.
- La c^{ij} con la que ligamos a h^i, m^i tendrá que estar libre, lo que significa que el correspondiente literal vale T, haciendo cierta a $\emptyset$
- Para equilibrar el cardinal de C con los de H y M se introducen los correspondientes nuevos elementos de estos últimos, que se relacionan con todo el mundo para no generar nuevas restricciones.

Generalizaciones del problema que devienen inmediatamente NP-completos

— EXACT COVER BY 3-SETS : $F = \{S_1, \dots, S_m\}$ con $S_i \subseteq U$
$|S_i| = 3$, $|U| = 3m$ ¿$\exists\ S_{i_1}, \dots S_{i_m}$ $U = \cup S_{i_k}$?

. EXACT COVER BY 3-SETS es NP-completo pues generaliza MATCHING TRIPARTITO : $U = H \cup M \cup C$ $r = \langle h, m, c \rangle \in R$ genera $\{h, m, c\} \in F$

— SET COVERING : $F = \{S_1, \dots, S_m\}$ con $S_i \subseteq U$. Dado $m \in \mathbb{N}$
¿$\exists\ S_{i_1}, \dots, S_{i_m}$ $U = \bigcup_{j=1}^{m} S_{i_k}$?

. SET COVERING es NP-completo pues generaliza EXACT COVER BY 3-SETS

— SET PACKING : $F = \{S_1, \dots, S_m\}$ con $S_i \subseteq U$. Dado $m \in \mathbb{N}$
¿$\exists\ S_{i_1} \dots S_{i_m}$ $S_{i_j} \cap S_{i_\ell} = \emptyset$ $\forall j, \ell$ $j \neq \ell$.

. SET PACKING es NP-completo pues también generaliza EXACT COVER BY 3-SETS : Toda familia de m conjuntos que cubre U ha de estar formada por conjuntos disjuntos y viceversa.

El problema de la mochila

— Programación lineal entera : $\exists\, \bar{x} \in \mathbb{Z}^n \quad \bar{x}\,\bar{a}_i \leq b_i \quad \forall\, i \in 1..k$?

. Generaliza SET COVERING, luego es NP-duro.
. Probar que está en NP no es sencillo, pero puede hacerse } NP-completo.
. Programación lineal (real) está en P, ¡ aunque el algoritmo del simplex tenga casos peores de coste exponencial !

— Problema de la mochila : Dados $p_i, v_i \in \mathbb{N}$, $i = 1..n$, $P \in \mathbb{N}$, $L \in \mathbb{N}$

$\exists\, I \subseteq 1..n \quad \sum_{i \in I} p_i \leq P \;,\; \sum_{i \in I} v_i \geq L$?

• MOCHILA es NP-completo
Basta restringirnos al caso $v_i = p_i$, $L = P$ que se corresponde con el problema LLENAR MOCHILA.
Reduciremos EXACT COVER BY 3-SETS a LLENAR MOCHILA :
$|S_i| = 3$, $S_i \subseteq 1..3m$. Consideramos la función característica de cada S_i, buscamos $\sum_{i \in I} \chi_i = 1_{1..3m}$ sin repeticiones. Podemos ver los valores de las χ_i como números con $3m$ bits y sumar los correspondientes, buscando totalizar $2^{3m} - 1$ pero sin llevadas. Para garantizar que estas jamás se producirán basta pasar a trabajar en base $n+1$ buscando el valor $\sum_{j=0}^{3m-1} (n+1)^j$, utilizando los valores característicos $p_i = \sum_{j \in S_i} (n+1)^{3m-j}$.

. Algoritmo de programación dinámica que resuelve MOCHILA en tiempo $O(nL)$
¡ No pasa nada, ya que el tamaño de L es logarítmico !
Problemas fuertemente NP-completos : no manejan enteros de valor exponencial con respecto al tamaño total de la entrada
Los problemas fuertemente NP-completos no pueden ser resueltos con algoritmos " pseudo-polinomiales " como el de programación dinámica para MOCHILA.

Optimización del tiempo medio ponderado de terminación en 2 procesadores

(TMPT2)

Datos del problema

- Duración, t_i, de cada tarea.
- Factor de ponderación, w_i, de cada tarea.

Tiempo medio ponderado de terminación

Denotaremos las aportaciones de P_1 con una barra y las de P_2 con dos. Redenominamos las tareas llamando $\overline{T_i}$ y $\overline{\overline{T_i}}$ a la i-ésima tarea a ejecutar en P_1 y P_2, respectivamente. Entonces:

$$TMPT2 = \frac{1}{n}\left[\left(\overline{w}_1\overline{t}_1 + \overline{w}_2(\overline{t}_1+\overline{t}_2)+\ldots+\overline{w}_k(\overline{t}_1+\ldots+\overline{t}_K)\right)+\left(\overline{\overline{w}}_1\overline{\overline{t}}_1+\overline{\overline{w}}_2(\overline{\overline{t}}_1+\overline{\overline{t}}_2)+\ldots+\overline{\overline{w}}_\ell(\overline{\overline{t}}_1+\ldots+\overline{\overline{t}}_\ell)\right)\right]$$

Consideraremos el caso particular en el que $\forall i \quad w_i = t_i$. Entonces,

$$n\cdot TMPT2 = \frac{1}{2}\left[\left(\sum\overline{w}_i\right)^2+\sum\overline{w}_i^2\right]+\frac{1}{2}\left[\left(\sum\overline{\overline{w}}_i\right)^2+\sum\overline{\overline{w}}_i^2\right] =$$

$$= \frac{1}{2}\left[\left(\sum\overline{w}_i\right)^2+\sum w_i^2+\left(\sum w_i-\sum\overline{w}_i\right)^2\right] = \frac{1}{2}\left[2\left(\sum\overline{w}_i\right)^2+\sum w_i^2+\left(\sum w_i\right)^2-2\sum w_i\sum\overline{w}_i\right]$$

Comprobemos ahora que $n\cdot TMPT2 \geq \frac{1}{2}\sum w_i^2+\frac{1}{4}\left(\sum w_i\right)^2$. Ello equivale a

$$2\left[2\left(\sum\overline{w}_i\right)^2+\left(\sum w_i\right)^2-2\sum w_i\sum\overline{w}_i\right] \geq \left(\sum w_i\right)^2,$$ que nos lleva a

$$4\left(\sum\overline{w}_i\right)^2+\left(\sum w_i\right)^2-4\sum w_i\sum w_i \geq 0,$$ que es cierto pues el lado izquierdo vale $\left(2\left(\sum\overline{w}_i\right)-\sum w_i\right)^2$.

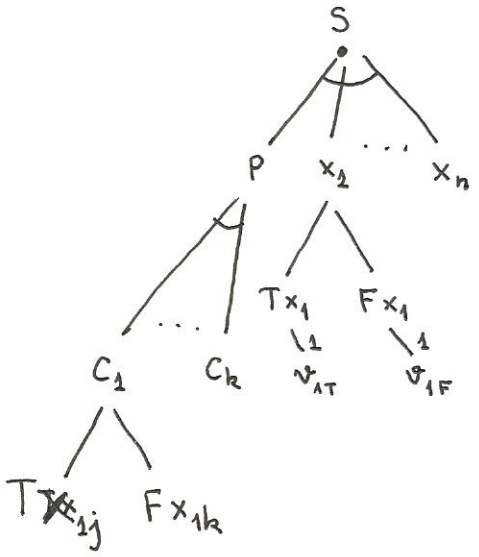
Versión de decisión del problema de optimización

Estudiamos si es posible conseguir que $TMPT2 \leq \frac{1}{n}\left(\frac{1}{2}\sum w_i^2+\frac{1}{4}\left(\sum w_i\right)^2\right)$.
Según la desigualdad probada arriba, sólo sería posible conseguir exactamente la igualdad, para lo cual se necesitaría que $\sum\overline{w}_i = \frac{1}{2}\sum w_i$.
En consecuencia el problema de la partición tiene solución para $\{t_1,\ldots,t_n\}$ sí y solo si el problema de decisión correspondiente a la minimalización del tiempo medio ponderado de terminación del sistema con tareas $\{T_1,\ldots,T_n\}$ con duración t_i y factor de ponderación $w_i = t_i$, tiene solución para la cota de tiempo $\frac{1}{n}\left(\frac{1}{2}\sum t_i^2+\frac{1}{4}\left(\sum t_i\right)^2\right)$.

AND - OR GRAPHS

CNF - satisfatibilidad $\longrightarrow$ AND-OR graph decisión.

$$P = \bigwedge_{i=1}^{k} C_i \qquad C_i = \bigvee l_j$$



Tx_1 será resoluble $\Leftrightarrow$ x_1 es cierto

x_{1j} aparece en C_1

$\overline{x_{1k}}$ aparece en C_1

Todos los arcos salvo los que salen de Tx_i y Fx_i tienen coste nulo.

* Una asignación que satisface P tendría coste n
* Cualquier otra tiene coste mayor.

TEOREMA DE COOK

Descripción de la máquina

- Memoria con palabras de longitud ω : $M(i,j)$ nos da el j-ésimo bit de la palabra en la dirección i.
- Tiempo de ejecución $p(n)$
  - * Dato de n palabras almacenado en la configuración inicial
  - * Se manejan $p(n)$ direcciones de memoria
- Instrucciones básicas
  - * Asignación (incluyendo manejo de vectores)
  - * Asignación no-determinista
- Instrucciones de salto y terminación

Descripción de las fórmulas

* Variables

  $B(i,j,t)$ estado de $M(i,j)$ en el instante t

  $S(j,t)$ contador de programa vale j en el instante t

* Formulas que definen la simulación
  - Inicialización : Se definen los valores de $B(i,j,0)$ y sólo $S(1,0)$ es cierto.
  - Determinismo de los cómputos : fijado t sólo un $S(j,t)$ será cierto
  - Mantenimiento del control : se actualiza el valor de $S(j,t+1)$
  - Actualización de la memoria al ejecutar asignaciones.
  - Detección de la instrucción de éxito tras tiempo $p(n)$.