.UBAfiuba
FACULTAD DE INGENIERÍA

Clases “P” y “NP”

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Introducción

Observamos

diferentes algoritmos para resolver problemas concretos

Llamamos

“tractables” a aquellos resolubles en tiempo polinomial.

Vimos

Que para algunos problemas existen algoritmos tractables (y para otros no)

Deseamos

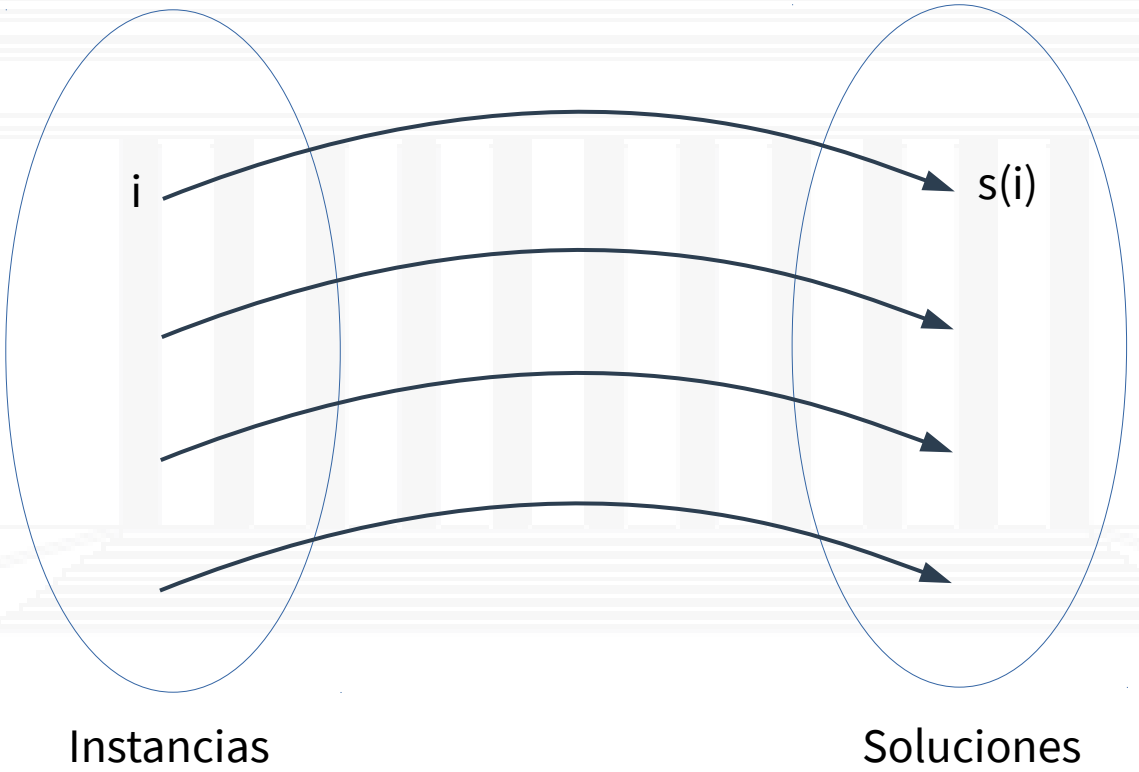
Poder clasificar a los problemas de acuerdo a la complejidad de su resolución

Y de esa forma poder compararlos.

Problema Abstracto vs Instancia de problema

Podemos definir un problema abstracto Q

Como una relación binaria entre un conjunto de instancias y de soluciones



Problema de optimización

Son aquellos problemas

que buscan la mejor solución para maximizar (o minimizar) un resultado.

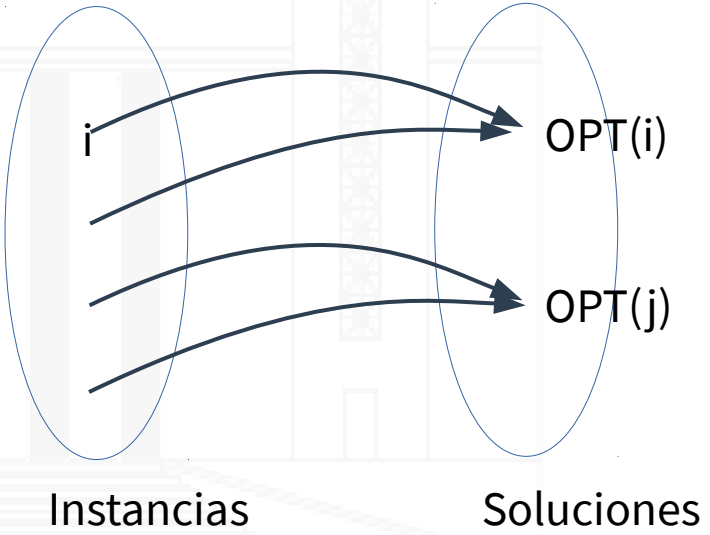
Ejemplos

Maximizar la ganancia que se puede ingresar en una mochila

Minimizar el cambio de monedas

Maximizar el flujo transportado en una red

Minimizar la longitud del circuito del viajante de comercio



Problema de decisión

Son aquellos problemas

Cuya solución solo puede tomar 2 valores: SI/NO

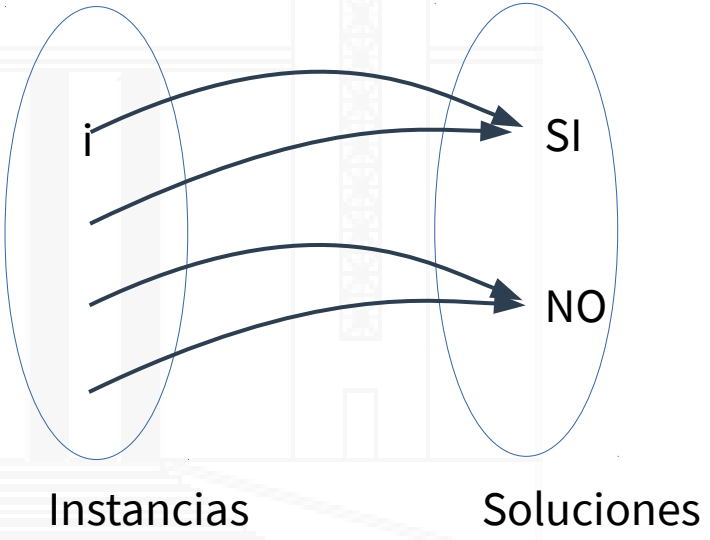
Ejemplos

¿ Es el número X primo?

¿ Existe un matching perfecto par los sets A y B ?

¿Es el grafo G fuertemente conexo?

¿ Se puede satisfacer la demanda de flujo para el problema de flujo máximo con demanda ?



Adecuación de problema optimización a decisión

Dado un problema de optimización Q

Podemos reformularlo como un problema de decisión (y viceversa)

Y encontrar la respuesta en tiempo polinómico (o pseudopolinómico)

Ejemplo

Optimización: Cambio mínimo en monedas del valor X

Decisión: Se puede dar cambio mínimo de X con m monedas?

```
Cambio mínimo en monedas del valor X
    Desde #m=1 hasta X
        Si (Se puede dar cambio cambio mínimo de X con #m monedas)
            retornar #m
```

Caracterización de la entrada de un algoritmo

Para resolver un problema concreto con una computadora

Se debe representar la instancia de modo que el programa lo entienda.

Los mismos corresponderán a los parámetros del algoritmo

Los parámetros de un problema computacional

Se codifican como una cadena binaria finita “s”

La longitud de la cadena “s”

La podemos medir como un parámetro n = |s|

Y la utilizamos para medir la complejidad del algoritmo que resuelve el problema

Resolución eficiente

Un algoritmo A

resuelve eficientemente un problema S

Si para toda instancia I de S,

Encuentra la solución en tiempo polinomial

→ existe una constante k / A = $O(n^k)$

Ejemplo:

Gale Shapley resuelve el problema “Stable Marriage Problem” en $O(n^2)$

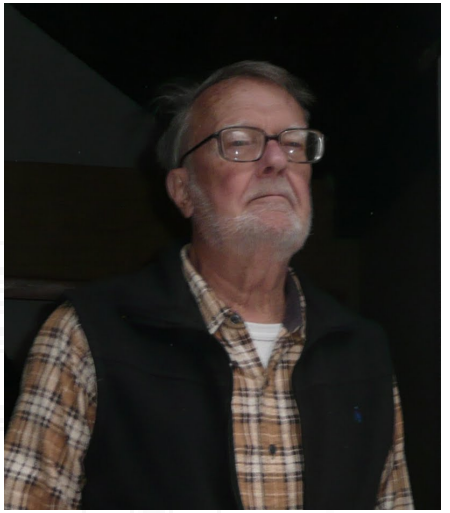
Clase “P”

Se conoce como “P”

Al conjunto de problemas de decisión

para los que existe

un algoritmo que lo resuelve en forma eficiente.



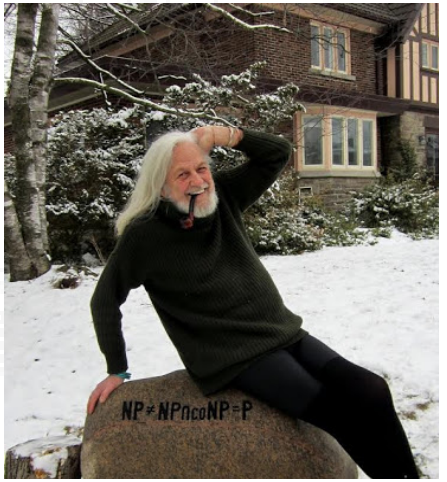
El concepto fue propuesto por:

“The intrinsic computational difficulty of functions”, Alan Cobham en 1964.

https://www.cs.toronto.edu/~sacook/homepage/cobham_intrinsic.pdf

“Paths, trees, and flowers”, Jack Edmonds en 1965.

https://math.nist.gov/~JBernal/p_t_f.pdf



Certificación eficiente

Un algoritmo B

Certifica (o verifica) eficientemente un problema de decisión S

Si para toda instancia I de S,

Dado un certificado t que contiene evidencia de la solución s(I) es “si”

Puede verificar esta afirmación en tiempo polinomial

→ existe una constante k / $B \leq O(n^k)$

El algoritmo B

Recibe 2 parámetros: la instancia I y t

Responde: “Si” o “No”

Certificación eficiente - Ejemplo

Dado el problema de decisión

¿Existe para el viajante de comercio un camino de longitud menor a L para un grafo G=(C,R) (que representa a ciudades y rutas)?

Un algoritmo de certificación recibe como parámetros

El grafo G (instancia del problema)

Un circuito t de ciudades para el viajante (certificado)

Calcula y verifica (en tiempo polinomial)

La longitud del circuito y que el mismo cubra todas las ciudades solo 1 vez

Retorna

“Si” si la evidencia otorgada satisface el problema.

Clase “NP”

Se conoce como “NP”

Al conjunto de problemas de decisión

para los que existe

un algoritmo que lo certifique en forma eficiente.

El concepto fue propuesto por:

“Paths, trees, and flowers”, Jack Edmonds en 1965.

https://math.nist.gov/~JBernal/p_t_f.pdf

¿ “P” ⊆ “NP” ?

Si el problema Q ∈ P

Existe un algoritmo A = $O(n^k)$ que lo resuelve

Podemos definir B(I,t) como

```
B(I,t)
    s = A(I)
    Si s == t
        retornar “si”
    retornar “no”
```

que certifica el problema Q

Y lo hace en tiempo polinomial

Si Q ∈ P ⇒ Q ∈ NP

¿ “NP” ⊆ “P” ?

Si el problema Q ∈ NP

Existe un algoritmo B = O(nk) que lo certifica

¿ Podemos asegurar que la existencia de B

Garantiza la existencia de un algoritmo A polinomial que lo resuelva?

En caso afirmativo: P=NP

“Tiene la misma complejidad resolver un problema que verificarlo”

Sino: P≠NP

“P” VS “NP”

Es un problema

sin resolver dentro de la ciencia de la computación

Propuesto en

"The Complexity of Theorem Proving Procedures", Stephen Cook en 1971

http://www.cs.toronto.edu/~sacook/homepage/1971.pdf

Una amplia mayoría considera que P≠NP

Pero no existe prueba

Millennium Problems

7 problemas matemáticos

Propuestos por Clay Mathematics Institute

El 24 de Mayo de 2000

Una solución correcta acredita un premio de 1millon de dólares

(hasta el momento solo 1 problema fue resuelto)

P vs NP es uno de esos problemas

https://www.claymath.org/millennium-problems

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

Reducciones polinomiales

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Introducción

Buscamos clasificar problemas computacionales

De acuerdo a la complejidad que requiere su resolución

Definimos 2 grandes clasificaciones

Clase P

Clase NP

(existen muchas mas dentro del "ecosistema"…)

Necesitamos una herramienta

Que nos permita comparar 2 problemas

Asignar un problema a una determinada clase

Reducciones

Reducir un problema

a otro conocido para resolverlo

Es un procedimiento

Que utilizamos profusamente en la resolución de problemas computacionales

Se lo puede pensar como

una caja negra

Reducciones – Caja negra

Dado una instancia “y” de un problema Y

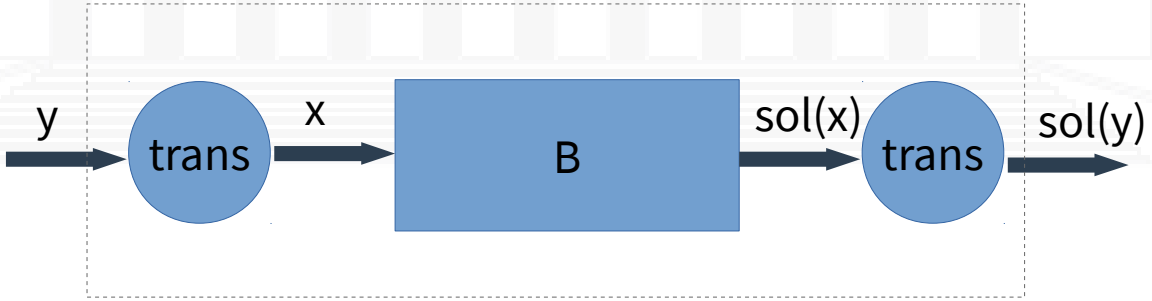
Realizamos una transformación a una instancia “x” del problema X

Resolvemos “x” mediante un algoritmo “B”

Obteniendo el resultado sol(x)

Transformamos el resultado sol(x)

En el resultado sol(y) del problema “y”



Reducción polinomial

Corresponde a una reducción

En la que ambas transformaciones se realizan en tiempo polinomial

Sean

X, Y problemas

Diremos

$Y \leq_p X$ (se lee: “Y” es polinomialmente reducible (en tiempo) a “X”)

Si

podemos transformar cualquier instancia de y en una instancia de x en tiempo polinómico

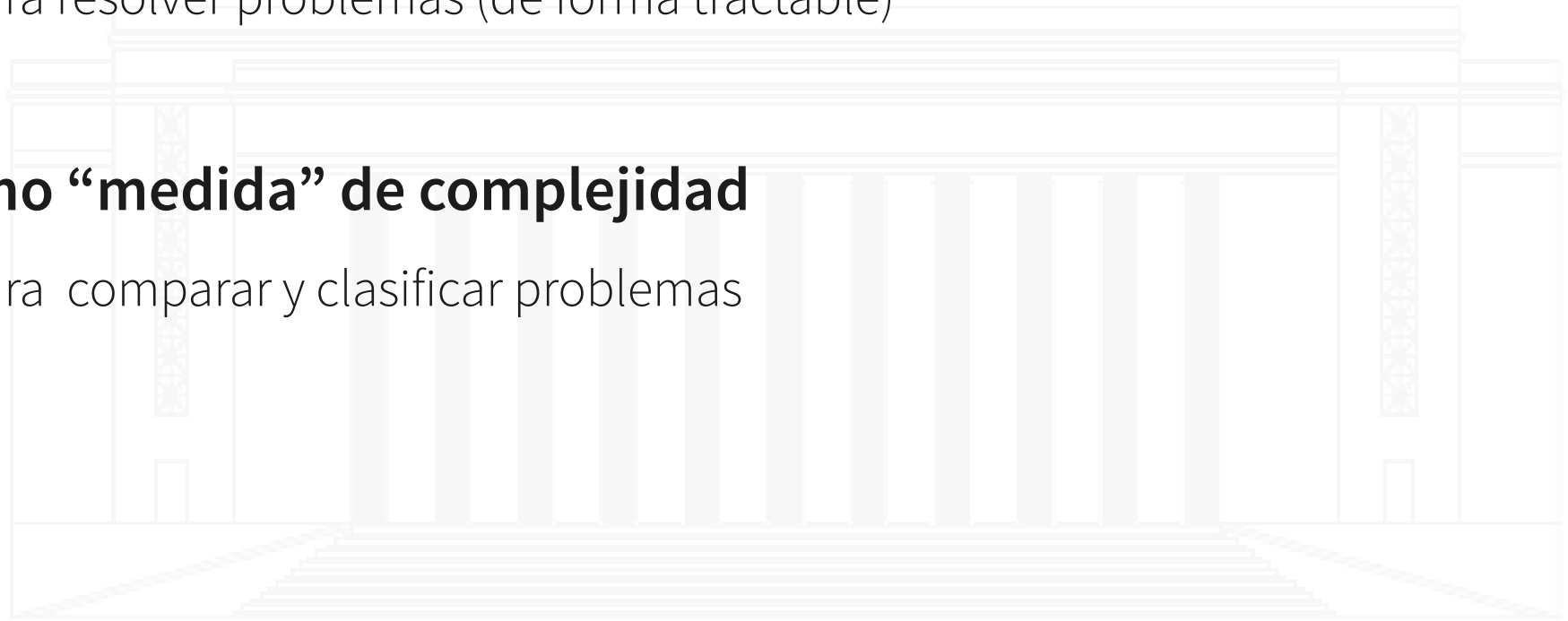
Usos de las reducciones polinómicas

Como “caja negra”

Para resolver problemas (de forma tractable)

Como “medida” de complejidad

Para comparar y clasificar problemas



Origen

Propuesto en:

"Reducibility among combinatorial problems", Richard M. Karp (1972)

https://people.eecs.berkeley.edu/~luca/cs172/karp.pdf

Comparar problemas con reducciones

Sean

X, Y problemas

Si

$Y \leq_p X$

Diremos

Que el problema X es al menos tan difícil que el problema Y

Ejemplo

Sea

El problema “hallar el matching mas grande en un grafo bipartito” (MAX-MATCHING)

Se puede

Reducir polinomialmente al problema de “flujo máximo en una red de flujo” (MAX-FLOW).

Por lo tanto

MAX-MATCHING $\leq_p$ MAX-FLOW

(MAX-FLOW es al menos tan difícil de resolver que MAX-MATCHING)

Acotar un problema a la clase “P”

Sean

X,Y Problemas

Si

$X \in$ “P”

$Y \leq_p X$

Entonces

$Y \in$ “P” (por que X es igual o mas complicado que Y)

En el ejemplo anterior

MAX-MATCHING $\leq_p$ MAX-FLOW

MAX-FLOW $\in$ “P” $\Rightarrow$ MAX-MATCHING $\in$ “P”

Acotar un problema a la clase “P” (cont.)

Sean

X,Y Problemas

Si

$Y \notin$ “P”

$Y \leq_p X$

Entonces

$X \notin$ “P” (por que X es igual o más complicado que Y)

Propiedad: Equivalencia

Sean

X,Y Problemas

Si

$Y \leq_p X$

$X \leq_p Y$

Entonces

X e Y tienen la misma complejidad

Propiedad: Transitividad

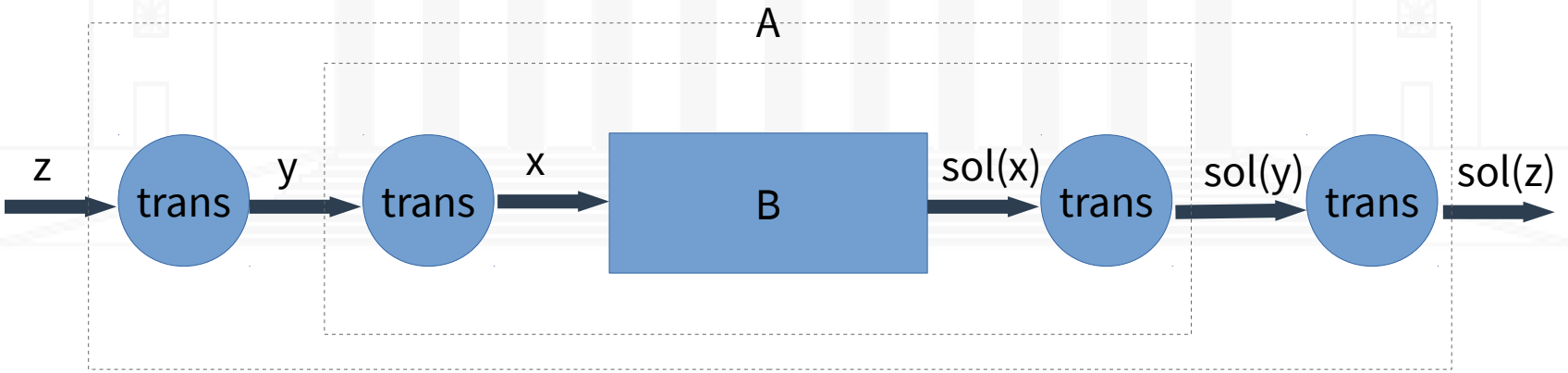
Si

$Z \leq_p Y$

$Y \leq_p X$

Entonces

$Z \leq_p X$



.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

Problemas NP-Completos

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Boolean satisfiability problem

Dado

Un conjunto de variable booleanas

Que definen una expresión booleana (utilizando operadores “OR”, “AND”, “NOT” y sin cuantificadores)

Determinar

Si existe una asignacion de valores de las variables

Tal que

El resultado de la expresión es “TRUE”

(No se conoce un algoritmo que lo resuelva en tiempo polinomial)

Ejemplo

Sea la expresión

$I = (V_1 \text{ or } V_2 \text{ or } \overline{V}_3) \text{ and } (\overline{V}1 \text{ or } V4) \text{ and } (V_2 \text{ or } V_3 \text{ or } \overline{V}_4 \text{ or } V_5) \text{ and } (\overline{V}_2 \text{ or } \overline{V}_5 \text{ or } \overline{V}_4)$

Si probamos la asignación

$V_1 = \text{true}$ $V_2 = \text{false}$ $V_3 = \text{true}$ $V_4 = \text{false}$ $V_5 = \text{true}$

Al evaluar I

Nos dará un valor final de FALSE

(LA segunda clausula: $\overline{V}1 \text{ or } V4$ es false, por lo tanto el resto lo es también

SAT ∈ “NP”

Sea

una instancia I del problema SAT

Un certificado que corresponde a un valor de asignación de cada variable

Podemos certificar en tiempo polinomial

Si esa asignación de variables produce un resultado “TRUE”

Ejemplo

I = (V_1 or V_2 or $\overline{V}_3$) and ($\overline{V}1$ or V4) and (V_2 or V_3 or $\overline{V}_4$ or V_5) and ($\overline{V}_2$ or $\overline{V}_5$ or $\overline{V}_4$)

V_1 = true V_2 = false V_3 = true V_4 = true V_5 = false

Teorema Cook-Levin

Sea

$X \in NP$

Entonces

$X \leq_p$ Boolean satisfiability problem (SAT)

“Todo problema perteneciente a NP es a lo sumo tan complejo de resolver que SAT”

Origen

Propuesto en:

"The complexity of theorem proving procedures", Stephen Cook (1971)

https://www.cs.toronto.edu/~sacook/homepage/1971.pdf

"Universal sorting problems", L. A. Levin (1973)

NP-HARD

Sea

un problema X

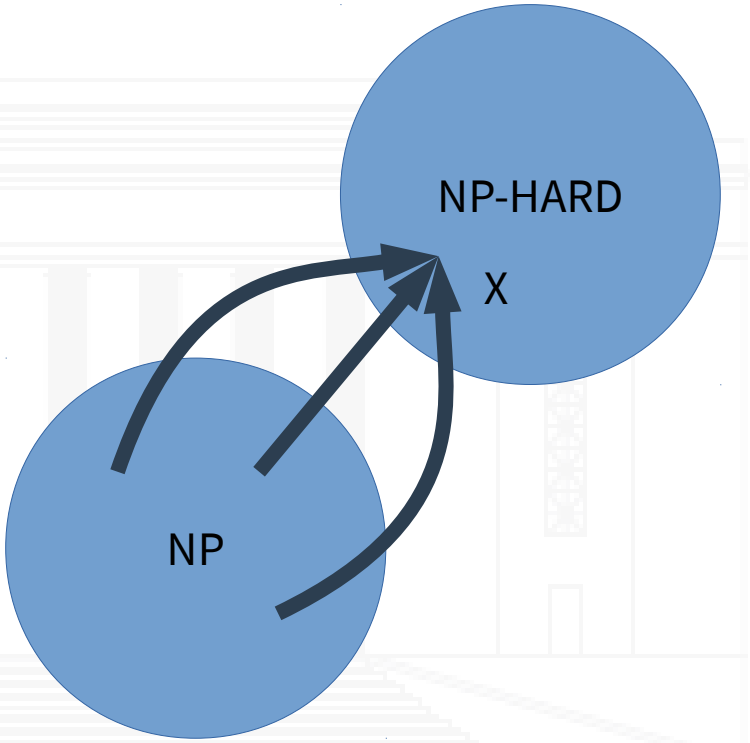
Tal que

Para todo problema $Y \in NP$

$Y \leq_p X$

Entonces

$X \in$ NP-Hard



“X es al menos igual de difícil que cualquier problema en NP”

NP-Complete (o NP-C)

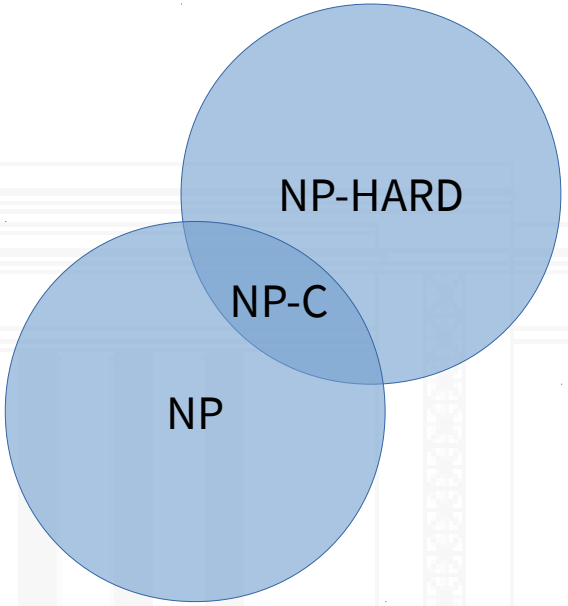
Sea

X ∈NP-Hard

X ∈NP

Entonces

X ∈NP-C



“X es uno de los problemas más difíciles dentro de NP”

SAT ∈ NP-C (demostrado por Cook y Levin)

Los 21 problemas NP-C de Karp

En su paper “Reducibility Among Combinatorial Problems”

Richard Karp presenta la “reducción polinomial”

Para cada problema X ∈ NP que analiza (21)

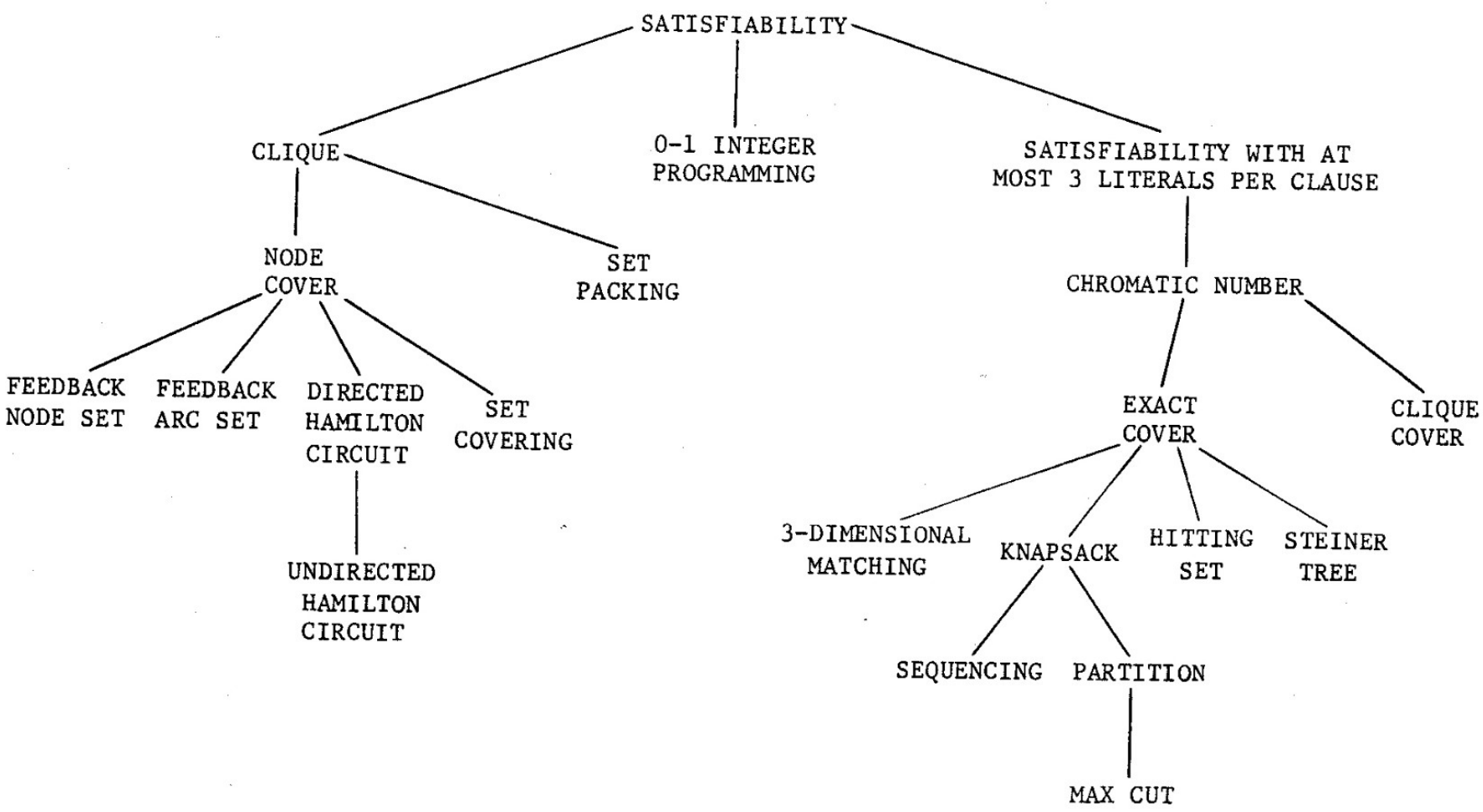
Toma un problema demostrado como NP-C y lo reduce a X

Con esto X pasa a ser NP-C

Existen cientos (o miles?) de problemas

Que se han demostrado como NP-Completos

Los 21 problemas NP-C de Karp (cont.)



Probar que un problema es NP-C

Sea

El problema X de decisión .

Probar que X ∈ NP

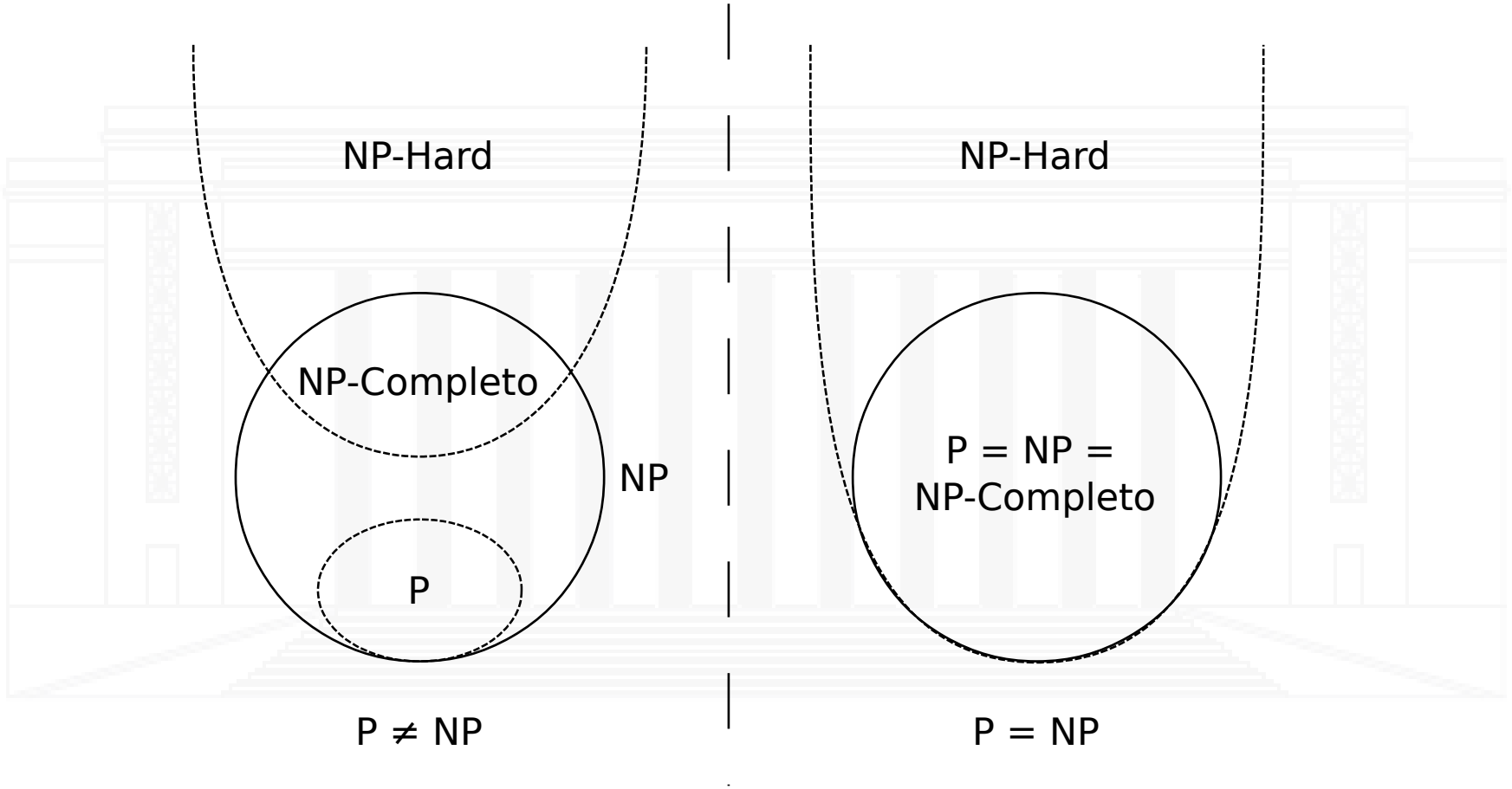
Definir el certificador eficiente

Probar que X ∈ NP-H

Dado un problema Y ∈ NP-C, reducir polinomialmente Y a X

$Y \leq_p X$

P VS NP (con NP-HARD y NP-Complete)



Pasos a realizar para probar que P=NP

Tomar el problema X NP-C

De su preferencia

Construir

Un algoritmo que resuelva X en tiempo polinomial

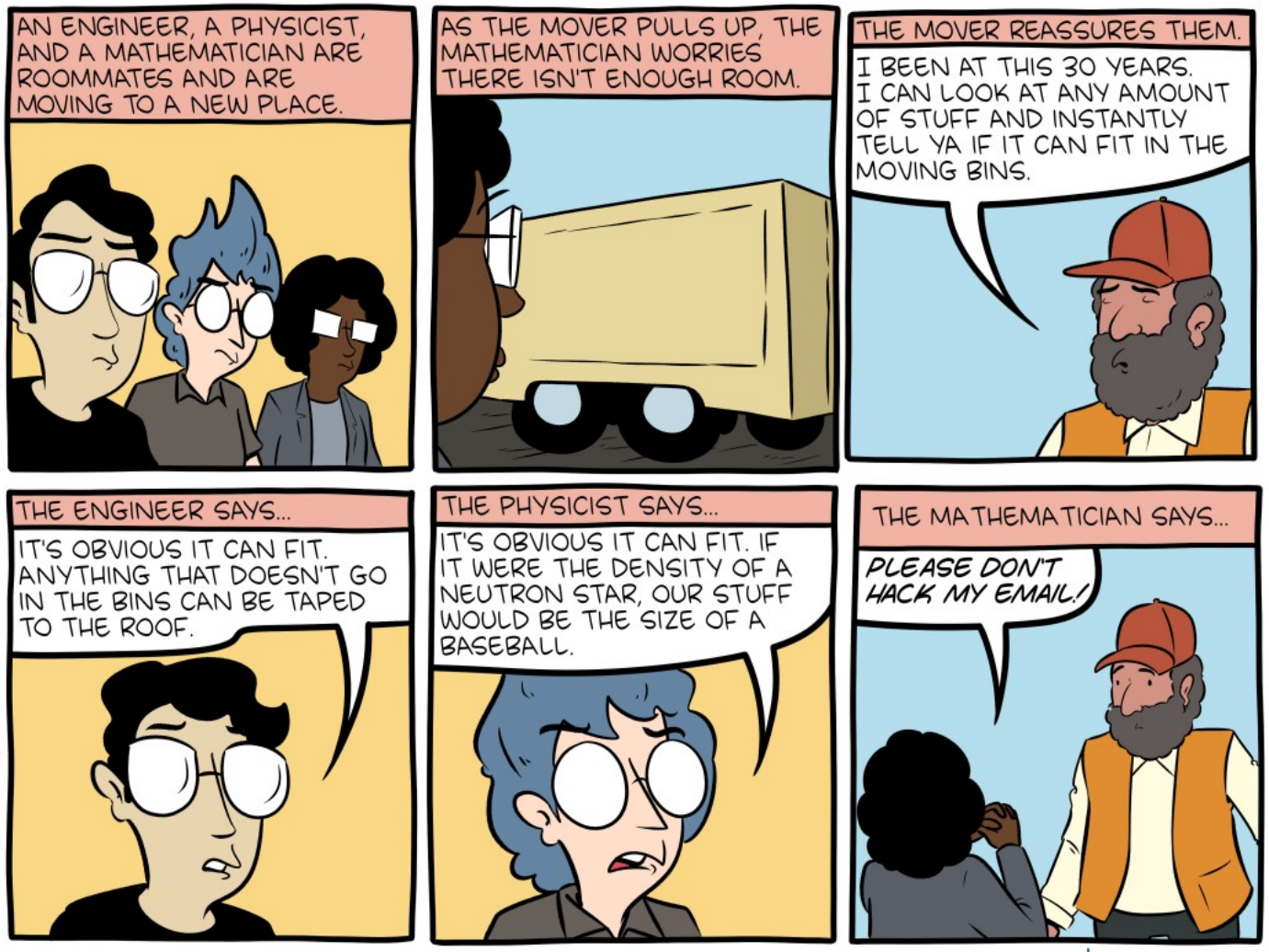
Todo problema NP-C se puede reducir entre si con igual complejidad

Y todo problema NP se puede reducir a otro NP-C

Entonces habrá probado

Que existe un algoritmo polinomial para todo problema NP y P=NP

Humor NP-Complete



.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

NP-C: Conjunto independiente

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Conjunto independiente

Sea

Un grafo G=(V,E)

Un valor k

Determinar

Si existe un conjunto independiente de nodos de como mucho tamaño K

(Problema conocido como “Independent Set”)

Definiciones

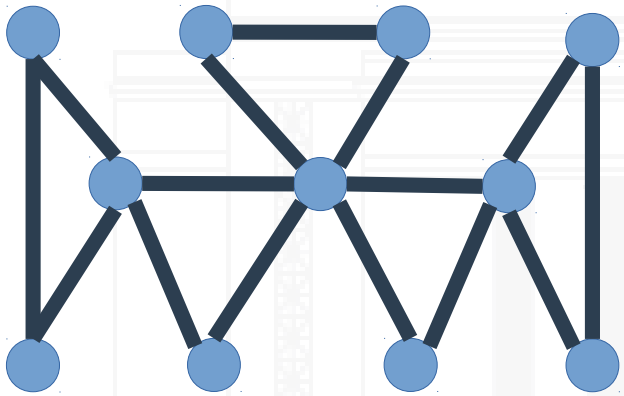
Un conjunto de nodos $C \subseteq V$ es independiente si

No existe $a,b \in C$ tal que existe eje $(a,b) \in E$

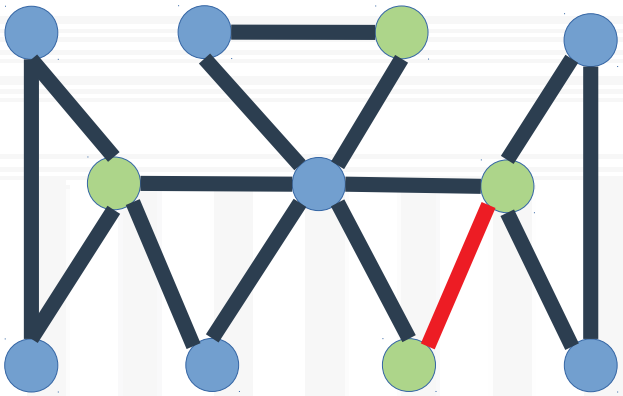
El tamaño del conjunto independiente

Corresponde a la cantidad de nodos dentro del conjunto C

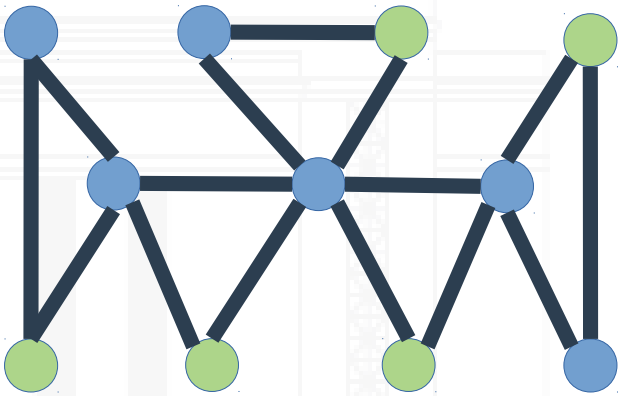
Ejemplo



Grafo G



Set no independiente



Set independiente de tamaño 5

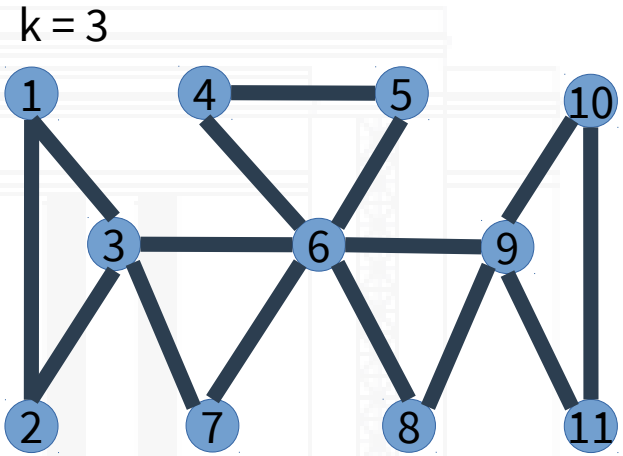
¿Conjunto independiente es “NP”?

Dado

G=(V,E)

k tamaño del conjunto

T certitifcado = subconjunto de nodos



T={3, 4, 9}

Puedo verificar (en tiempo polinomial)

|T| = k

∀ a,b ∈ T, !∃ (a,b) ∈ E

⇒ INDEPENDENT-SET ∈ NP

¿Conjunto independiente es “P”?

No se conoce algoritmo

Que resuelva INDEPENDENT-SET en tiempo polinómico

Si probamos que

INDEPENDENT-SET $\in$ NP-C

(Utilizaremos 3SAT)

Entonces

INDEPENDENT-SET $\in$ P $\Leftrightarrow$ P = NP

3SAT

Es una variante de SAT

Karp probó en 1972 $SAT \leq_p 3SAT \Rightarrow 3sat \in NP\text{-}C$

Dado

$X=\{x_1,...x_n\}$ conjunto de n Variables booleanas = {0,1}

k clausulas booleanas $Ti = (t_{i1} \vee t_{i2} \vee t_{i3})$

Con cada $t_{ij} \in X \cup \overline{X} \cup \{1\}$

Determinar

Si existe asignación de variables tal que $T_1 \wedge T_2 \wedge ... \wedge T_k = 1$

Ejemplo

Sea la expresión

$$E = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$$

La asignación

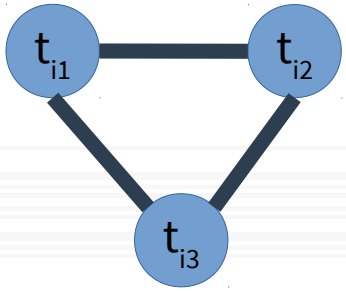
X1 = 0 X2 = 0 X3 = 0 X4 = 0, Genera E=0

X1 = 1 X2 = 0 X3 = 0 X4 = 1, Genera E=1

Reducción de 3SAT a INDEPENDENT-SET

Por cada clausula Ti = ($t_{i1} \vee t_{i2} \vee t_{i3}$)

Se crearan 3 vértices conectados entre si



Por cada $t_{ij} = x_a$, $t_{kl} = \overline{x}_a$

Crear un eje entre t_{ij} y t_{kl}

El grafo resultante G

Corresponde a una instancia del problema INDEPENDENT-SET

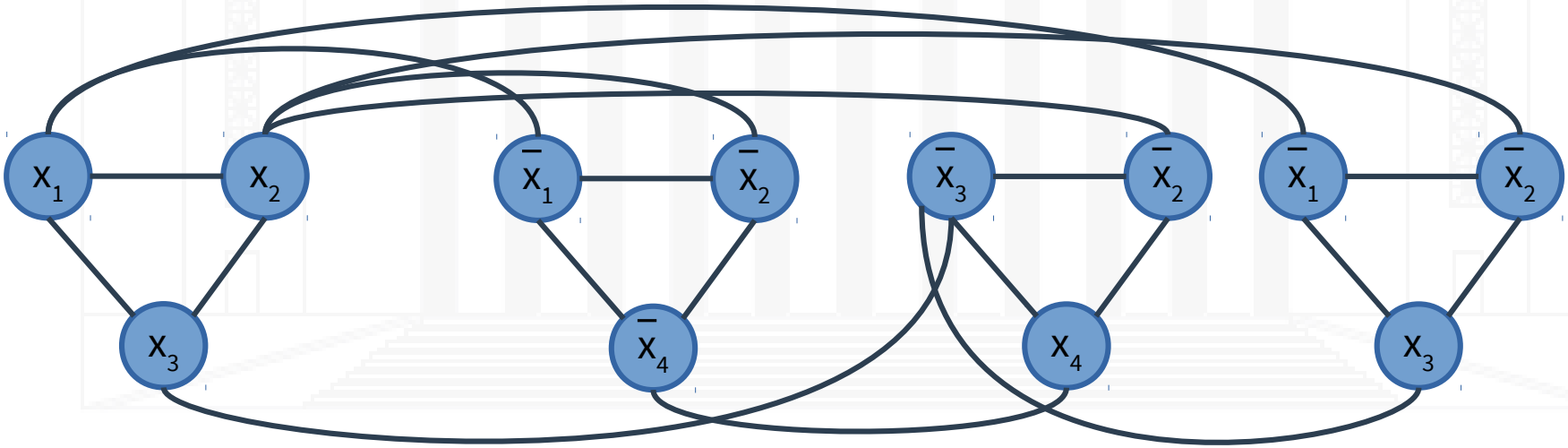
Con k=numeros de clausulas en la expresión

Ejemplo

Sea la expresión

$$E = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$$

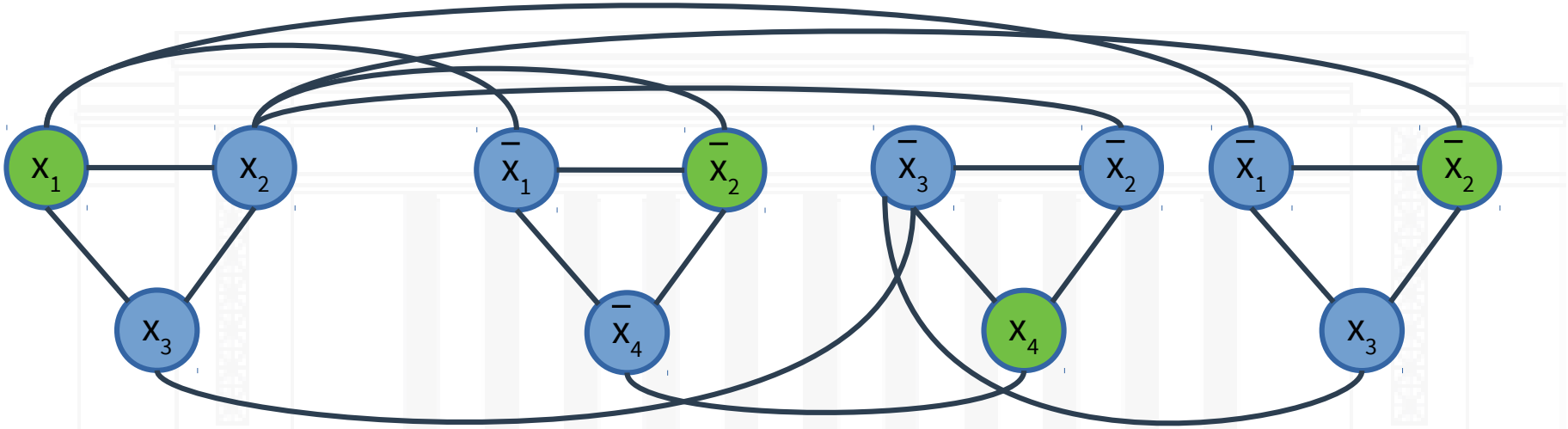
La reducimos polinomialmente a:



k=4

Ejemplo (cont.)

Si resuelvo



Entonces resuelvo 3SAT

$X_1 = 1 \quad \bar{X}_2 = 1 \rightarrow X_2 = 0 \qquad X_4 = 1 \qquad X_3 = 0$ (en este caso es indistinto 0 o 1)

INDEPENDENT-SET es NP-C

Como

INDEPENDENT-SET $\in$ NP

Y 3SAT $\leq_p$ INDEPENDENT-SET NP

Entonces

INDEPENDENT-SET $\in$ NP-C

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

NP-C: Vertex Cover

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Cobertura de Vértices

Sea

Grafo G=(V,E)

Diremos

Set $S \subseteq V$ es una cobertura de vértices

Si

$\forall$ eje $e \in E=(u,v)$, $u \in S$ y/o $v \in S$

Problema de decisión de cobertura de vértices

Sea

Grafo G=(V,E)

Determinar

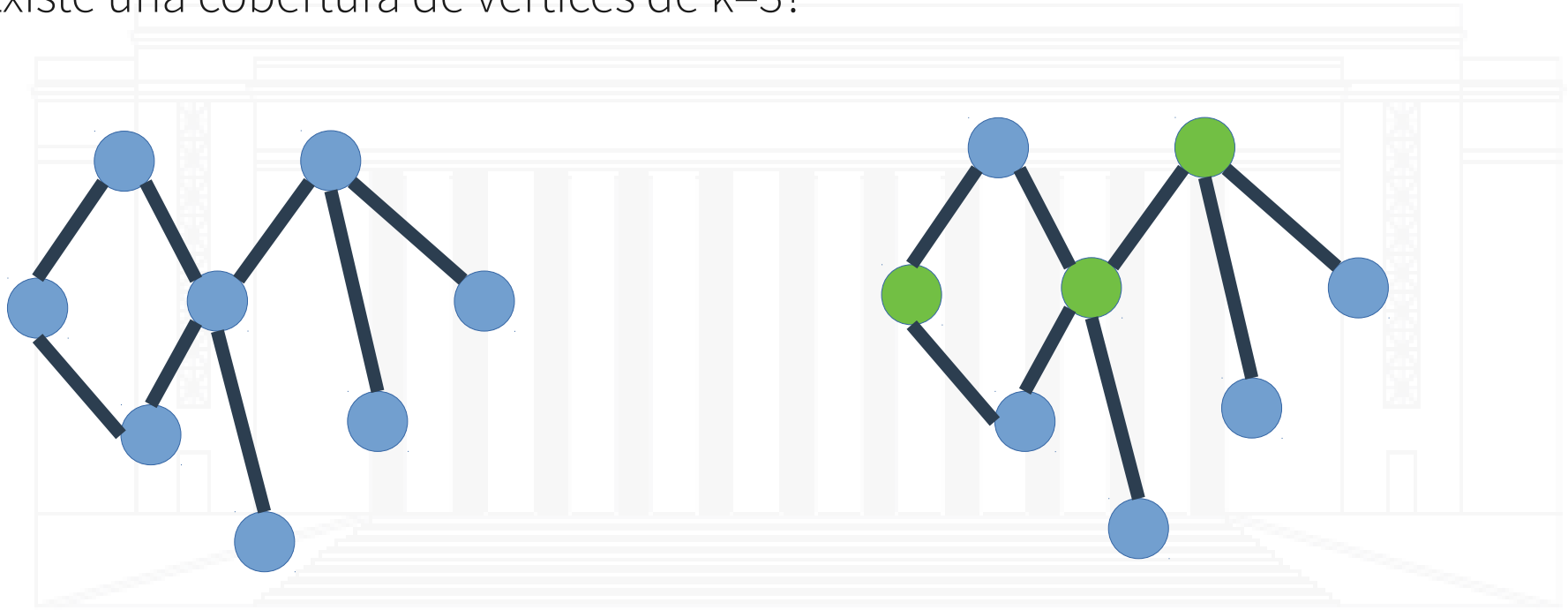
Si existe una cobertura de vértices (VERTEX-COVER) de tamaño al menos k

(El problema de optimización busca el subconjunto de menor tamaño)

Ejemplo

Dado el Grafo

Existe una cobertura de vértices de k=3?



VERTEX-COVER ∈ “NP”

Sea

Grafo G=(V,E)

Certificado t: conjunto de nodos de V que forman el cubrimiento

Verificamos

Para todo e=(u,v) ∈ E , si u ∈ t o v ∈ t → O(VE)

Si |t| = k

⇒ VERTEX-COVER ∈ “NP”

INDEPENDENT-SET

Sea

Un grafo G=(V,E)

Un valor k

Determinar

Si existe un conjunto independiente de nodos de como mucho tamaño K

Un conjunto de nodos $C \subseteq V$ es independiente si

No existe $a,b \in C$ tal que existe eje $(a,b) \in E$

Relación entre INDEPENDENT-SET y VERTEX-COVER

Sea

Grafo G=(V,E)

S conjunto independiente de tamaño |S|

Llamaremos

C = V-S (complemento de S)

Para todo eje

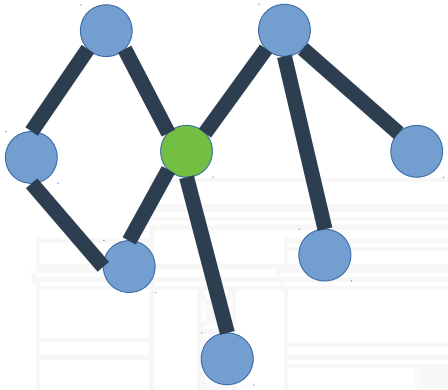
e=(u,v) $\in$ E , u $\in$ S $\Rightarrow$ v $\in$ C (porque S es set independiente)

Por lo tanto para todo eje al menos un vértice pertenece a V-S

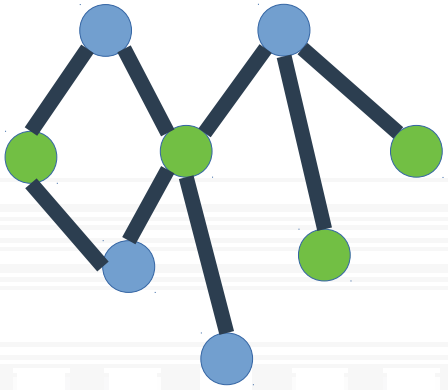
Entonces

V-S es una cobertura de vértices de G de tamaño |V-S|

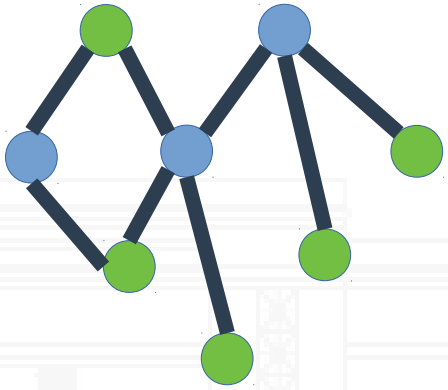
Ejemplo



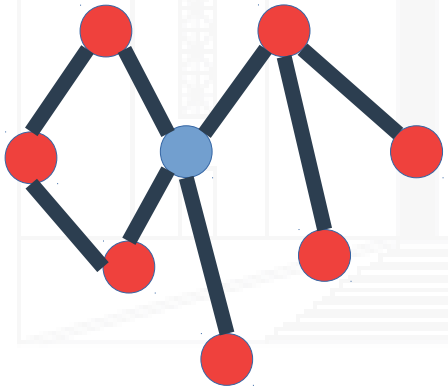
Set independiente k=1



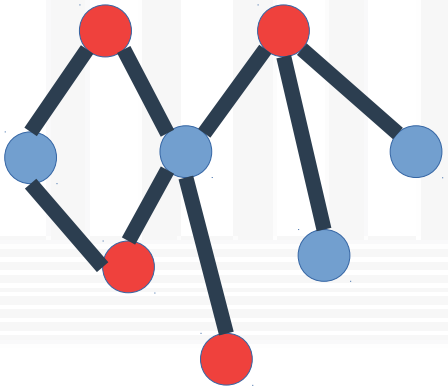
Set independiente k=4



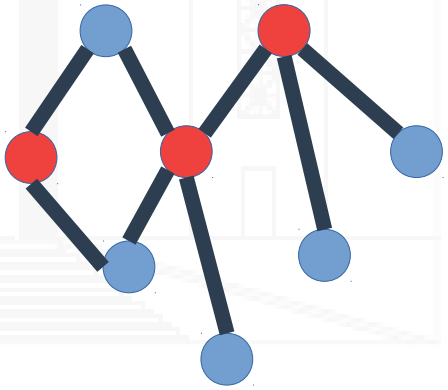
Set independiente k=5



Cobertura de vértices k=7



Cobertura de vértices k=4



Cobertura de vértices k=3

VERTEX-COVER es NP-C

INDEPENDENT-SET $\leq_p$ VERTEX-COVER

Mantengo el mismo gráfo G=(V,E)

Y tomo k' = |V|-k

De forma equivalente

VERTEX-COVER $\leq_p$ INDEPENDENT-SET

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

NP-C: Set Cover

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

SET-COVER

Sea

Un conjunto de U de n elementos

Una colección $S_1,\ldots,S_m$ de subconjuntos de U

Existe

Una colección de como mucho k de los subconjuntos cuya unión es igual a U?

Ejemplo

U = {a,b,c,d,e,f,g,h,i}

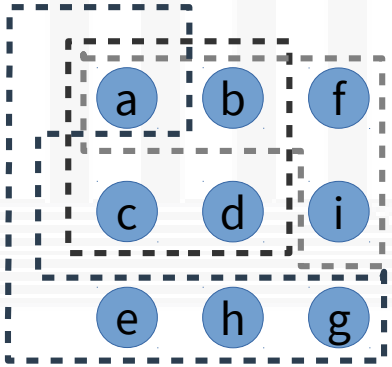
$S_1 = \{a,b,c,d\}$ $S_2 = \{a,b,f,i\}$ $S_3 = \{a,e,h,g\}$

$S_4 = \{b,c,g\}$ $S_5 = \{a,d,e\}$ $S_6 = \{g,h\}$

$S_7 = \{e,i\}$ $S_8 = \{f\}$ $S_9 = \{i\}$

Existe k=3 ?

$S_1 \cup S_2 \cup S_3 = U$



¿SET-COVER es “NP”?

Sea

U conjunto de elementos,

K tamaño buscado

los Subset $S_1,\ldots,S_m$

T certificado con subconjunto de conjuntos

Verificar

$|T| = k$

Para todo elemento en U, si existen en algunos de los subconjuntos de T

Se puede hacer en Tiempo polinomial

SET-COVER $\in$ “NP”

VERTEX-COVER

Sea

Grafo G=(V,E)

Determinar

Si existe una cobertura de vértices (VERTEX-COVER) de tamaño al menos k

Con

$\forall$ eje e $\in$ E=(u,v), u $\in$ S y/o v $\in$ S

Intentaremos demostrar que

VERTEX-COVER $\leq_p$ SET-COVER

Reducción de VERTEX-COVER a SET-COVER

Partimos de

G=(V,E) y k

Queremos

Que todos los ejes queden cubiertos

Construimos

Set de elementos U=E

Por cada Vértice $v \in V$, crearemos un subconjunto S_v con todos los ejes incidentes a él

Mantenemos en K la cantidad de subconjuntos a buscar para cubrir U

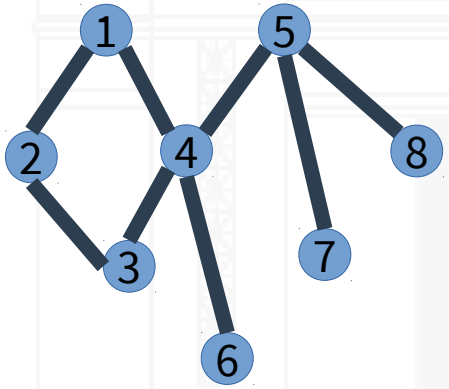
Si

Encontramos el subconjunto, eso nos dirá que vértices seleccionar.

Ejemplo

Sea el problema

Vertex-cover (k=3)



Reducimos a

Set Cover (k=3)

U= {1-2, 1-4, 2-3, 3-4, 4-5, 4-6, 5-7, 5-8}

S_1 = {1-2, 1-4}

S_2 = {1-2, 2-3}

S_3 = {2-3, 3-4}

S_4 = {1-4, 3-4, 4-5, 4-6}

S_5 = {4-5, 5-7, 5-8}

S_6 = {4-6}

S_7 = {5-7}

S_8 = {5-8}

SET-COVER ∈ NP-C

Reducimos

Vertex-cover a set-cover en tiempo polinomial

Si resolvemos cualquier instancia de set-cover, podemos resolver cualquier instancia de vertex-cover

Por lo tanto

SET-COVER ∈ NP-C

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

NP-C: 3 Dimensional Matching

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

3 Dimensional Matching

Dados

3 sets disjuntos X.Y,Z de tamaño n cada uno.

Un set $C \subseteq X,Y,Z$ de triplas ordenadas

Determinar

Si existe un subset de n triplas en C tal que cada elemento de $X \cup Y \cup Z$ sea contenido exactamente en una de esa triplas?

Ejemplo

Ver si es posible asignar un chofer, auto y pasajeros según preferencias

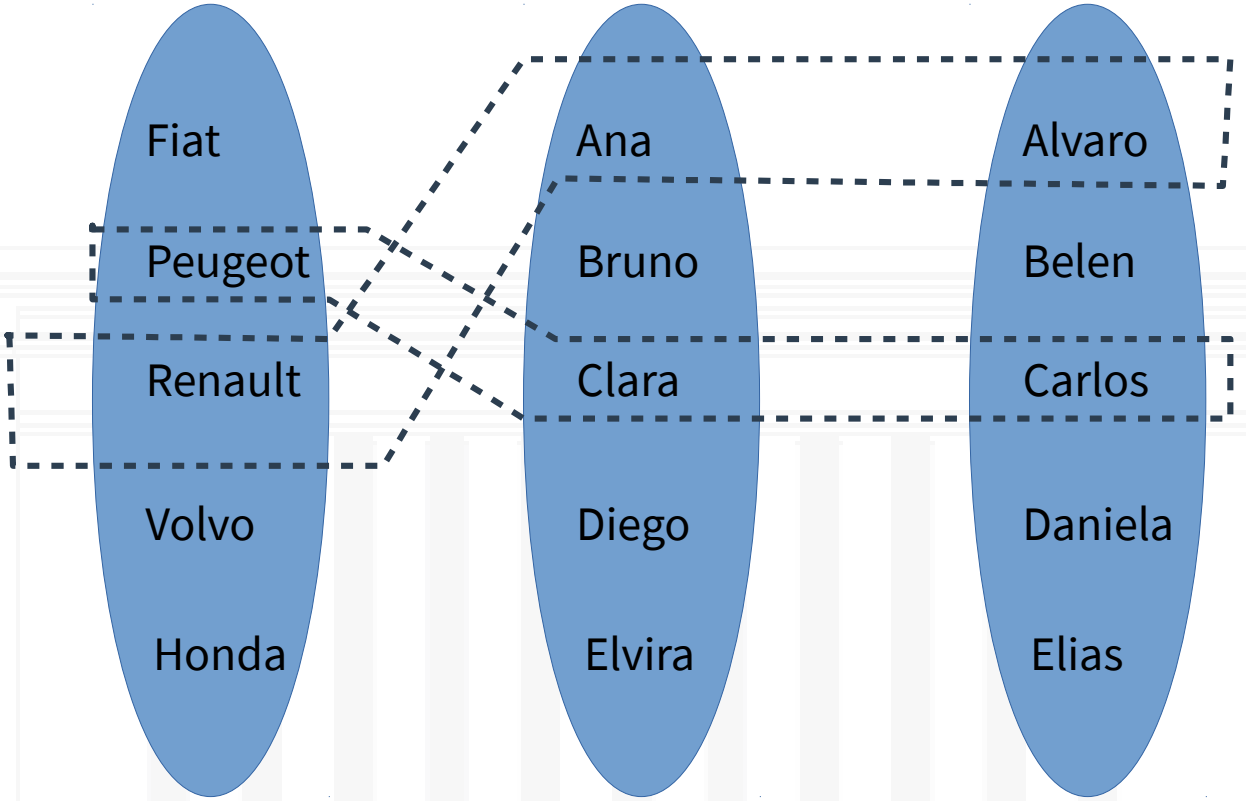
Autos={Fiat, Peugeot, Renault, Volvo, Honda}

Choferes={Ana, Bruno, Clara, Diego, Elvira}

Pasajeros={Alvaro, Belen, Carlos, Daniela, Elias}

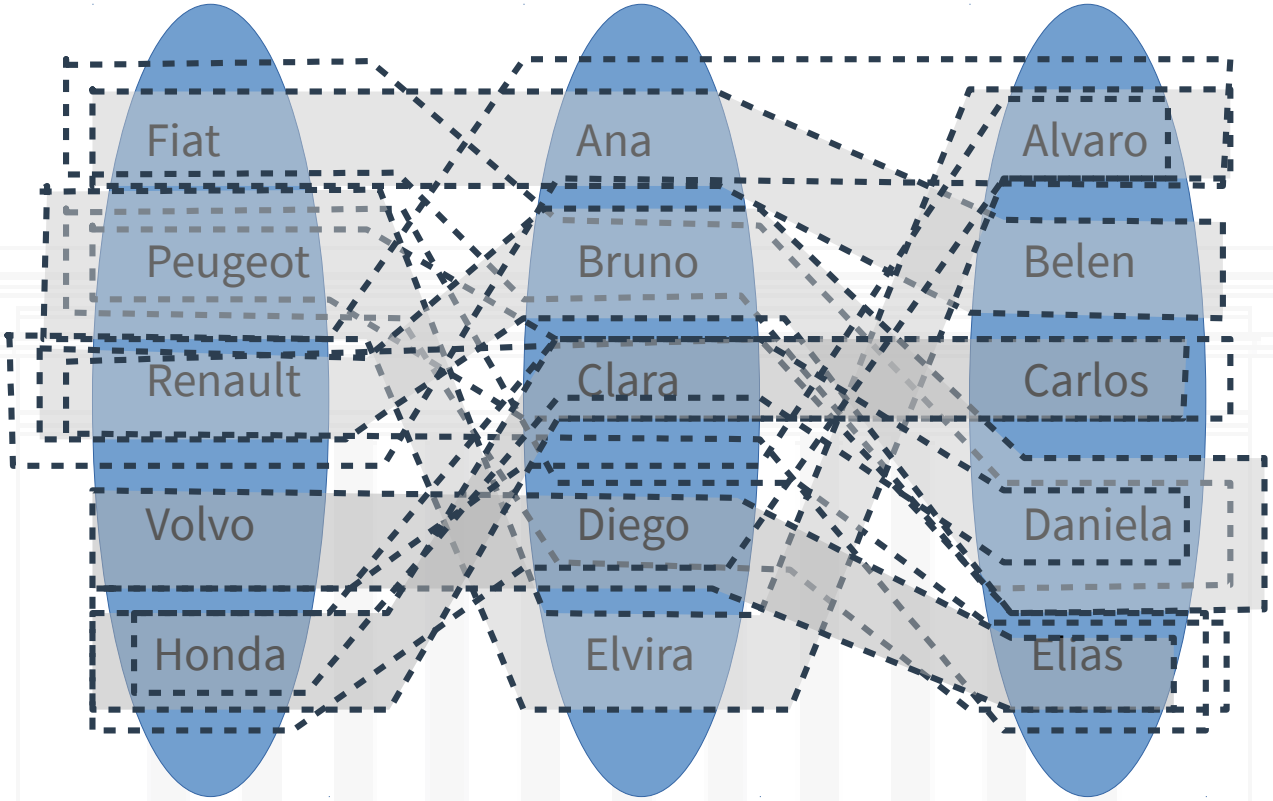
Posibles equipos = { (Fiat,Ana,Belen), (Fiat,Bruno,Daniela), (Peugeot,Clara,Carlos), (Peugeot,Diego,Elias), (Peugeot,Elvira,Alvaro), (Renault,Bruno,Daniela), (Renault,Ana,Alvaro), (Renault,Clara,Elias), (Volvo,Diego,Elias), (Honda,Clara,Carlos), (Honda,Clara,Daniela), (Honda,Diego,Alvaro) }

Ejemplo (cont)



Posibles equipos = { (Fiat,Ana,Belen), (Fiat,Bruno,Daniela), (Peugeot,Clara,Carlos), (Peugeot,Diego,Elias), (Peugeot,Elvira,Alvaro), (Renault,Bruno,Daniela), (Renault,Ana,Alvaro), (Renault,Clara,Elias), (Volvo,Diego,Elias), (Honda,Clara,Carlos), (Honda,Clara,Daniela), (Honda,Diego,Alvaro) }

Ejemplo (cont)

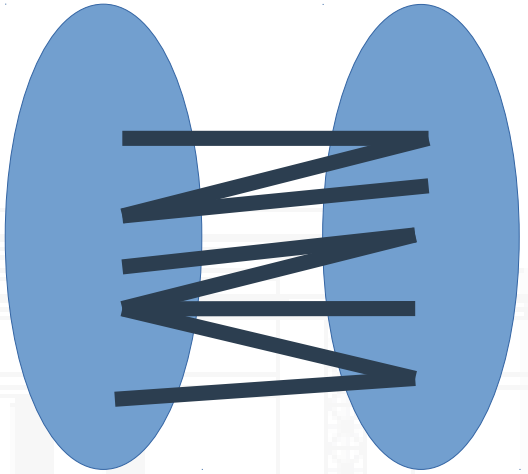


Posibles equipos = { **(Fiat,Ana,Belen)**, (Fiat,Bruno,Daniela), (Peugeot,Clara,Carlos), (Peugeot,Diego,Elias), **(Peugeot,Elvira,Alvaro)**, **(Renault,Bruno,Daniela)**, (Renault,Ana,Alvaro), (Renault,Clara,Elias), **(Volvo,Diego,Elias)**, **(Honda,Clara,Carlos)**, (Honda,Clara,Daniela), (Honda,Diego,Alvaro) }

3DM: variante de 2DM

2 Dimensional Macthing

(También conocido como bipartite Mathing)



Versión de decisión

Existe un subconjunto de tamaño máximo que empareje a todos los elementos de los 2 conjuntos

Existe un algoritmo polinomial que lo resuelve

Analizamos uno cuando trabajamos redes de flujo

¿ 3DM ∈ “NP”?

Dado

X,Y,Z conjuntos de n elementos

C=(x,y,z) conjunto de triplas

T certificado, triplas con un subconjunto de C

Podemos certificar en tiempo polinomial

|T| igual a n

Todo elemento en X,Y y Z, se encuentra 1 y solo 1 vez en algun T_i

$\Rightarrow$ 3DM ∈ NP

¿ 3DM ∈ “NP-Hard”?

Probaremos que

$3SAT \leq_p 3DM$

Sea

I instancia de 3SAT

con n variables = $\{x_1,...x_n\}$

y k clausulas = $\{c_1,\ldots,c_k\}$

Reduciremos en tiempo polinomial

La instancia I a un problema de 3DM

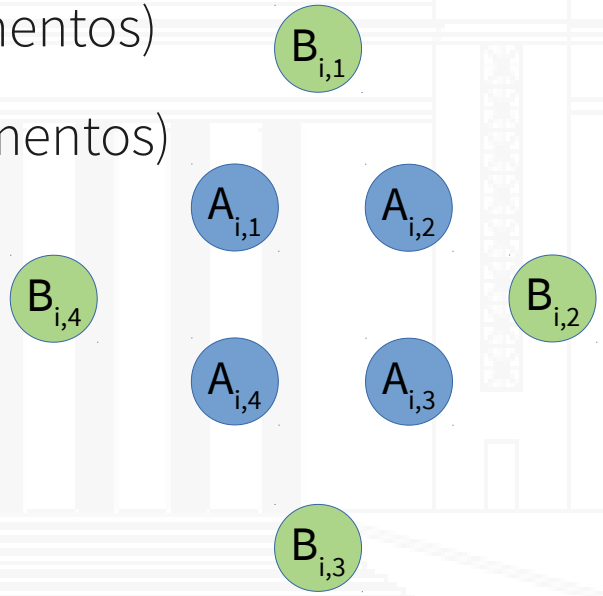
Reducción de 3SAT a 3DM

Por cada variable Xi

Crearemos un “gadget” formado por los siguientes elementos:

$A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,2k}\}$ ← nucleo del gadget (2k elementos)

$B_i = \{b_{i,1}, b_{i,2}, \ldots, b_{i,2k}\}$ ← puntas del gadget (2k elementos)



Variable i: ejemplo para k=2 clausulas

Reducción de 3SAT a 3DM (cont.)

Por cada variable Xi

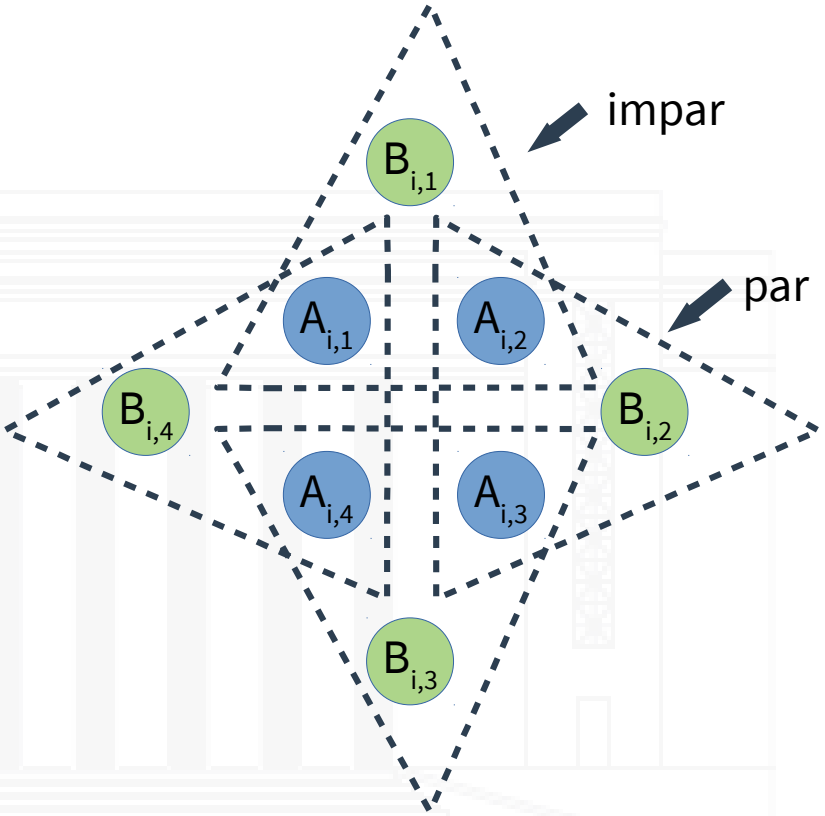
Crearemos las triplas:

$t_{ij} = \{a_{i,j}, a_{i,j+1}, b_{i,j}\}$

Llamaremos

Tripla par, si j es par

Tripla inpar si j es impar



Variable i: ejemplo para k=2 clausulas

Reducción de 3SAT a 3DM (cont.)

Por cada clausula Cj

Crearemos un set de elementos núcleo:

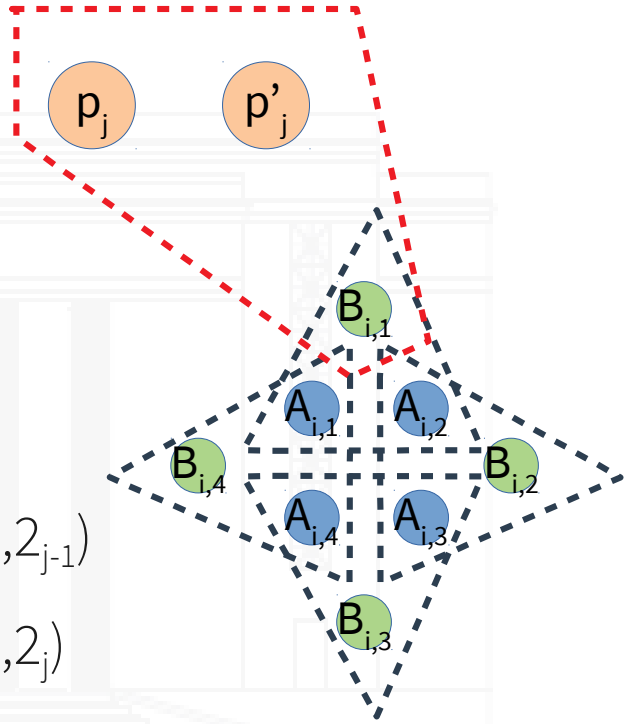
Cj = {p_j, p'_j}

Por cada variable i en la clausula Cj

Si contiene la variable $\bar{x}_i$ → Crearemos un tripla (p_j, p'_j,bi,2_{j-1})

Si contiene la variable x_i → Crearemos un tripla (p_j, p'_j,bi,2_j)

(cada clausula tendrá 3 triplas)



Clausula 1: con la variable $\bar{x}i$

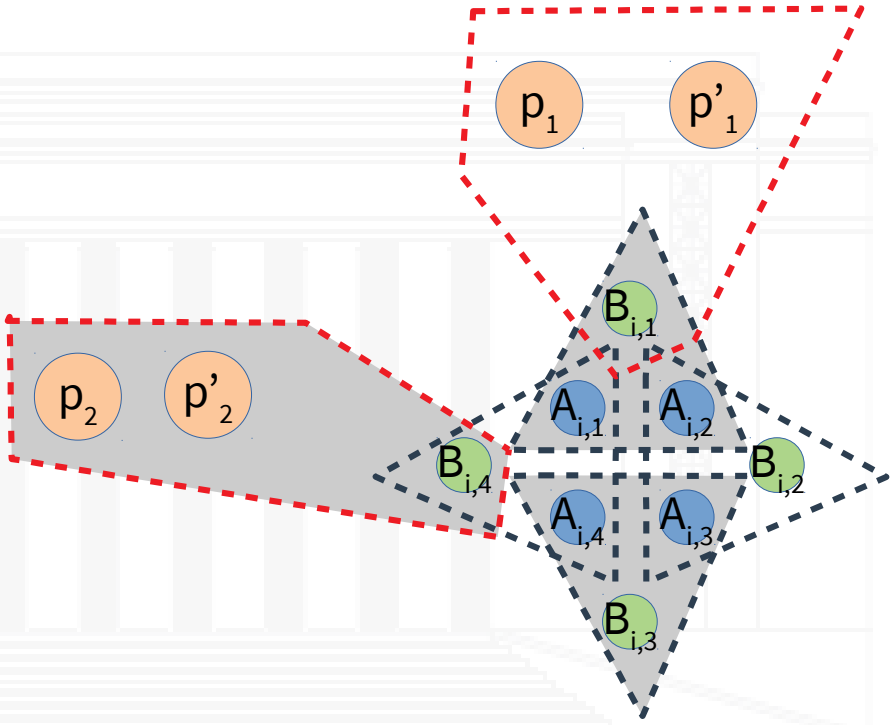
Reducción de 3SAT a 3DM (cont.)

Si una variable i en la solución esta en 1 (x_i=1)

Las puntas del gadget i correspondientes a su valor 0 estarán cubiertas por las triplas de su nucleo.

Las puntas correspondientes a su valor 1, pueden usarse para activar clausulas

(lo mismo aplica para la variable en 0)

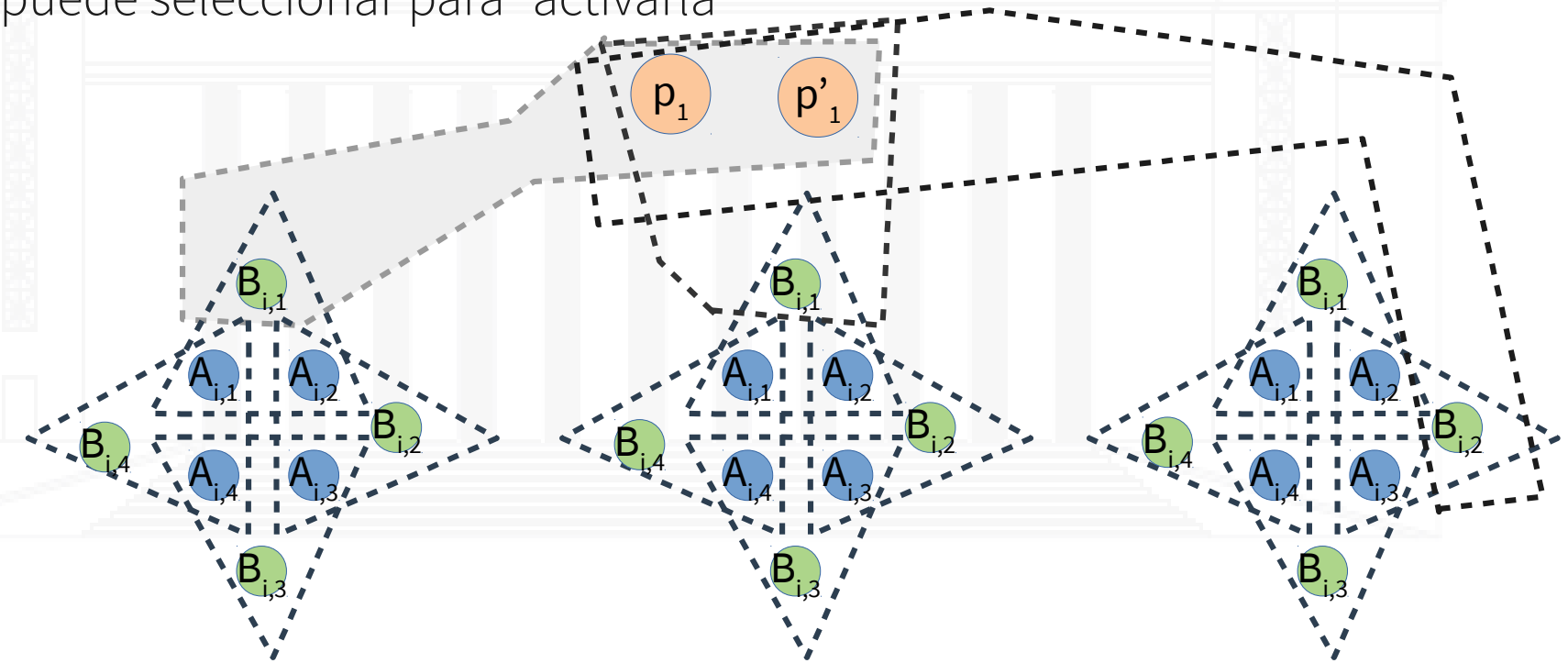


Reducción de 3SAT a 3DM (cont.)

Cada clausula

Tiene 3 triplas asociadas

Solo 1 se puede seleccionar para “activarla”



Reducción de 3SAT a 3DM (cont.)

En total hay 2*n*k puntas, si hay solución

- Las triplas de la clausulas cubren k de ellas
- Las triplas de los gadget cubren nk puntas
- Faltan cubrir (n-1)k puntas

Agregaremos un tipo de gadget

- “Cleanup gadgets”

Reducción de 3SAT a 3DM (cont.)

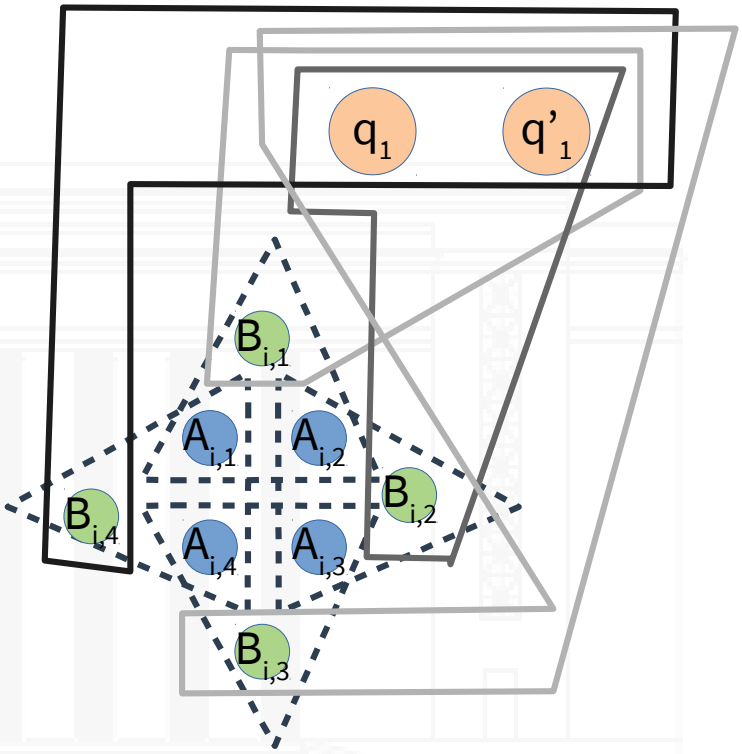
Construiremos (n-1)k

Cleanup gadgets

Cada Cleanup gadget i

Tendrá los elementos Qi = $\{q_i, q'_1\}$

Agregaremos las triplas $\{q_i, q'_1, b\}$ con b son todas las puntas de los gadgets



Ejemplo de cleanup gadget (solo mostrando las triplas en 1 gadget de variable)

Reducción de 3SAT a 3DM (cont.)

Para terminar

Necesitamos construir los conjuntos disjuntos X,Y,Z

Conjunto X

a_{ij} con j par (de los widgets variables) → nk

p_j (de los widget clausula) → k

q_j (de los widget cleanups) → (n-1)k

Conjunto Y

a_{ij} con j impar (de los widgets variables) → nk

p'_j (de los widget clausula) → k

q'_j (de los widget cleanups) → (n-1)k

Conjunto Z

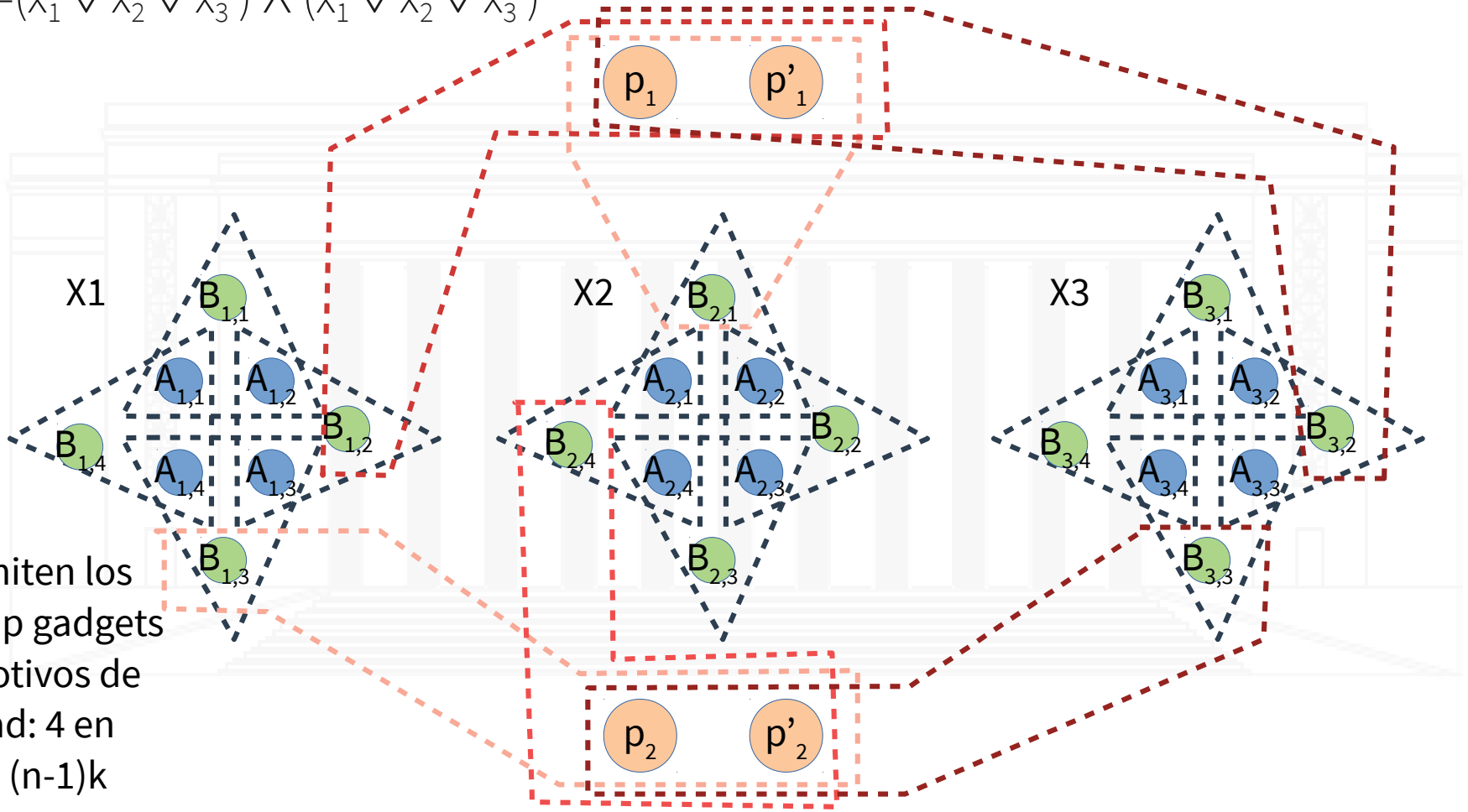
Todos los b_{ij} (de los widgets variables) → 2nk

Triplas "C"

serán todas las triplas definidas

Ejemplo

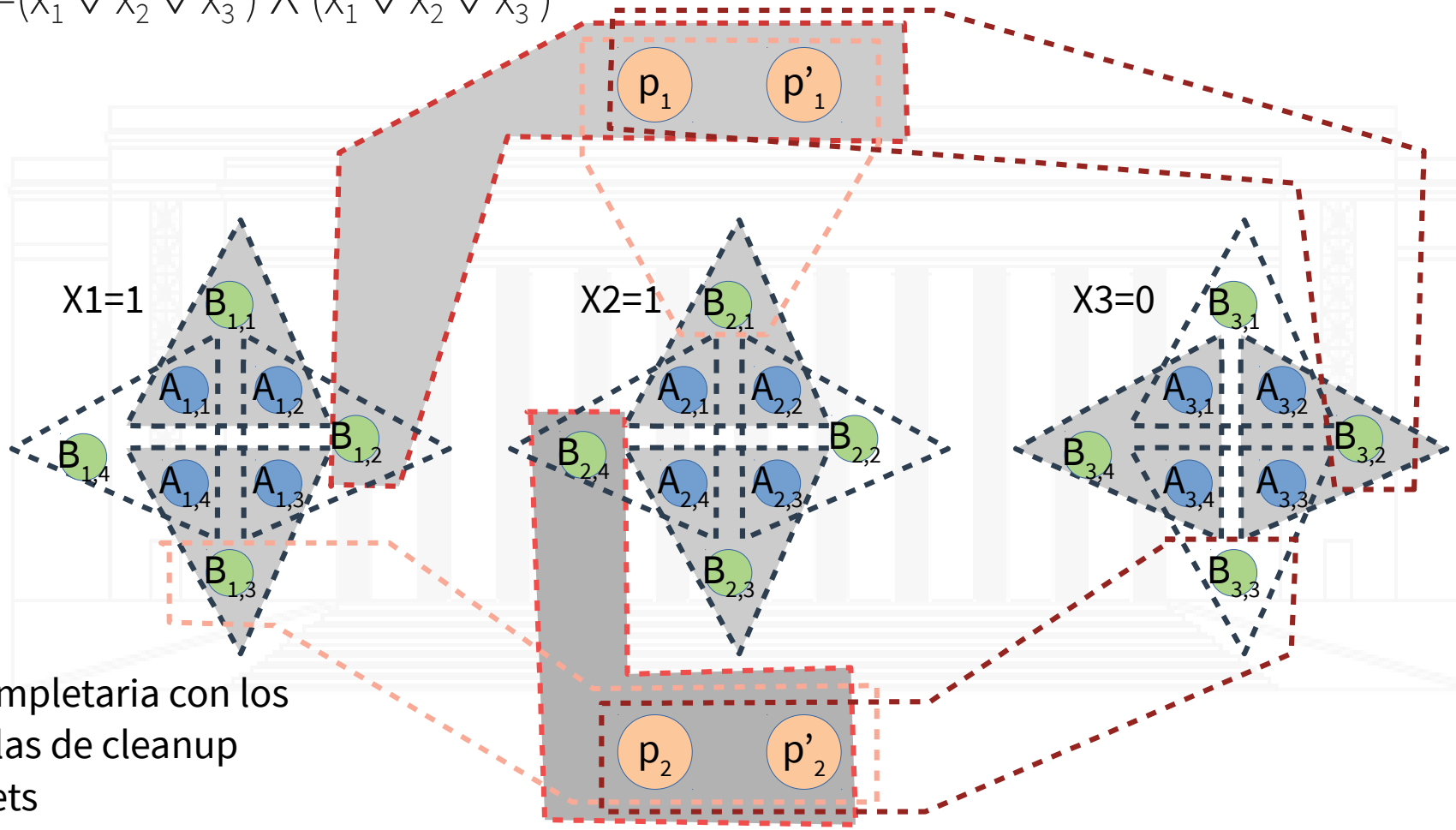
$E = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$



*Se omiten los cleanup gadgets por motivos de claridad: 4 en total → (n-1)k

Ejemplo

$E = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$



Se completaria con los 4 triplas de cleanup gadgets

3DM es NP-C

Como

3DM $\in$ NP

Y 3SAT $\leq_p$ 3DM

Entonces

3DM $\in$ NP-C

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

NP-C: Ciclo Hamiltoniano

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Ciclo Hamiltoneano

Sea

G = (V,E) grafo direccionado

Definimos

Un ciclo C en G como hamiltoneano

Si

Visita cada vértice 1 y solo 1 vez

Comienza y termina por el mismo vértice

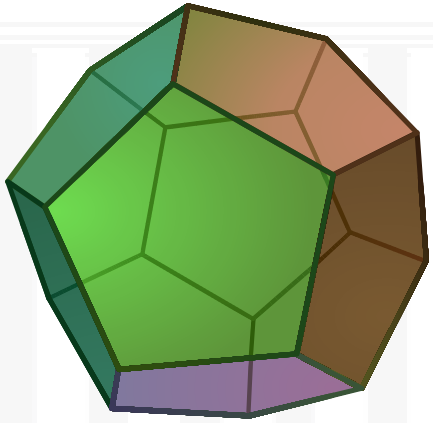
Ciclo Hamiltoneano (cont.)

Recibe su nombre

Por Sir William Rowan Hamilton ,

En 1857

Inventa el Juego “Icosian”



Dado un dodecaedro

Encontrar un ciclo que recorra todos los vértices, iniciando y finalizando por el mismo vértice

Problema de decisión de Ciclo Hamiltoneano

Sea

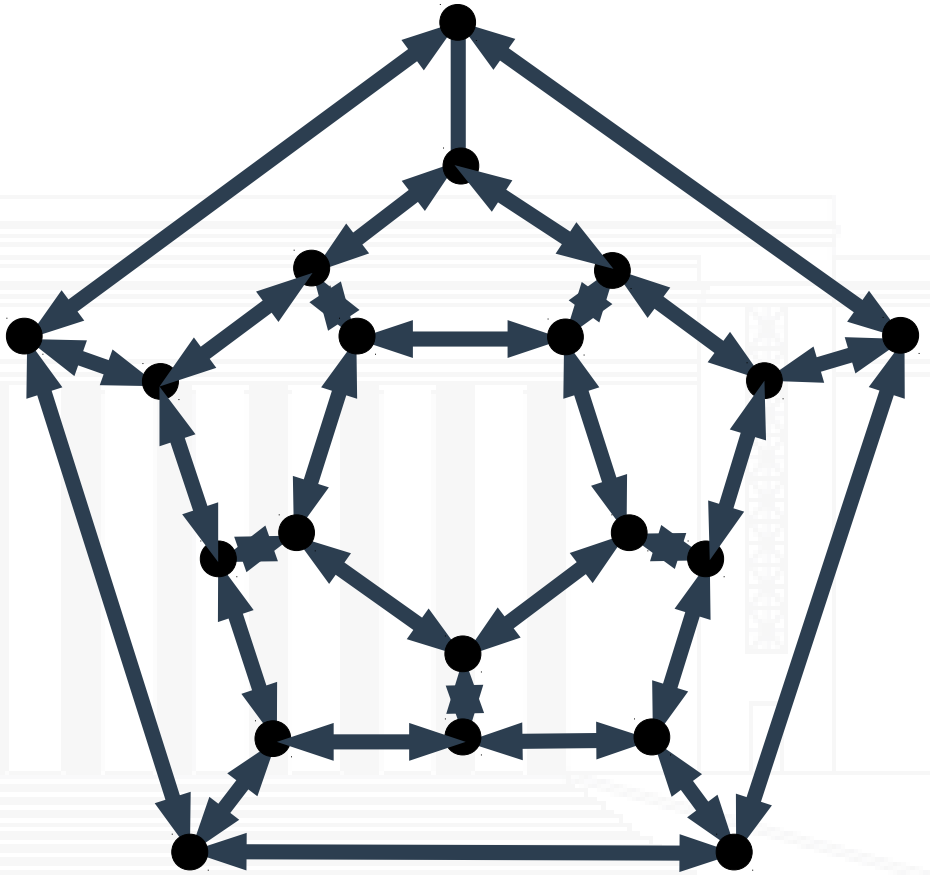
G = (V,E) grafo direccionado

Existe

Un ciclo hamiltoneano?

Llamaremos al problema

HAM-CYCLE



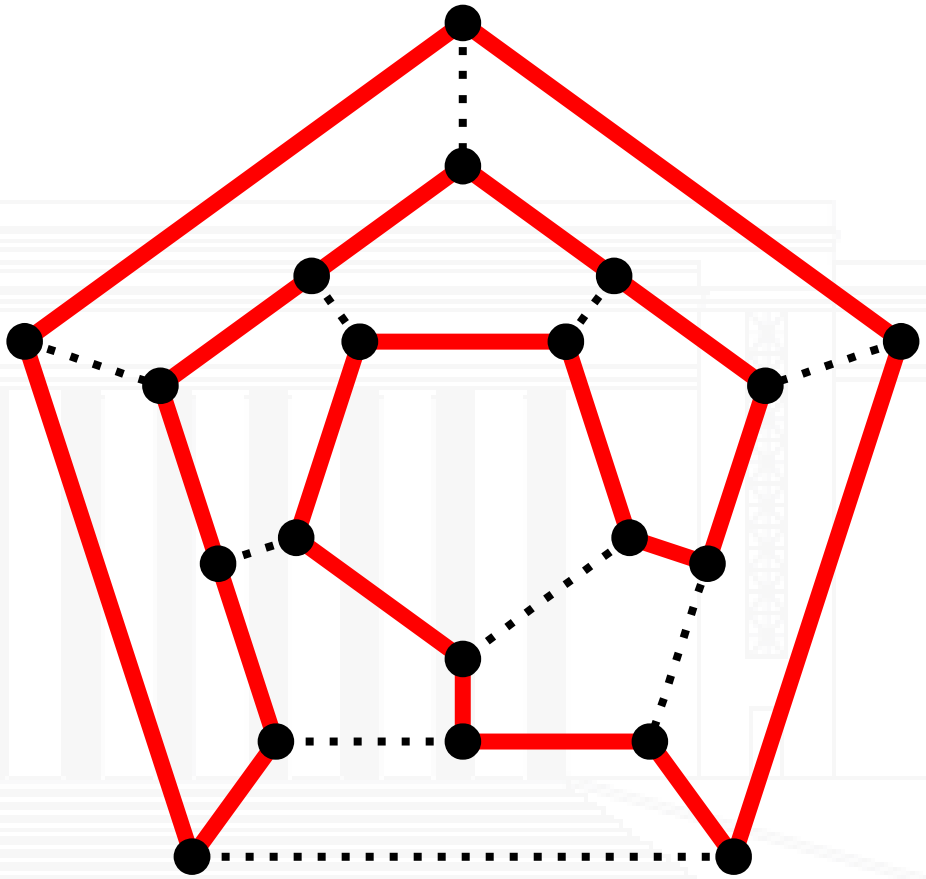
Problema de decisión de Ciclo Hamiltoneano

Sea

G = (V,E) grafo direccionado

Existe

Un ciclo hamiltoneano?



HAM-CYCLE ∈ "NP"

Dado

G=(V,E)

T certitifcado = $\{t_0,...t_{|V|}\}$ lista ordenada de vértices

Puedo verificar (en tiempo polinomial)

$|T| = |V|$, $t_0=t_{|V|}$

Todos los vértices de V están en T

Para todo t_i $t_{i+1} \in T$, $(t_i,t_{i+1}) \in E$

$\Rightarrow$ HAM-CYCE ∈ NP

¿HAM-CYCLE es “P”?

No se conoce algoritmo

Que resuelva HAM-CYCLE en tiempo polinómico

Si probamos que

HAM-CYCLE $\in$ NP-C

(Utilizaremos 3SAT)

Entonces

HAM-CYCLE $\in$ P $\Leftrightarrow$ P = NP

3SAT

Dado

$X=\{x_1,...x_n\}$ conjunto de n Variables booleanas = {0,1}

k clausulas booleanas $Ti = (t_{i1} \vee t_{i2} \vee t_{i3})$

Con cada $t_{ij} \in X \cup \overline{X} \cup \{1\}$

Determinar

Si existe asignación de variables tal que $T_1 \wedge T_2 \wedge ... \wedge T_k = 1$

Reducción de 3SAT a HAM-CYCLE

Para I instancia de3SAT

Con n variables $x_1,\ldots,x_n$

Y k clausulas $c_1,\ldots,C_k$

(existen 2^n asignaciones de variables posibles)

Construiremos

Un grafo G=(V,E) donde encontrar el ciclo hamiltoneano.

Reducción de 3SAT a HAM-CYCLE (cont.)

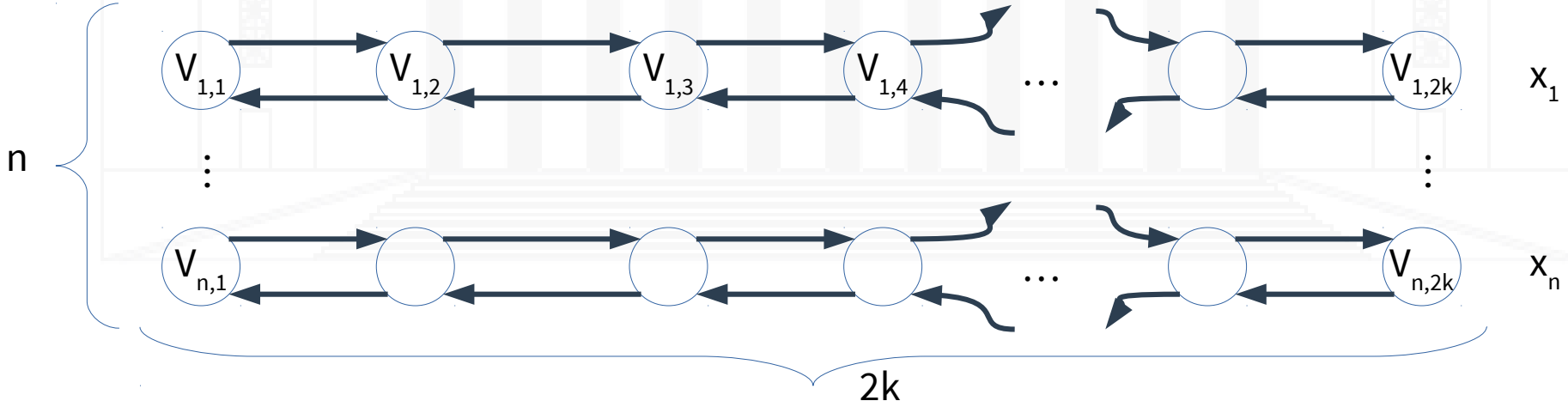
Construimos n caminos p1, …, pn

cada uno representa a una variable

Cada camino estará conformado por 2*k nodos

Unidos entre si por aristas de ida y vuelta

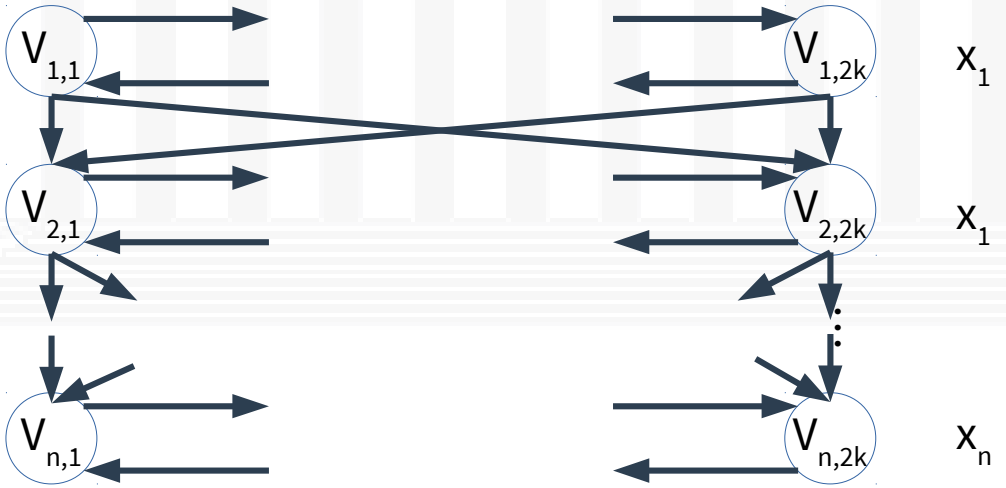
(cada 2 nodos corresponden a la variable en una clausula)



Reducción de 3SAT a HAM-CYCLE (cont.)

Uniremos cada camino

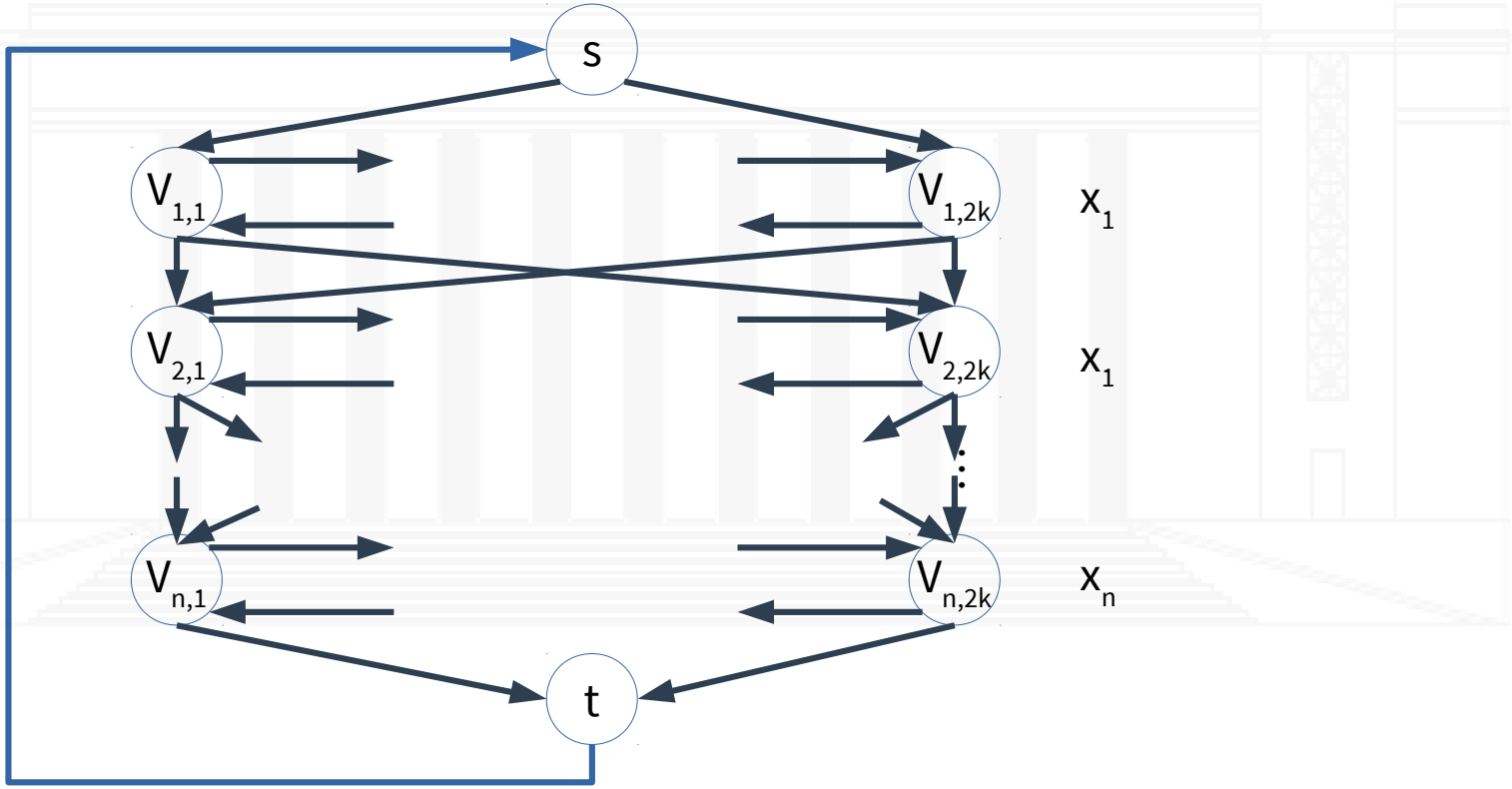
- Desde su nodo inicial hasta el nodo inicial del camino siguiente
- Desde su nodo inicial hasta el nodo final del camino siguiente
- Desde su nodo final hasta el nodo inicial del camino siguiente
- Desde su nodo final hasta el nodo final del camino siguiente



Reducción de 3SAT a HAM-CYCLE (cont.)

Creamos nodos: s y t

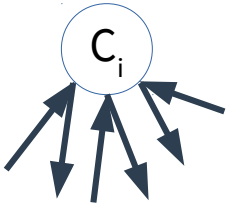
Unimos con los ejes s-$v_{1,1}$, s-$v_{1,2k}$, $v_{n,2k}$-t , $v_{n,2k}$-t, t-s



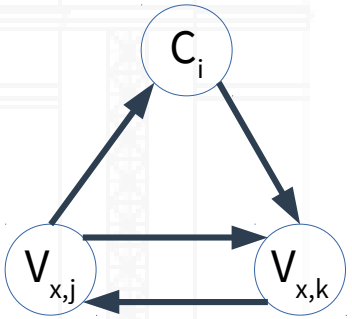
Reducción de 3SAT a HAM-CYCLE (cont.)

Creamos 1 nodo por cada clausula: Ti = ($t_{i1} \vee t_{i2} \vee t_{i3}$)

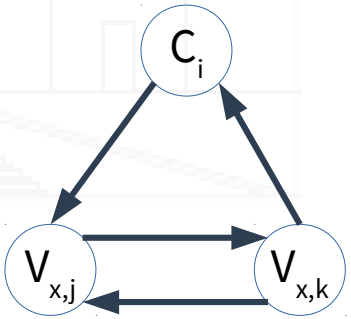
Cada clausula tendrá 3 pares de ejes



Variable sin negar



Variable negada



Para cada variable x de la clausula i

Se une 2 nodos del camino de la variable x con el nodo de la clausula i

en la posición j=i*2 y k=i*2+1

(el sentido de los ejes depende si la variable esta negada o no)

Existencia de camino Hamiltoneano

Si existe un camino hamiltoneano

Que parte de s y llegue a t

(regresando a s desde t para conformar el ciclo)

Pasando por todos los nodos (y por lo tanto activando las clausulas)

Entonces hay forma de satisfacer la expresión booleana

El sentido por el que se recorre el camino que representa la variable determina si su valor es 0 o 1

El eje seleccionado para acceder al nodo “clausula” nos dice que variable la activa

Si una variable requiere estar negado y no estarlo para activar varias clausulas, el ciclo es imposible de construir

Ejemplo

Si tengo la expresión

$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4)$

Con 4 variables y 3 clausulas

Tendré

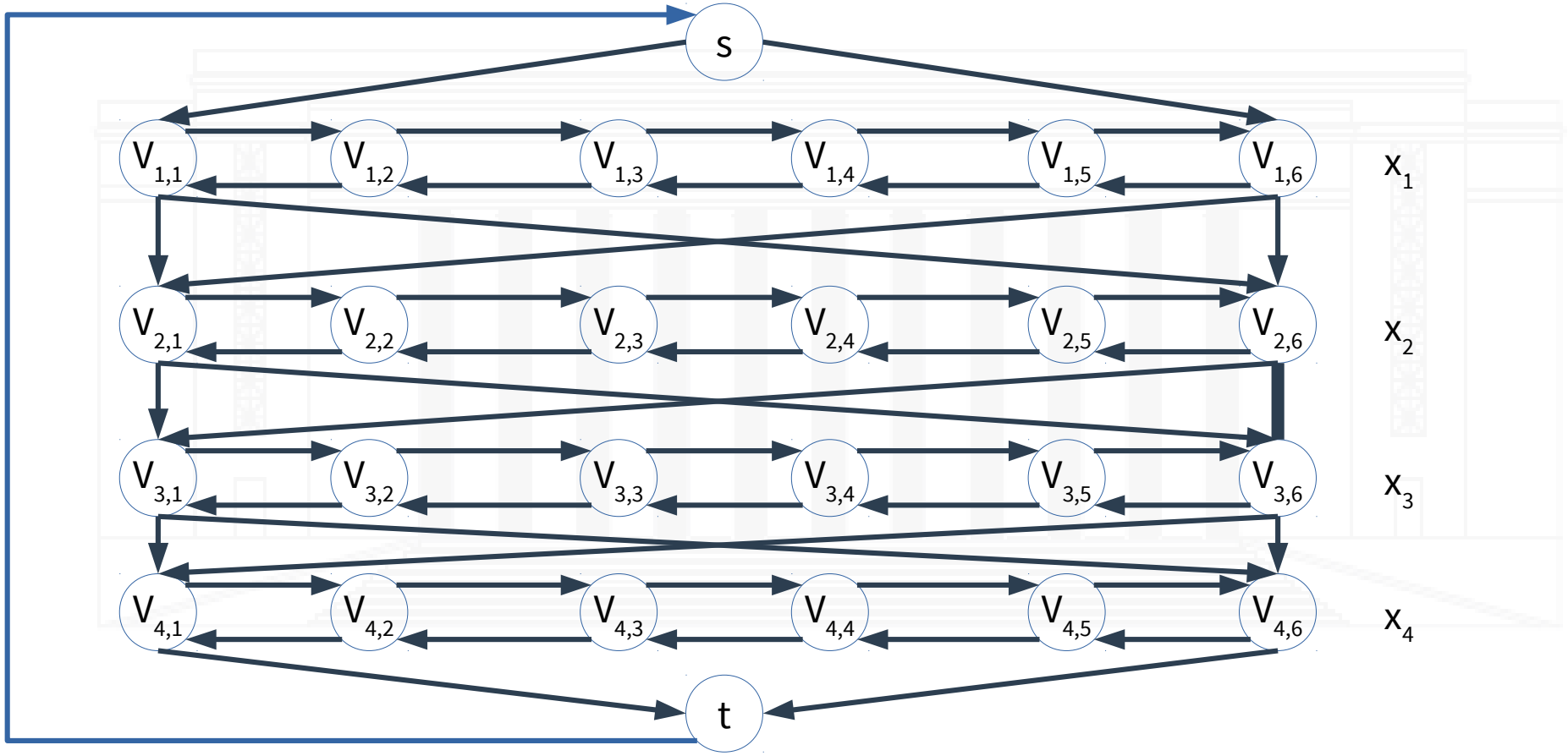
4 caminos (n=4)

3*2 = 6 nodos por caminos

3 nodos clausulas

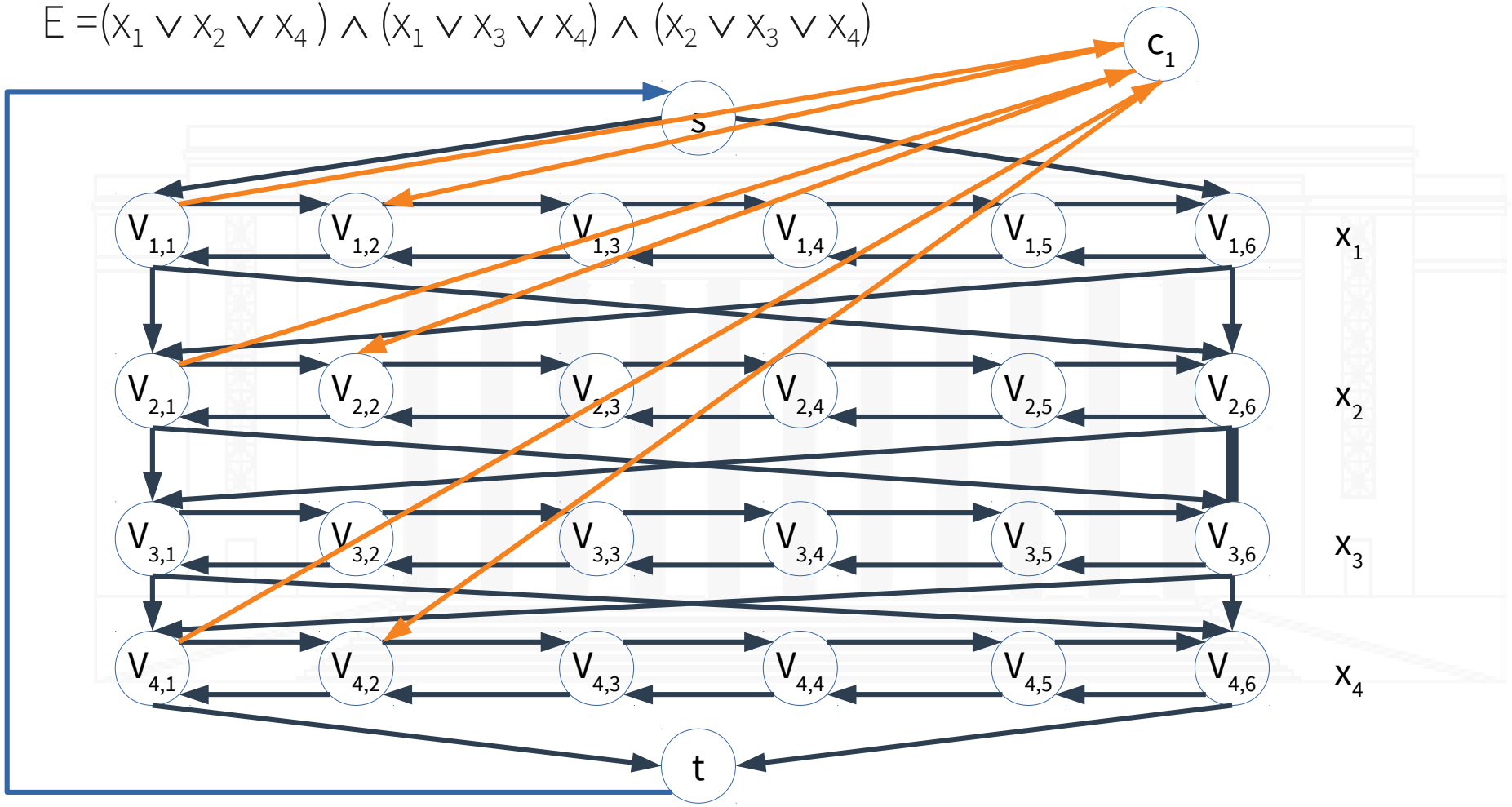
Ejemplo (cont.)

$$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4)$$



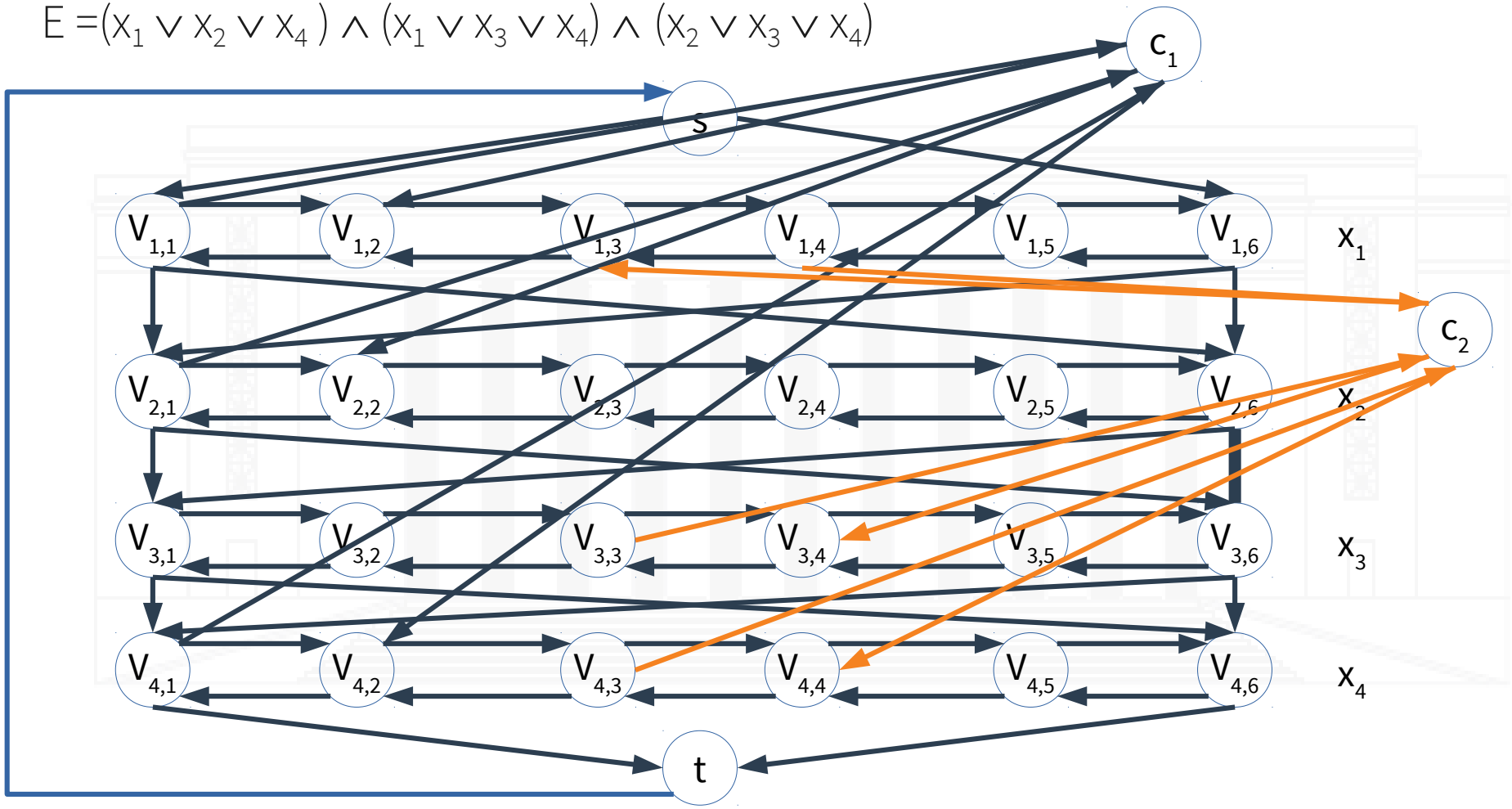
Ejemplo (cont.)

$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4)$



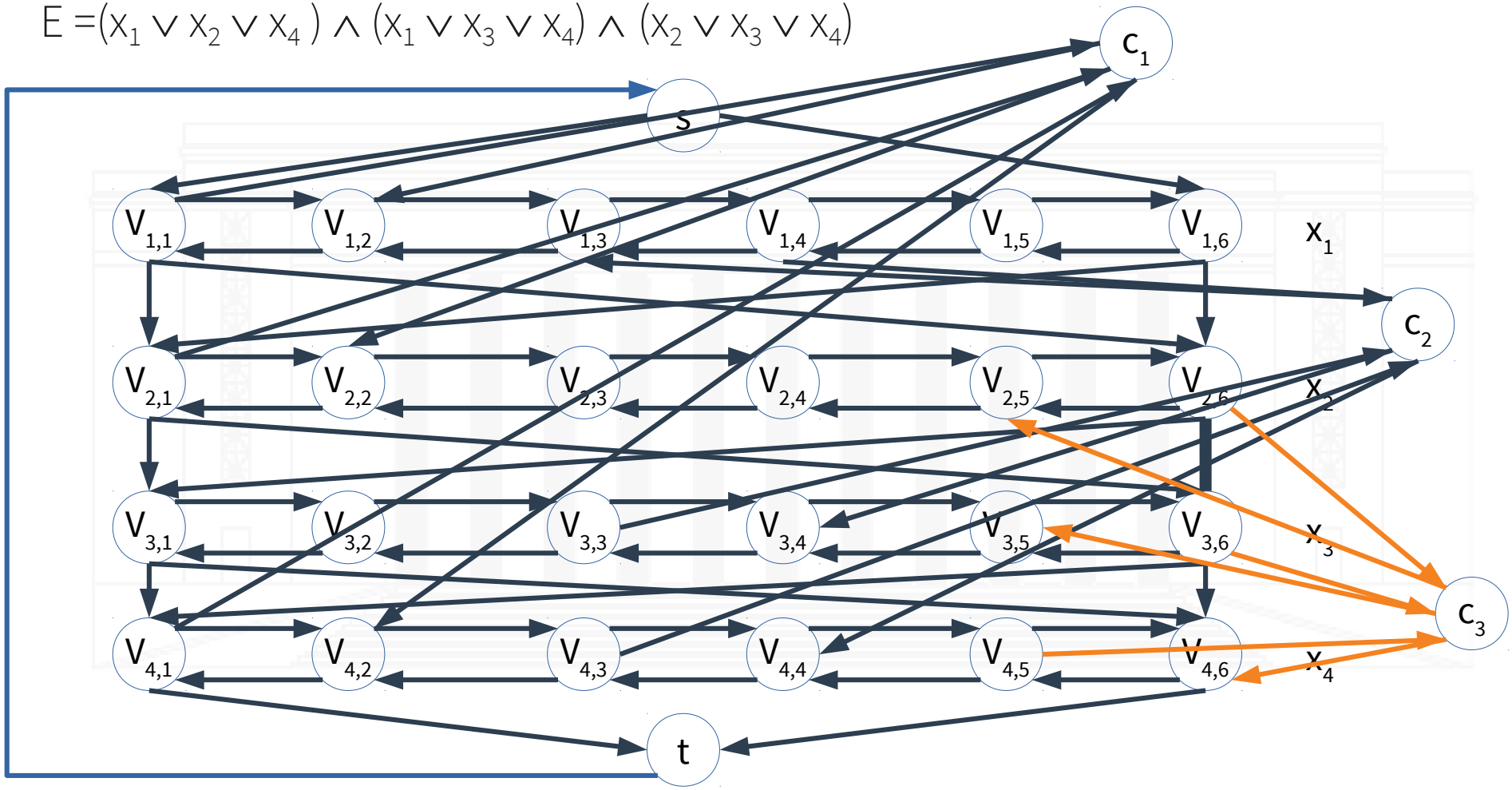
Ejemplo (cont.)

$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4)$



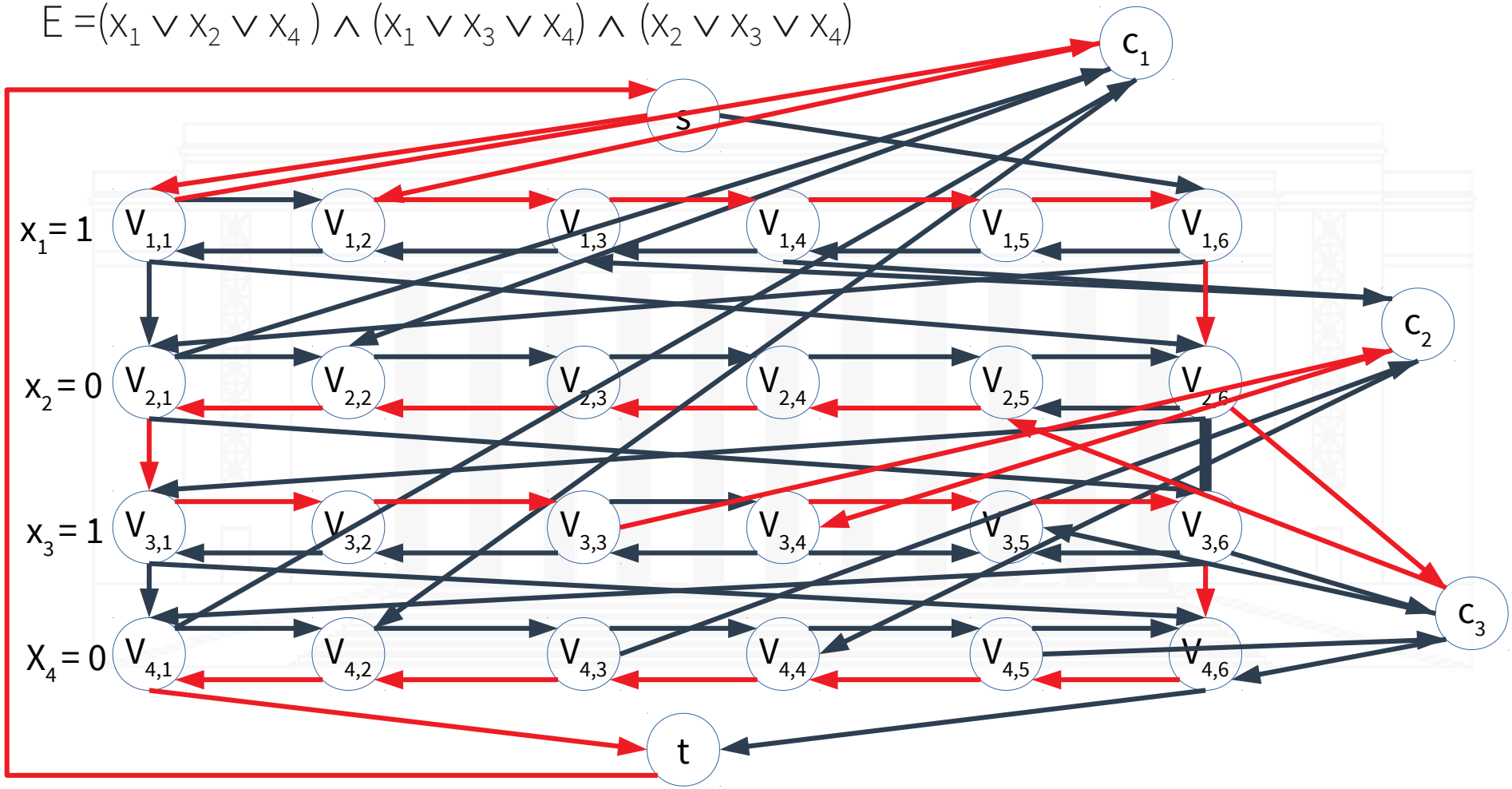
Ejemplo (cont.)

$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4)$



Ejemplo (cont.)

$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4)$



HAM-CYCLE es NP-C

Como

HAM-CYCLE $\in$ NP

Y 3SAT $\leq_p$ HAM-CYCLE NP

Entonces

HAM-CYCLE $\in$ NP-C

De igual manera (quitando en la transformación el eje t-s)

Puedo demostrar que HAM-PATH es NP-C

HAM-CYCLE para grafos no dirigidos

Utilizando grafos dirigidos

Se conoce al problema como DIRECTED HAMILTONIAN CYCLE

Utilizando grafos no dirigidos

Se conoce al problema como UNDIRECTED HAMILTONIAN CYCLE

Karp demostró en 1972

Que ambos problemas son NP-C

(utilizando otro camino)

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

NP-C: Problema del viajante

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Problema del viajante de comercio

Un viajante debe

Recorrer n ciudades $v_1, v_2, \ldots, v_n$

Partiendo de v1 se debe construir un tour visitando cada ciudad una vez y retornar a la ciudad inicial

Para cada par de ciudades vx, vy

Se especifica una distancia $d(v_x, v_y)$

No necesariamente hay simetria: $d(v_x, v_y)$ puede ser diferente a $d(v_y, v_x)$

No necesariamente se cumple la desigualdad triangular: $d(v_i, v_j) + d(v_j, v_k) > d(v_i, v_k)$

Problema decisión del viajante de comercio

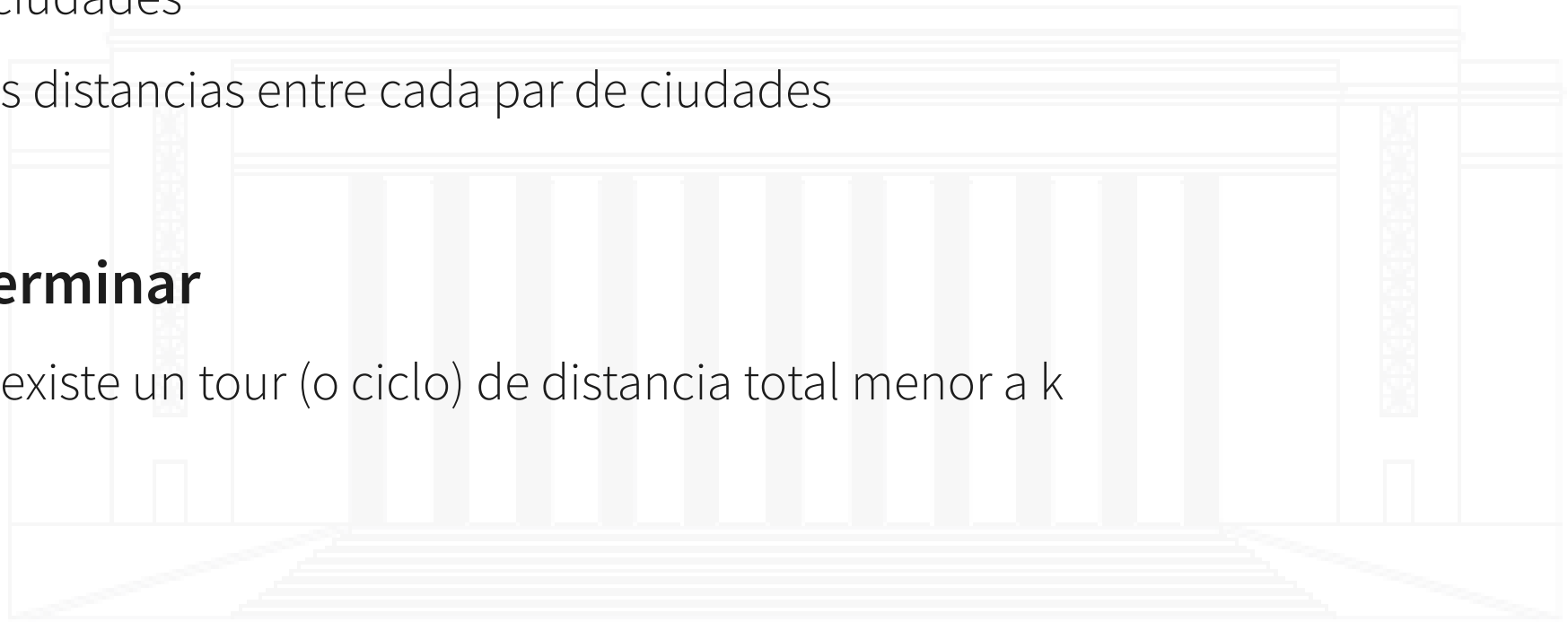
Dado

n ciudades

Las distancias entre cada par de ciudades

Determinar

Si existe un tour (o ciclo) de distancia total menor a k



Problema decisión del viajante es NP

Sea

n ciudades

Las distancias entre cada par de ciudades

T certificado = tour de ciudades

K distancia como límite

Se debe verificar

T contiene todas las ciudades (solo 1 vez) y termina y comienza en la misma

La suma de la distancia recorrida es menor a k

¿Viajante es “P”?

No se conoce algoritmo

Que resuelva Viajante en tiempo polinómico

Si probamos que

Viajante $\in$ NP-C

(Utilizaremos HAM-CYCLE)

Entonces

Viajante $\in$ P $\Leftrightarrow$ P = NP

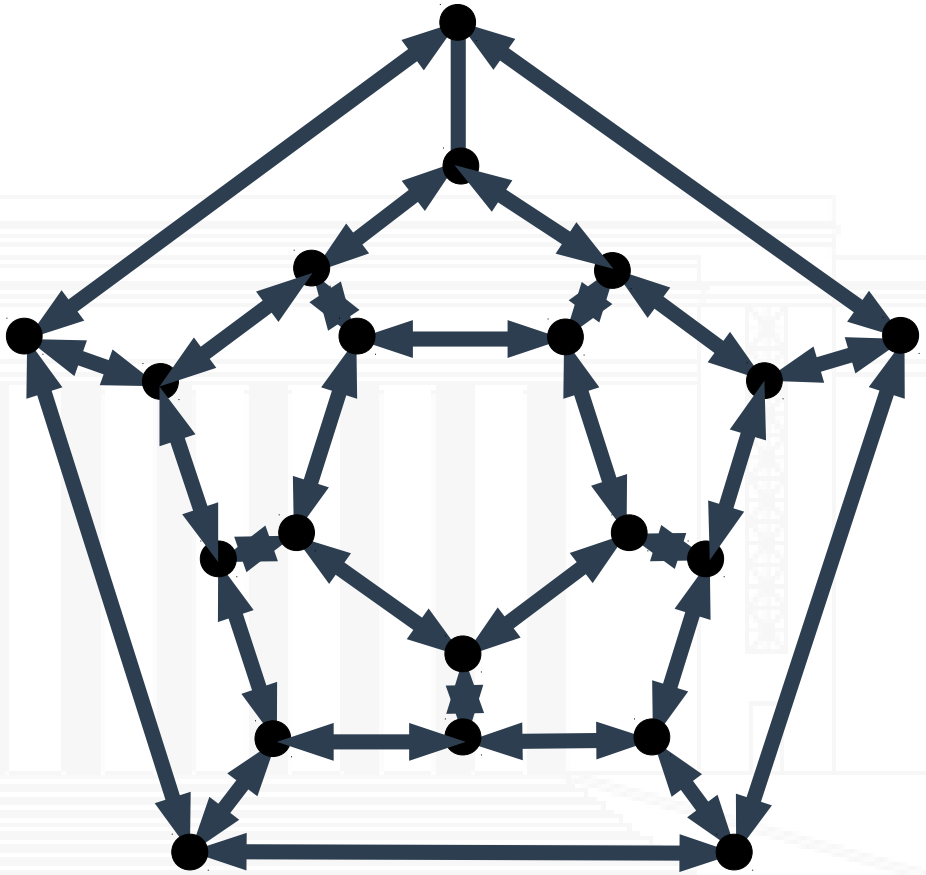
Problema de decisión de Ciclo Hamiltoneano

Sea

G = (V,E) grafo direccionado

Existe

Un ciclo hamiltoneano?



Reducción de HAM-CYCLE a Viajante

Sea una instancia I de HAM-CYCLE

G=(V,E)

Por cada

Vértice $v_i \in V \rightarrow$ creamos una ciudad v'_i

Arista $e_{i,j} \in E \rightarrow$ definiremos la distancia $d(v'_i,v'_j)=1$

Aquellas distancias que no están definidas (no tienen aristas) las crearemos con valor 2

Ponemos como valor $k = |V|$ (numero de vértices.

Solucionamos viajante con k definido

Si existe camino con longitud k, entonces existe ciclo hamiltoneano.

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

NP-C: Coloreo de grafos

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Función de coloreo

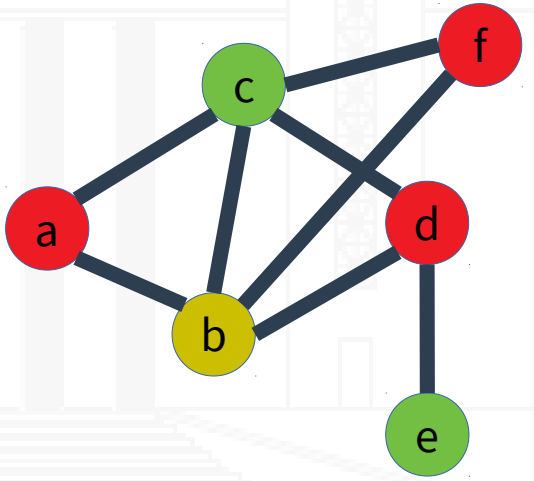
Sea

G=(V,E) grafo no dirigido

F: v → {1,2,…,k} función que asigna un color a cada vértice

Tal que

Para todo e=(u,v) ∈ E, f(u) ≠ f(v)



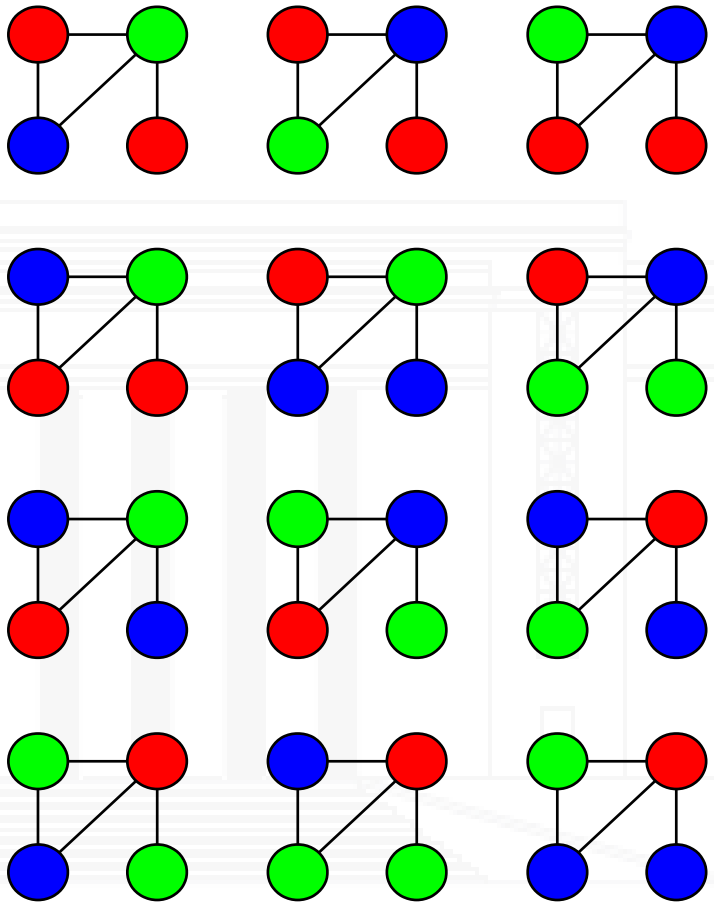
Número y polinomio cromático

Definimos al número cromático χ(G)

como el menor numero de colores necesario para colorear un grafo

Definimos polinomio cromático

Al la ecuación que permite contar el número de maneras en las cueles puede ser coloreado un grafo usando no mas de k colores

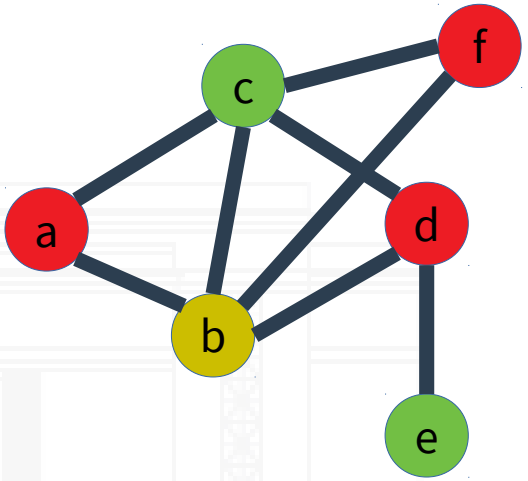


χ(G)=3 y puedo colorear de 12 maneras con 3 colores o menos

Clase de color y conjunto k-partito

Llamaremos clase de color

Al subconjunto de vértices coloreados con el mismo color



Una k-coloración

Equivale a una partición de nodos en k conjuntos independientes.
Si un grafo es k-coloreables entonces es k-partito

X={a,d,f}

Y={c,e}

Z={b}

Problema de decisión coloreo de grafos

Sea

G=(V,E) grafo no dirigido

K valor numérico.

¿Es posible

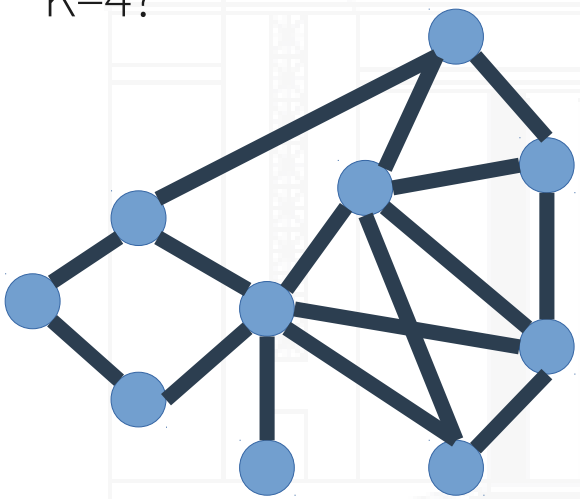
Definir una función de coloreo que utilice k o menos colores?

Ejemplo

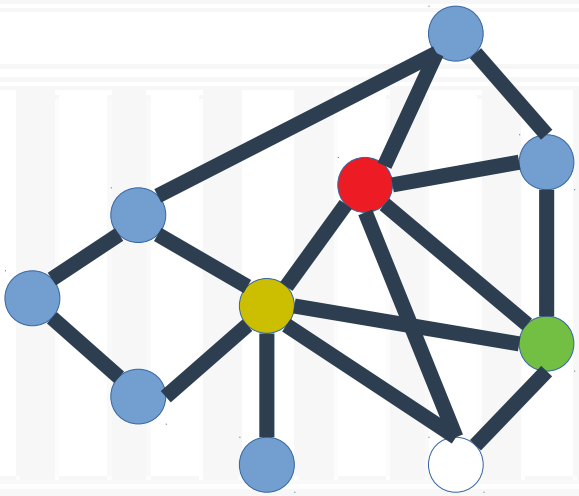
Se puede colorear G con menos de:

K=3?

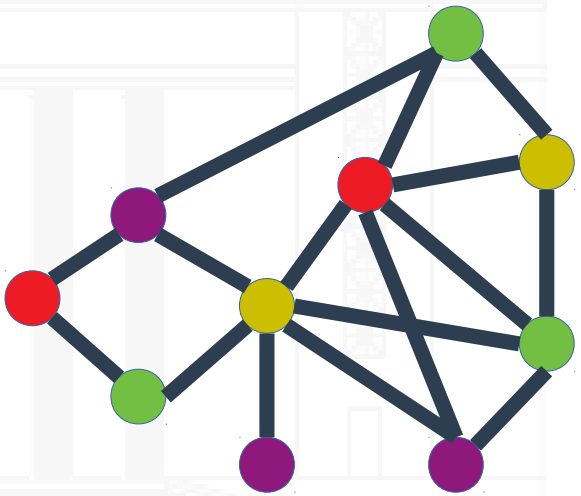
K=4?



G=(V,E)



No se puede colorear con solo 3 colores

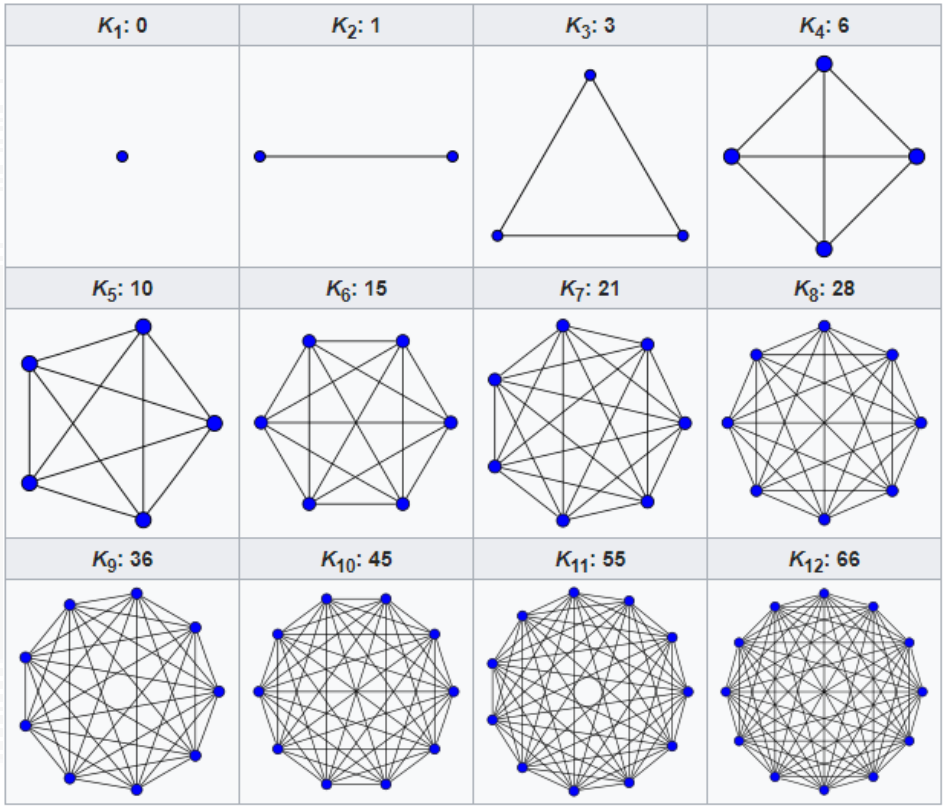


Posible coloración con 4 colores

Casos especiales: Grafo completo

Sea G=(V,E)

Grafo simple y completo (todos los vértices están comunicados entre si)



Entonces

Se requieren |V| colores para colorear el grafo

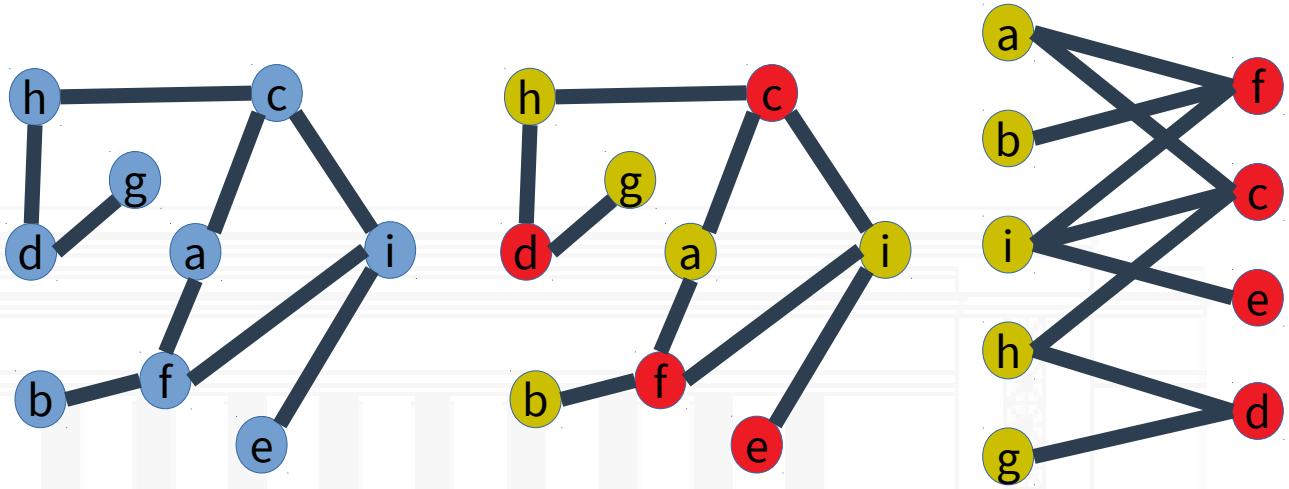
Podemos determinar rápidamente si G es completo

Si tiene |V|(|V|-1)/2 ejes

Casos especiales: Grafo bipartito

Sea G=(V,E)

Grafo bipartito



Entonces

Se requieren 2 colores para colorear el grafo

Mediante un algoritmo de coloreo

derivado de BFS podemos etiquetar cada nodo y determinar si el grafo es bipartito en tiempo polinomial

¿Coloreo de grados ∈ “NP”?

Dado

G=(V,E) grafo

K colores

T certificado: para todo vector que asigna a cada vértice un color

Puedo verificar (en tiempo polinomial)

Todos los nodos en V están en T

Las cantidad de colores usados en T son menores o iguales a k

No existen en G dos vértices adyacentes que en T tengan el mismo color

⇒ COLOREO DE GRAFOS ∈ NP

¿Coloreo de grados ∈ “NP-Hard”?

Probamos que

COLOREO-GRAFO ∈ NP-C

Utilizaremos 3SAT

3SAT $\leq_p$ COLOREO-GRAFO

3SAT $\leq_p$ COLOREO-GRAFO

Dada una

instancia I de 3SAT con k clausulas y n variables

Crearemos

Diferentes gadgets que seran subgrafos para colorear

1 gadget para las variables

1 gadget por clausula

3SAT $\leq_p$ COLOREO-GRAFO

Definimos

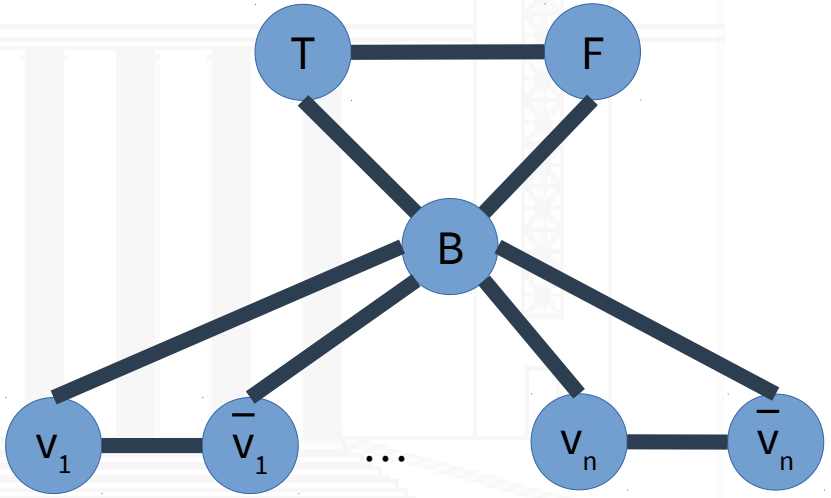
los nodos v_i y $\bar{v}_i$ correspondientes a cada variable x_i y su negada $\bar{x}_i$

los nodos especiales T (true), F (false) y B (base)

Generar los siguientes ejes

Cada v_i y $\bar{v}_i$ entre ellas y con B

T, F y B entre ellas



(la variable tendrá el valor true si tiene el mismo color que T)

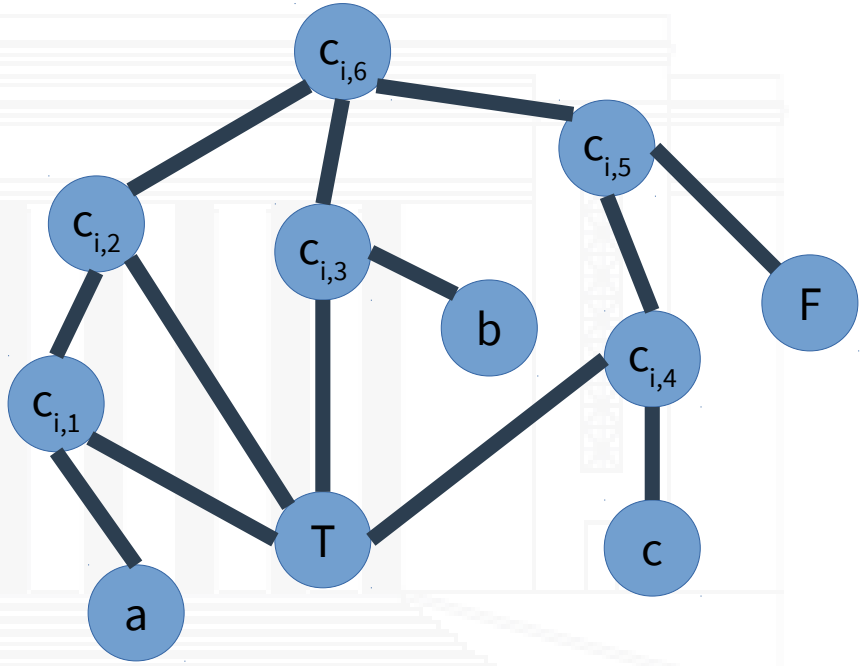
3SAT $\leq_p$ COLOREO-GRAFO

Crearemos 1 gadget para cada clausula

la clausula $c_i = (a,b,c)$ con $a,b,c \in \{x_1,\bar{x}_1,\ldots,x_n,\bar{x}_n\}$

las variables de la clausula corresponden a los nodos creados para las variables

(si $a=x_1$, el gadget en $ci,1$ se conecta a v_1)



3SAT $\leq_p$ COLOREO-GRAFO

Definimos

K =3 (cantidad de colores)

Si

El grafo resultante se puede pintar con 3 colores, entonces la expresion I se puede satisfacer

Los nodos correspondientes a las variables con igual color a T, deben estar en true.

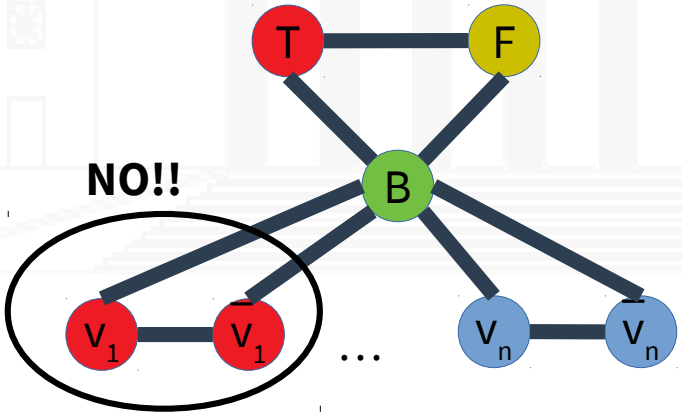
3SAT $\leq_p$ COLOREO-GRAFO

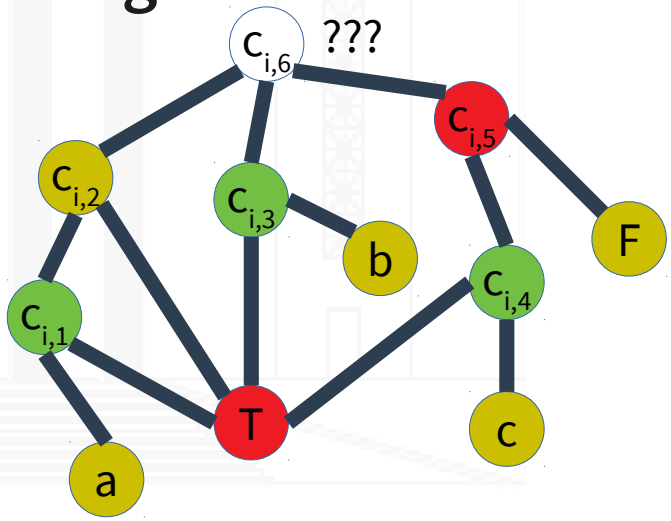
Con al menos 1 variable en True de la clausula i

el subgrafo correspondiente al gadget de la clausula i se puede colorear con 3 colores

Se se requiere que 1 variable este negada y sin negar

No se puede colorear el grafo

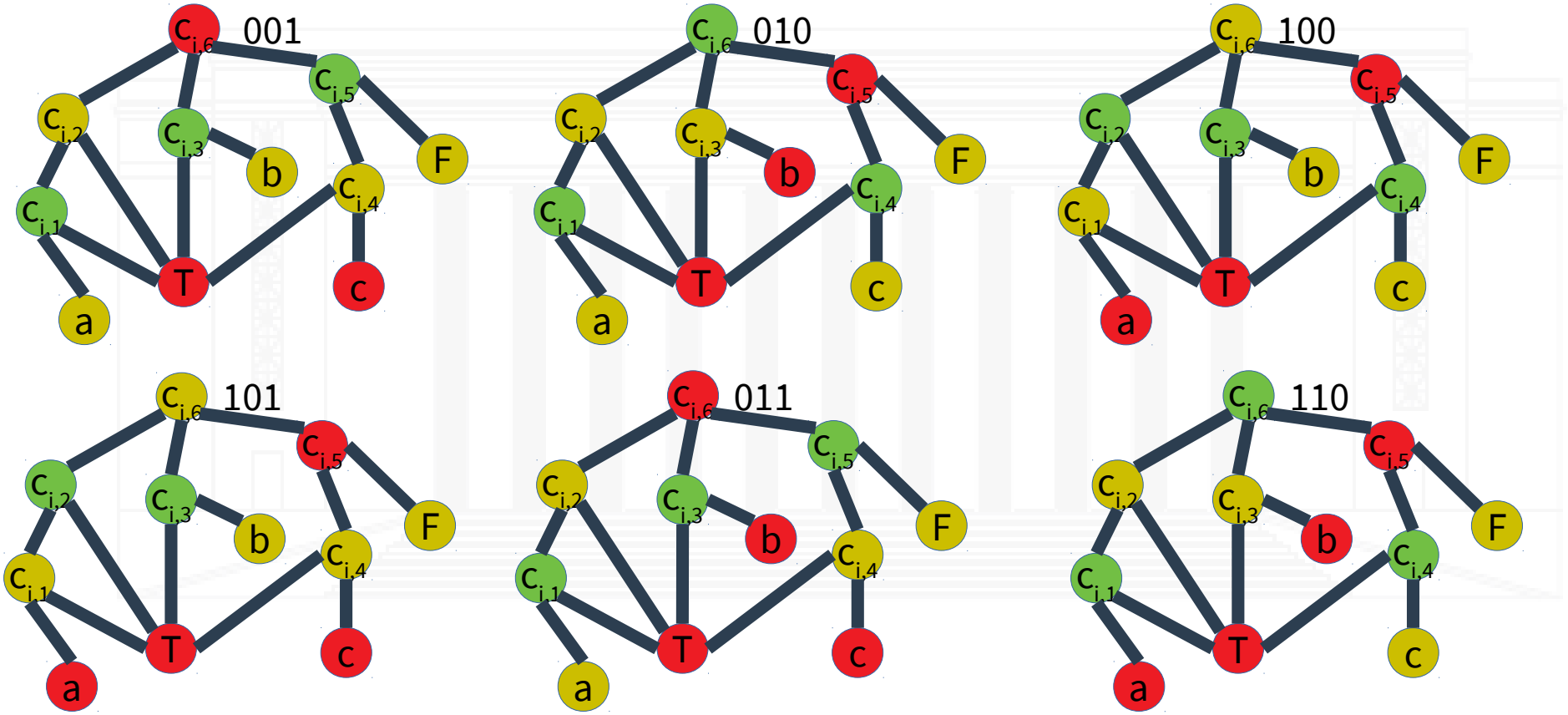




Con a,b,c en “false” no se puede colorear el gadget

3SAT $\leq_p$ COLOREO-GRAFO

Ejemplos de activación de la clausula

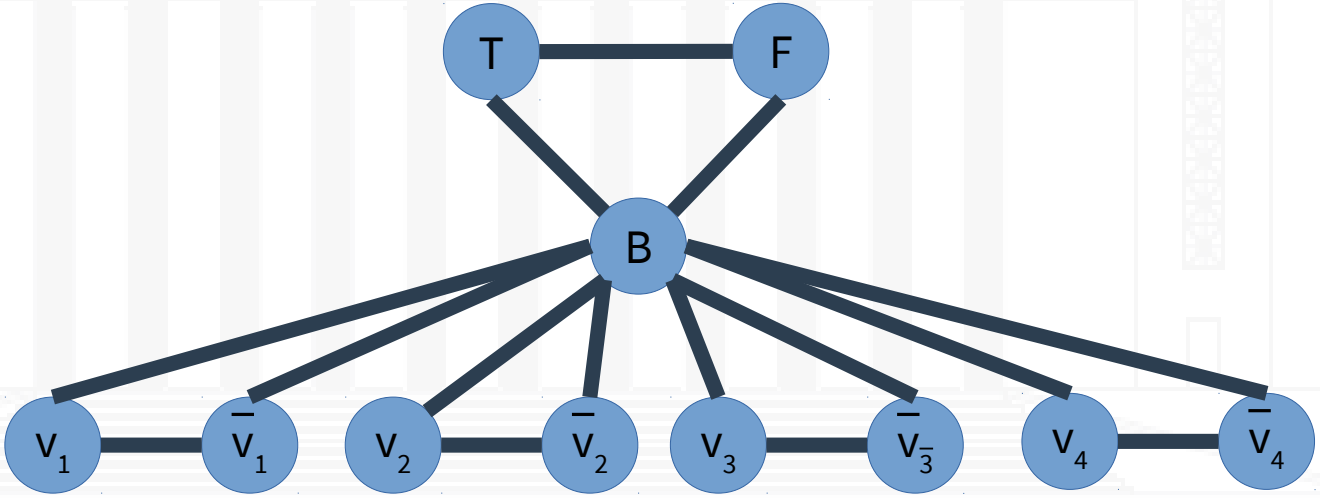


Ejemplo

Si tengo la expresión

$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4)$

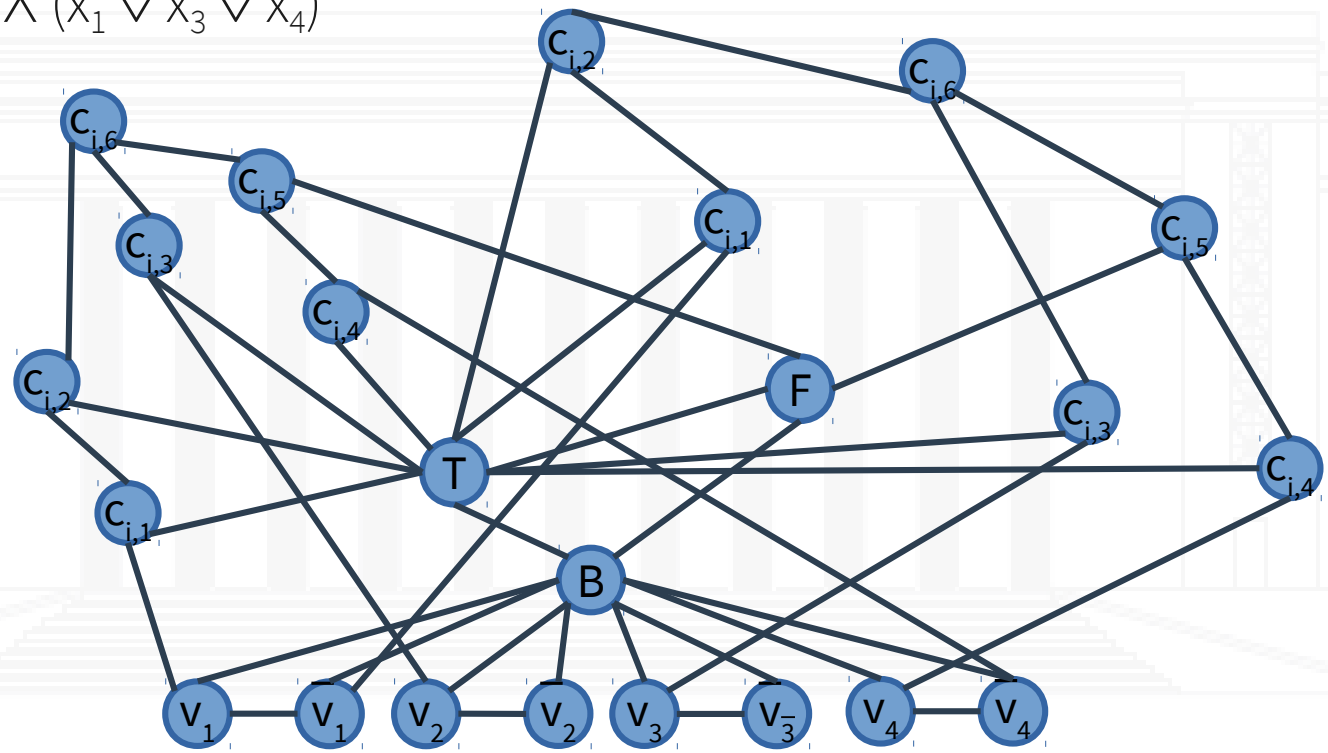
Con 4 variables y 2 clausulas



Ejemplo

Si tengo la expresión

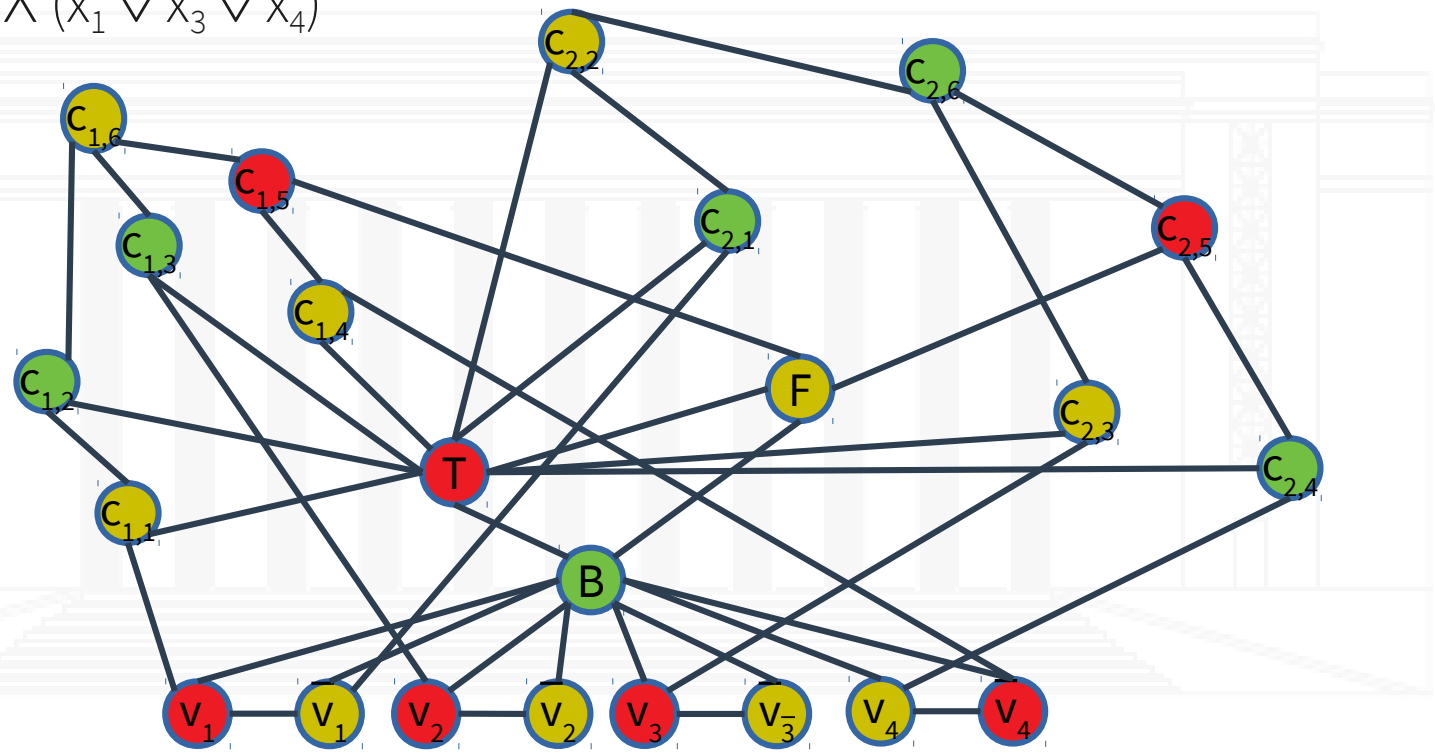
$$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4)$$



Ejemplo

Si tengo la expresión

$$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4)$$



$x_1=1, x_2=1, x_3=1, x_4=0$

COLOREO-GRAFO ∈ "NP-C"

Como

COLOREO-GRAFOS ∈ NP

Y 3SAT $\leq_p$ COLOREO-GRAFOS

Entonces

COLOREO-GRAFOS ∈ NP-C

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

NP-C: Subset Sum

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Subset Sum

Sea

Conjunto de $C=\{w_1,w_2,\ldots,w_n\}$ números naturales

Determinar

Si es existe un subconjunto de C que sume exactamente W

Es un problema de decisión

Relacionado con el problema de la mochila

¿Subset Sum ∈ “P”?

Utilizando programación dinámica

Se puede resolver en tiempo pseudo-polinomial O(Wn)

Si representamos W en bits

El algoritmos crece exponencialmente a la cantidad de bits de W

¿Existe una solución polinomial?

No se conoce!

¿Subset Sum ∈ "NP"?

Dado

C conjunto de n números naturales

W numero a sumar

T certificado con subconjunto de C

Podemos determinar polinomialmente

$$\sum_{t_i \in T} t_i = W$$

Todo $t_i \in C$

⇒ SUBSET SUM ∈ NP

¿Subset Sum ∈ “NP-Hard”?

Probaremos que

SUBSET-SUM ∈NP-C

Utilizaremos 3DM

3DM $\leq_p$ SUBSET-SUM

3 Dimensional Matching

Dados

3 sets disjuntos X.Y,Z de tamaño n cada uno.

Un set $C \subseteq X,Y,Z$ de triplas ordenadas

Determinar

Si existe un subset de n triplas en C tal que cada elemento de $X \cup Y \cup Z$ sea contenido exactamente en una de esa triplas?

Reducción de 3DM a SUBSET SUM

Sea

I instancia de 3DM

Con

X,Y,Z conjuntos de n elementos

$T \subseteq X \times Y \times Z$ conjuntos de m triplas

Podemos

Representar cada tripla como un vector de bits

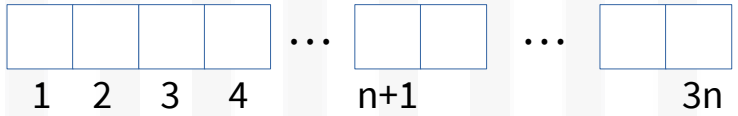
Representación vectorial

Utilizaremos vectores de 3*n bits

Los primeros n bits representan los elementos del conjunto X

Los siguientes n bits representan los elementos del conjunto Y

Los últimos n bits representan los elementos del conjunto Z



A cada elemento de los conjuntos X (y similarmente para Y,Z)

Les asignaremos un orden arbitrario de 1 a n

Llamaremos $pos_x(x)$, $pos_y(y)$, $pos_z(z)$ a las funciones que dado el elemento nos retorna su posición en el set

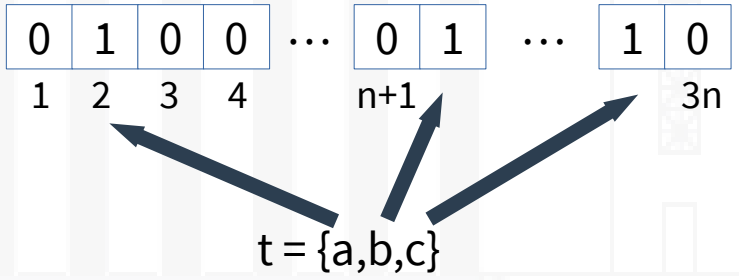
Representación vectorial

A cada t tripla de T

$t = \{x_i, y_j, z_k\}$

La representaremos como un vector V_t de 3n bits

Con todos los bits en cero



Pondremos los siguientes bits en 1

$pos_x(x_i)$

$pos_y(y_j) + n$

$pos_z(z_k) + 2n$

Transformación en un numero entero

Cada vector v_t

Se puede interpretar como un numero w_t en el subconjunto de subset Sum

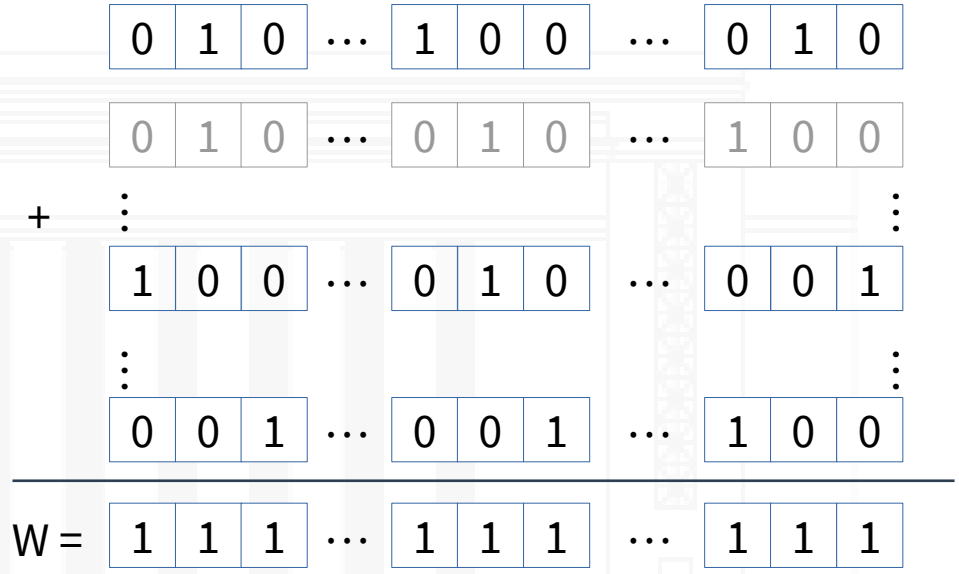
El Valor W lo formaremos como

el vector con todos los bits en 1

Para sumar W

Tenemos que encontrar aquellos wt que sumados den W

… pero esta propuesta tiene un problema



Problema de Overflow

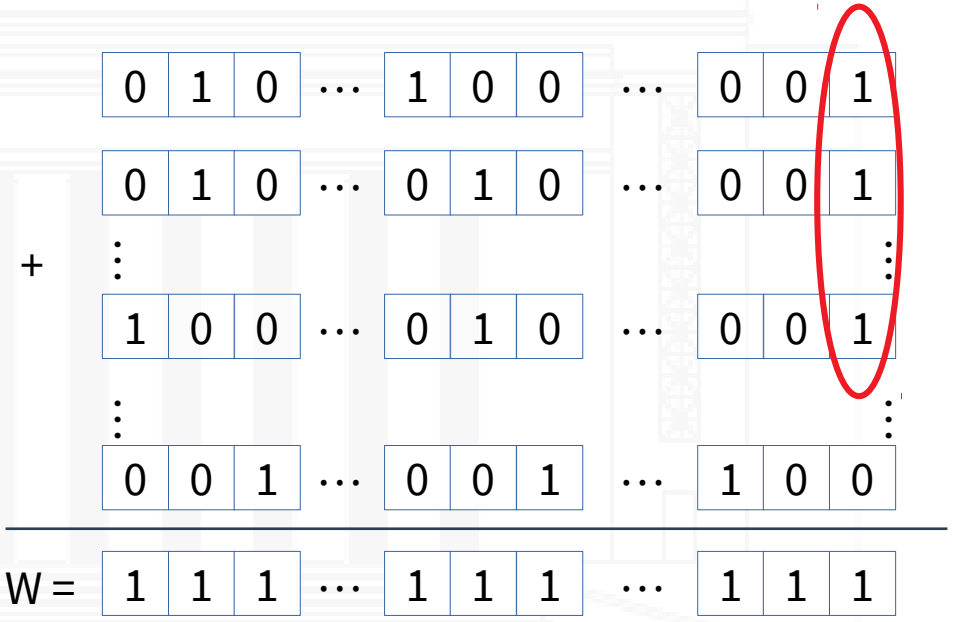
Puedo elegir 2 o mas números con el mismo bit prendido

Eso genera un overflow

Y podríamos llegar erroneamente a W

Para evitarlo

Podemos redefinir como expresar wi en base al vector



Transformación en un numero entero

Seleccionaremos una base d

Y representaremos cada tripla como un número de la manera:

$$w_t = \sum_{i=1}^{3n} v_t[i] * d^{i-1}$$

Como todos los elementos del vectores estarán en 0 excepto 3 se pude ver como:

$$w_t = d^{pos_x(x_i)-1} + d^{pos_x(y_i)+n-1} + d^{pos_x(z_i)+2n-1}$$

Para la base d

Utilizaremos un valor mayor a m (cantidad de triplas) → d = m+1

(con eso, aunque las m triplas contengan un mismo elemento no será posible el overflow)

Ejemplo

Sea la instancia de 3DM

$X = \{x_1, x_2, x_3\}$

$Y = \{y_1, y_2, y_3\}$

$Z = \{z_1, z_2, z_3\}$

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

Tenemos

$n=3$

$m=7$

Ejemplo (cont.)

Definimos

- Base a utilizar: d=m+1=8
- Tamaño del vector: 3*n=9

Calculamos

- El vector de cada tripla vt
- El vector W

Representamos

- Cada vector como un numero en base d

t	v_t		w_t
(x_1,y_1,z_1)	001001001	262144+512+1	262657
(x_2,y_2,z_3)	100010010	16777216+4096+8	16781320
(x_3,y_3,z_1)	001100100	262144+32768+64	294976
(x_1,y_1,z_2)	010001001	2097152+512+1	2097665
(x_3,y_3,z_2)	010100100	2097152+32768+64	2129984
(x_1,y_3,z_2)	010100001	2097152+32768+1	2129921
(x_3,y_3,z_3)	100100100	16777216+32768+64	16810048
W	111111111		19173961

Ejemplo (cont.)

La instancia de Subset Sum

C = {262657, 16781320, 294976, 2097665, 2129984, 2129921, 16810048}

W=19173961

Resolvemos

(en este caso por fuerza bruta... es NP-C)

Y verificamos que triplas conforman el 3DM

$$
\begin{array}{rll}
 & 262657 & \rightarrow (x_1,y_1,z_1) \\
+ & 16781320 & \rightarrow (x_2,y_2,z_3) \\
 & 2129984 & \rightarrow (x_3,y_3,z_2) \\
\hline
 & 19173961 & =W
\end{array}
$$

SUBSET SUM es NP-C

Como

SUBSET SUM $\in$ NP

Y 3DM $\leq_p$ SUBSET SUM

Entonces

SUBSET SUM $\in$ NP-C

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020

.UBAfiuba
FACULTAD DE INGENIERÍA

NP-C: Clique

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Clique

Sea

G=(V,E) grafo no dirigido

Llamaremos

clique a un subconjunto V’ $\subseteq$ V de vértices

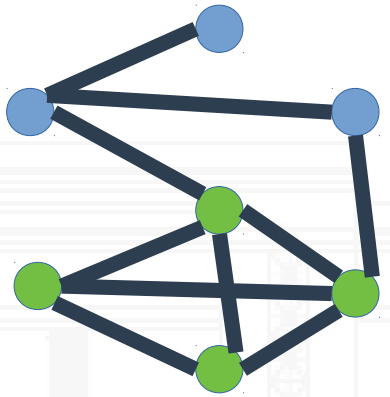
tal que

para todo v,u $\in$ V’, el eje (u,v) $\in$ E

(Es decir que todos los vértices están conectados entre si)

La cantidad de nodos en V’

Determina el tamaño del clique



Clique de 4 nodos,
(clique de tamaño 4)

Problema de decisión de cliques

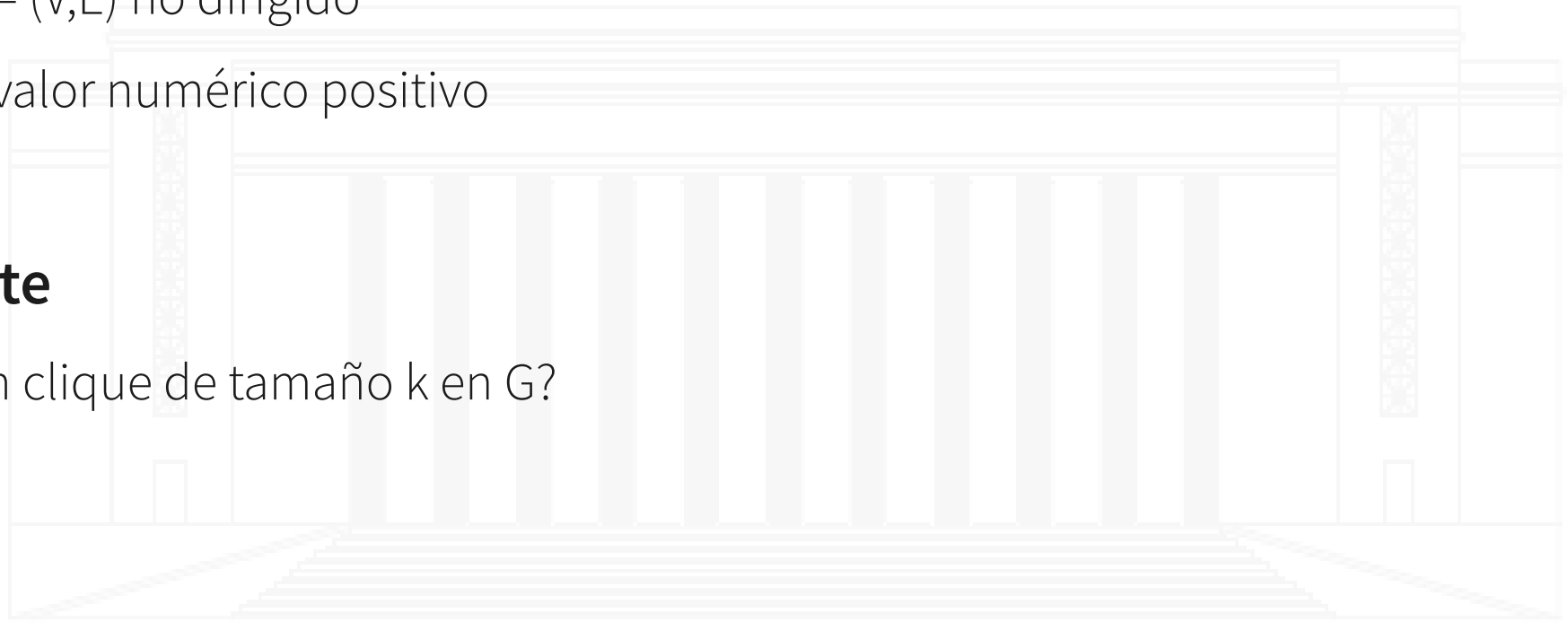
Dado

G = (V,E) no dirigido

K valor numérico positivo

Existe

Un clique de tamaño k en G?



¿Cliques ∈ “NP”?

Dado

G=(V,E) grafo

K tamaño del clique

T certificado: subconjunto de nodos de V

Puedo verificar (en tiempo polinomial)

La cantidad de nodos en T es igual a K

Cada nodo en T está conectado a los otros nodos de T

⇒ CLIQUES ∈ NP

¿Cliques ∈ “P”?

Por fuerza bruta

Puedo probar todos las combinaciones de nodos con k nodos.

$$\binom{k}{|V|} = \frac{|V|!}{k! \cdot (|V| - k)!}$$

Y por cada posibilidad probar si están conectados entre si los k nodos en k^2 operaciones.

Si busco cliques pequeños con |V| grandes

La complejidad total “parece” polinomial

Si busco cliques grandes

la complejidad total es exponencial

¿Cliques ∈ “NP-Hard”?

Probaremos que

CLIQUES ∈NP-C

Utilizaremos 3SAT

$3SAT \leq_p CLIQUES$

3SAT $\leq_p$ CLIQUES

Dada una

instancia I de 3SAT con k clausulas y n variables

Crearemos

Un nodo por cada variable en una clausula

Por cada par de variables de diferentes cláusulas

Crearemos un eje entre ellas si no corresponden a la misma variable negada

Buscaremos un clique de tamaño k

Ejemplo

$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_4)$

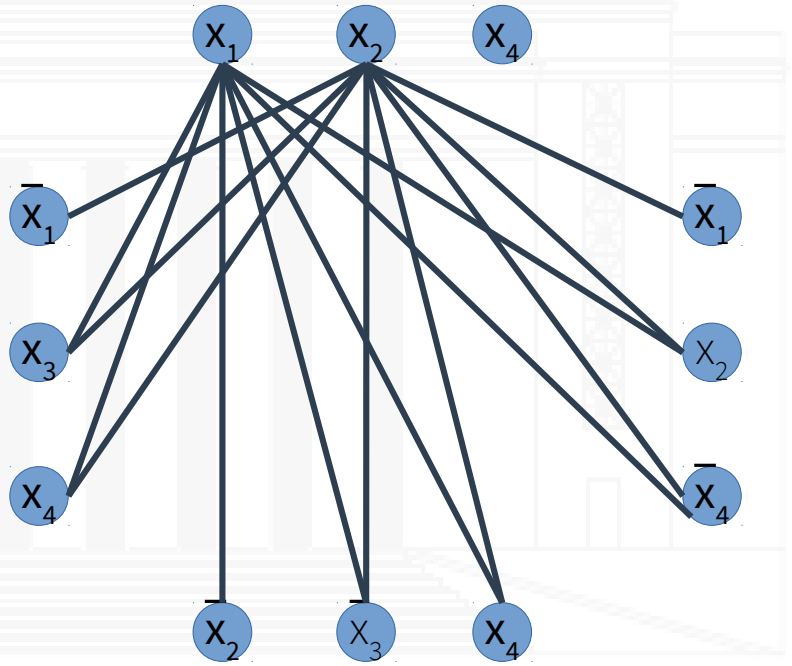
Con 4 variables y 4 clausulas

Armo los nodos

para cada variable de cada clausula

Agrego los ejes

según condición de construcción



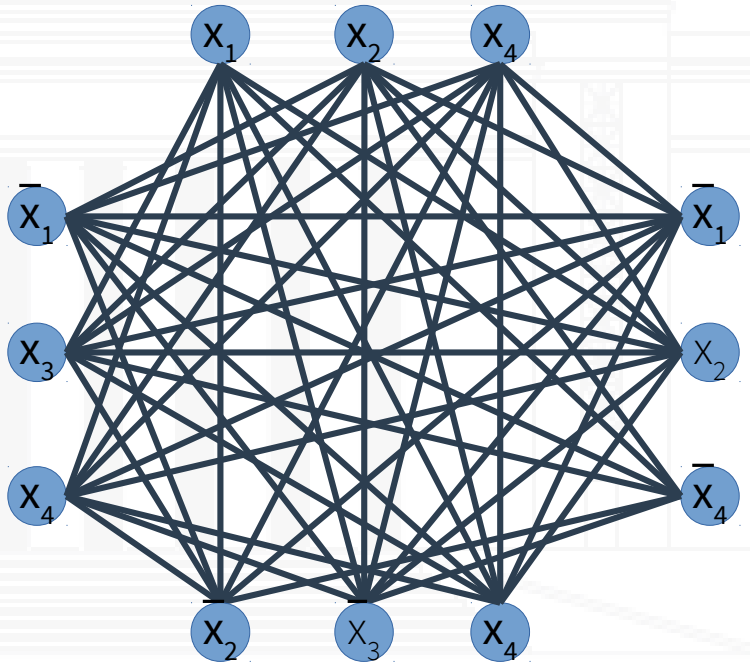
Ejemplo (cont.)

$$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_4)$$

Busco clique

De tamaño k=4

(para activar las 4 clausulas)



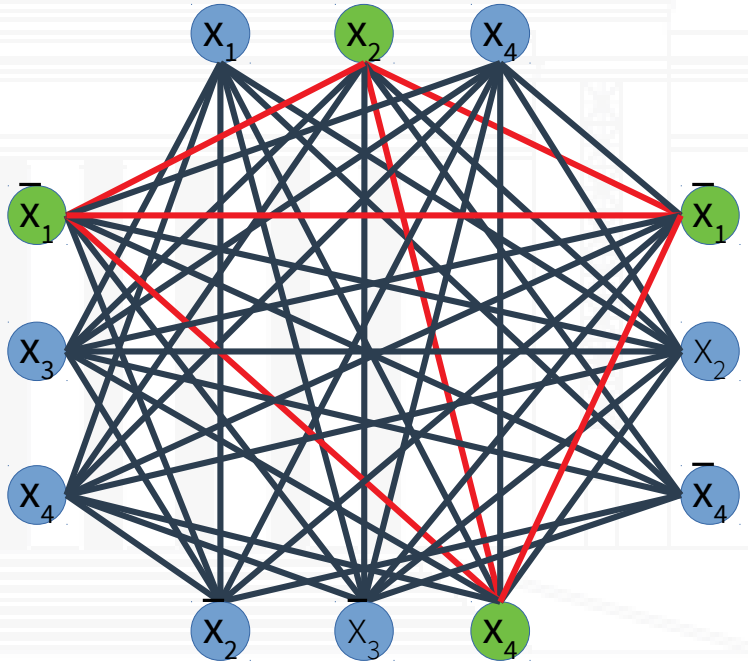
Ejemplo (cont.)

$$E = (x_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_4)$$

Busco clique

De tamaño k=4

(para activar las 4 clausulas)



Los nodos dentro del clique

Indican el valor de las variables

(las variables que no están en el clique se pueden poner en true o en false)

CLIQUE ∈ “NP-C”

Como

CLIQUE ∈ NP

Y 3SAT $\leq_p$ CLIQUE

Entonces

CLIQUE ∈ NP-C

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Junio de 2020