Demostración de $\mathcal{NP}$-completitud LaTeX

3 DIMENSIONAL MATCHING

Sofía Pizarro Arbelo

alu0100831696

Curso 2017/2018

Índice

1. Introducción

$\mathcal{NP}$ es el conjunto de lenguajes resolubles en tiempo polinómico no determinista, mientras que $\mathcal{NP}$-completo son los más difíciles de la clase, se caracterizan por ser todos iguales. La teoría $\mathcal{NP}$-completo se basa en el concepto de **transformación polinomial**.

Un ejemplo de $\mathcal{NP}$-completo: supongamos que queremos hacer algoritmo eficiente para un problema muy complicado a resolver. ¿Es algo imposible? Probablemente no, veamos por qué. Después de romperte la cabeza durante horas solo se te ocurre un algoritmo de *fuerza bruta*, con su respectivo tiempo exponencial y entonces piensas que se trata de un problema intratable, que no existe un algoritmo para ello y es ahí cuando puedes demostrar de alguna manera que es $\mathcal{NP}$-completo. Los $\mathcal{NP}$-completos parecen intratables, aunque nadie ha sabido demostrar que los $\mathcal{NP}$-completos son intratables. Son todos equivalentes, es decir:

- Si se encuentra un algoritmo eficiente para un $\mathcal{NP}$-completo entonces tenemos un algoritmo eficiente para cualquiera de ellos.
- Si probamos que un $\mathcal{NP}$-completo no tiene algoritmos eficientes entonces ninguno los tiene.

La necesidad de buscar una solución al problema no desaparecerá, pero podemos:

- Dejar de buscar un algoritmo en tiempo polinómico para el problema.
- Buscar un algoritmo eficiente para un problema diferente relacionado con el original.
- O bien intentar usar el algoritmo exponencial a ver qué tal funciona con varios valores a probar.

2. Clase P y NP

2.1. Definición

- Un problema de decisión está en la clase P si las instancias "sí" son reconocidas por una MTD polinomial.
- Un problema de decisión está en la clase $\mathcal{NP}$ si las instancias "sí" son reconocidas por una MTND polinomial.

- Alternativamente, La clase $\mathcal{NP}$ se puede definir como el conjunto de problemas de decisión que admiten un certificado polinomial.

2.2. Relaciones entre las clases:

1. $\mathcal{P} \subseteq \mathcal{NP}$
2. Problema abierto: ¿Es $\mathcal{P} = \mathcal{NP}$?

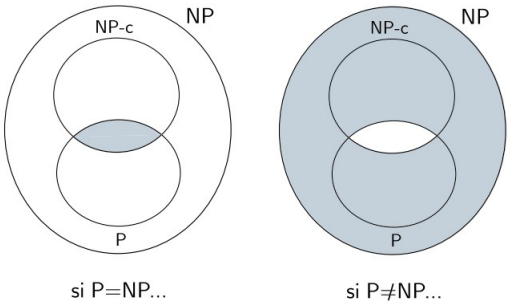
3. Transformaciones polinomiales

- Una transformación o reducción polinomial de un problema de decisión Π1 a uno Π2 es una función que se computa en tiempo polinomial y transforma una instancia I1 de Π1 en una instancia I2 de Π2 tal que I1 tiene respuesta "sí" para Π1 si y solo si I2 tiene respuesta "sí" para Π2.
- El problema de decisión Π1 se reduce polinomialmente a otro problema de decisión Π2, Π1 $\leq$p Π2, si existe una transformación polinomial de Π1 a Π2.
- Si Π" $\leq$p Π' y Π' $\leq$p Π entonces Π" $\leq$p Π, ya que la composición de dos reducciones polinomiales es una reducción polinomial.

4. Problemas $\mathcal{NP}$ Completos

Un problema Π es $\mathcal{NP}$-completo si:

1. Π $\in \mathcal{NP}$.
2. Para todo Π' $\in \mathcal{NP}$, Π' $\leq$p Π.



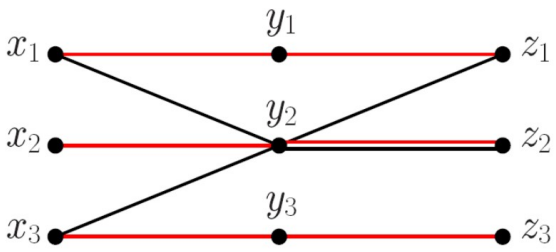
5. Descripción del problema 3DM

1. Instancia

   - Un conjunto M c W x X x Y
     - W ∩ Y ∩ X = ϕ (disjuntos)
     - |W| = |X| = |Y| = q

2. ¿L1 ≤n L?

   - |M'| = q
   - Todos los elementos W u X u Y están en alguna terceta de M' sin repetir ninguno.



$$M = \{(x_1, y_1, z_1), (x_1, y_2, z_2), (x_2, y_2, z_2), (x_3, y_3, z_3), (x_3, y_2, z_1)\}$$

6. Demostrar que un 3DM es NP completo

A partir del Teorema de Cook, la técnica estándar para probar que un problema Π es $\mathcal{NP}$-completo aprovecha la transitividad de ≤p, y consiste en lo siguiente:

1. **Mostrar que Π está en $\mathcal{NP}$.**

2. **Elegir un problema Π' apropiado que se sepa que es $\mathcal{NP}$-completo.**

3. **Construir una reducción polinomial f de Π' en Π.**

La segunda condición en la definición de problema $\mathcal{NP}$-completo sale usando la transitividad: sea Π0" un problema cualquiera de $\mathcal{NP}$. Como Π' es $\mathcal{NP}$-completo, Π" ≤p Π'. Como probamos que Π' ≤p Π, resulta Π" ≤p Π.

En este caso específico:

1. **Demostrar que 3DM es $\mathcal{NP}$.**

2. **Seleccionar un problema Π' que sea $\mathcal{NP}$-completo**

3. **Construir una transformación (f) Π' α 3DM**

4. **Comprobar que la transformación f se hace en tiempo polinomial.**

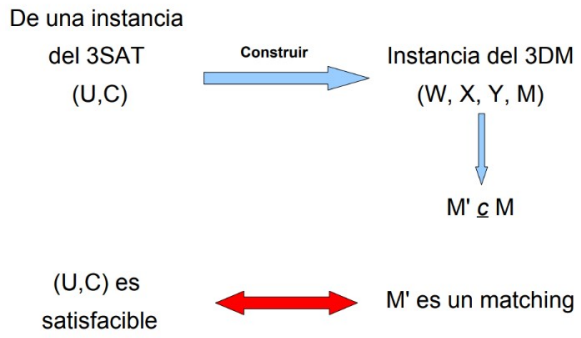
6.1. Demostrar que un 3DM es NP

Dada una instancia (M, X, Y, W) del 3DM se construye un algoritmo no determinista que genere una solución de |W| tercetas de M y compruebe en tiempo polinomial que no hay dos tercetas con elementos comunes.

6.1.1. ALGORITMO Π'

⋆3 SATISFABILITY (3SAT) Instancia: - Conjunto de m cláusulas C = c1, ..., cm ∘ |ci| = 3 , 1 ≤ i ≤ m - Sobre un conjunto finito de n variables booleanas ∘ U = u1, ..., un ¿Existe alguna asignación válida de U que satisfaga todas las cláusulas de C.?

6.1.2. 3SAT α 3DM



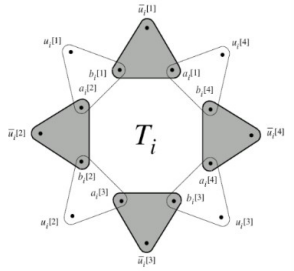
- Técnica para la demostración de $\mathcal{NP}$-completitud: Diseño de Componentes
- Notación:

3SAT		3DM
Variables: $u_1, ..., u_n$	⟶	**Variables:** $u_1(j), a_i(j), b_i(j), s_x(j), g_y(j)$
Literales: $u_1, \neg u_1$	⟶	**Variables:** $u_1(j), \neg u_1(j)$
Cláusulas: $c_j = (u_1, \neg u_2, u_3)$	⟶	**Tercetas:** $C_j = \{(u_1(j), s_x(j), s_y(j)),$ $(\neg u_2(j), s_x(j), s_y(j)),$ $(u_3(j), s_x(j), s_y(j))\}$

- La demostración se basa en la construcción de tres tipos de componentes:
  - Tercetas de asignación Para cada variable ui ∈ U se introduce una componente Ti.
    - Ti depende del número de cláusulas m de C
    - Estructura de Ti:
      1. Elementos internos:
         - ai [j] ∈ X, 1 ≤ j ≤ m ; bi [j] ∈ Y, 1 ≤ j ≤ m

           No van a pertenecer a otras tercetas de otro Ti
      2. Elementos externos:
         - ui [j] , ¬ [j] ∈ W, 1 ≤ j ≤ m

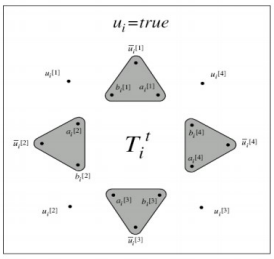
           Pueden pertenecer a otras tercetas
    - El literal ui en 3SAT puede ser usado en varias cláusulas, en el 3DM debemos tener muchas m copias de ui.

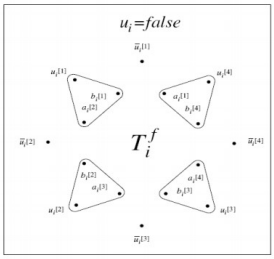


$T_i^t = \{(\overline{u}_i[j], a_i[j], b_i[j]) : 1 \leq j \leq m\}$

$T_i^f = \{\,(u_i[j], a_i[j+1], b_i[j]) : 1 \leq j \leq m\} \cup \{(u_i[m], a_i[1], b_i[m]) : 1 \leq j \leq m\}$

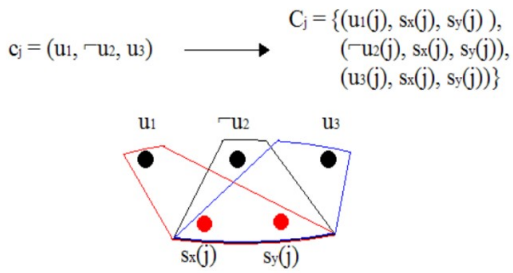
- Si ningún elemento interno de la componente Ti aparece en otra Th (i *neq* h):
  - M' será un matching con m elementos de Ti





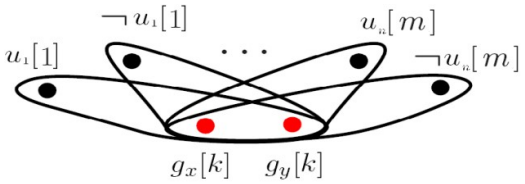
  - Si ui = true se elegirá como M' las tercetas grises, dejando libre el resto para poder utilizarlas en la construcción del resto de componentes.

- Tercetas de satisfacción
  - Para cada cláusula cj ∈ C introducimos una componente Cj.
  - - Estructura:
    1. Elementos Internos:
       - sx [j] ∈ X, sy [j] ∈ Y : 1 ≤ j ≤ m
    2. Elementos externos:
       - ui [j] , ¬ui [j] ∈ W : 1 ≤ i ≤ n;1 ≤ j ≤ m
  - Cj = (ui [j], sx [j], sy [j]): si el literal ui ∈ cj U (¬ui [j], sx [j], sy [j]): si el literal ¬ui ∈ cj



- - Cualquier matching M' c M debe contener una terceta de Cj para emparejar los elementos internos sx [j] y sy [j]:
    1. Sx [j] y Sy [j] pueden ser emparejados, sí sólo sí, al menos uno de los literales (ui) de cj no ha sido emparejado en alguna componente "Truth seeting" Ti (Ti ∩ M')
    2. o Si tenemos una 3SAT-Instancia satisfacible, entonces las variables Sx[j] y Sy[j] pueden ser emparejadas
    3. o Si tenemos una 3SAT-Instancia no satisfacible, entonces las variables Sx[j] y Sy[j] no pueden ser emparejadas.

- Tercetas de relleno
    - Hay muchos ui [j] que no se emparejan con componentes de relleno ni con componentes de satisfacción
    - Introducimos m (n -1) variables nuevas:
        1. gx [k] ∈ X , gy [k] ∈ Y: $1 \leq k \leq m(n-1)$
    - ¿ Por qué m (n-1) variables?
        1. Hay mxn variables u sin emparejar después de calcular las tercetas de asignación.
        2. Si todas las m cláusulas se satisfacen se han emparejado m variables.
        3. Finalmente quedan sin emparejar (mxn) – m = m(n-1)
    - Cada pareja (gx [k], gy [k]) se enlazará con una única variable ui [j] o ¬ui [j] que no estén en las tercetas que se han formado con las componentes anteriores:



$$G = \{(u_i[j], g_x[k], g_y[k]), (\overline{u}_i[j], g_x[k], g_y[k])$$
$$: 1 \leq k \leq m(n-1), 1 \leq i \leq n, 1 \leq j \leq m\}$$

En resumen:

$W = \{(u_i[j], \overline{u}_i[j] : 1 \leq i \leq n, 1 \leq j \leq m\}$ $(2mn)$

$X = A \cup S_X \cup G_X$ $(2mn)$

$A = \{a_i[j] : 1 \leq i \leq n, 1 \leq j \leq m\}$
$S_X = \{s_x[j] : 1 \leq j \leq m\}$
$G_X = \{g_x[j] : 1 \leq j \leq m(n-1)\}$

$Y = B \cup S_Y \cup G_Y$ $(2mn)$

$B = \{b_i[j] : 1 \leq i \leq n, 1 \leq j \leq m\}$
$S_Y = \{s_y[j] : 1 \leq j \leq m\}$
$G_Y = \{g_y[j] : 1 \leq j \leq m(n-1)\}$.

$$M = \left(\bigcup_{i=1}^{n} T_i\right) \cup \left(\bigcup_{j=1}^{m} C_j\right) \cup G. \quad (2mn+3m+2m^2n(n-1))$$

- Se ha observado que las tercetas resultantes M son el producto cartesiano de W x X x Y
- Esta forma de definir las tercetas:
  1. Desde su definición en términos de una instancia (U,C) del 3SAT
  2. M se construye en tiempo polinomial

Para completar la demostración de NP Completitud falta por demostrar:

M contiene un matching M' $\Longleftrightarrow$ (U,C) es satisfacible

(U,C) es satisfacible $\Longrightarrow$ **M' c M es un matching**

- Sea t: U $\rightarrow$ T, F el dominio de valores para U que satisface las cláusulas C.
- Se construye un matching M' c M del modo siguiente:

1. Para cada cláusula cj ∈ C:
   - Zj ∈ ui, ¬ui: 1 ≤ i ≤ n ∩ cj

     Literales con asignación verdadera.

     Debe de existir al menos uno, ya que t satisface a cj.
2. Se construye la M':

$$M' = \left(\bigcup_{t(u_i)=T} T_i^t \right) \cup \left(\bigcup_{t(u_i)=F} T_i^f \right) \cup \left(\bigcup_{j=1}^{m} \{(z_j[j], s_x[j], s_y[j])\} \right) \cup G'$$

   - G': conjunto de m(n-1) tercetas de G que incluyen:

     Todos los gx [k] ∈ X , gy [k] ∈ Y

     Y los ui[j] , ¬ui[j] ∈ W que no se han emparejado.
   - Es fácil de verificar que siempre se puede construir un G' para que el resultado del conjunto M' sea un matching.

M' ⊆ M es un matching ⟹ **(U,C) es satisfacible**

- Se ha visto que para cada ui ∈ U, M' incluía exactamente m tercetas de Ti: Tt i o Tf i.
- Sea t: U → T, F donde t(ui) = T ↔ M' ∩ Ti = Tt i
  1. t será una asignación correcta que satisface C.
- Consideremos una cláusula arbitraria cj ∈ C
  1. Para cubrir los elementos internos de la componente Cj (de la componente de testing)
     - Se necesita al menos una terceta de Cj contenida en M'.
     - Esta terceta contiene un literal de cj ∈ C, que no estará en M' ∩ Ti
  2. Como t(ui) = T ↔ M' ∩ Ti = Tt i
     - Entonces t satisface la cláusula cj ∩ Ti
  3. Si todas las cláusulas cj ∈ C se satisfacen
     - (U,C) es satisfacible

3-Dimensional Matching es $\mathcal{NP}$-COMPLETO