.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

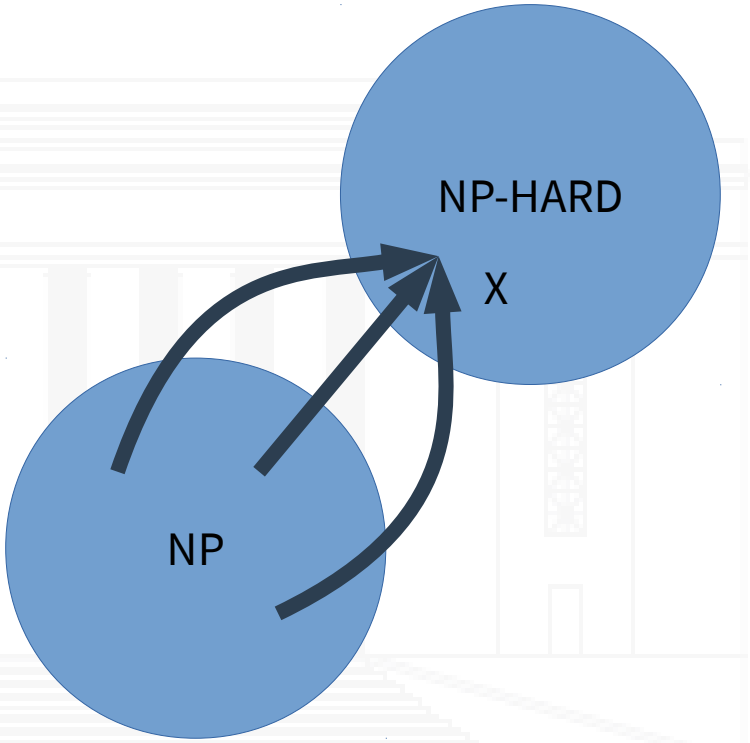
Tal que

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)

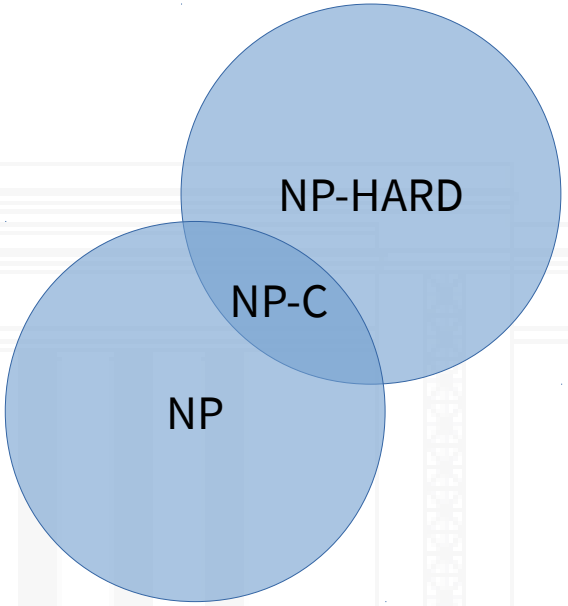
Sea

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)

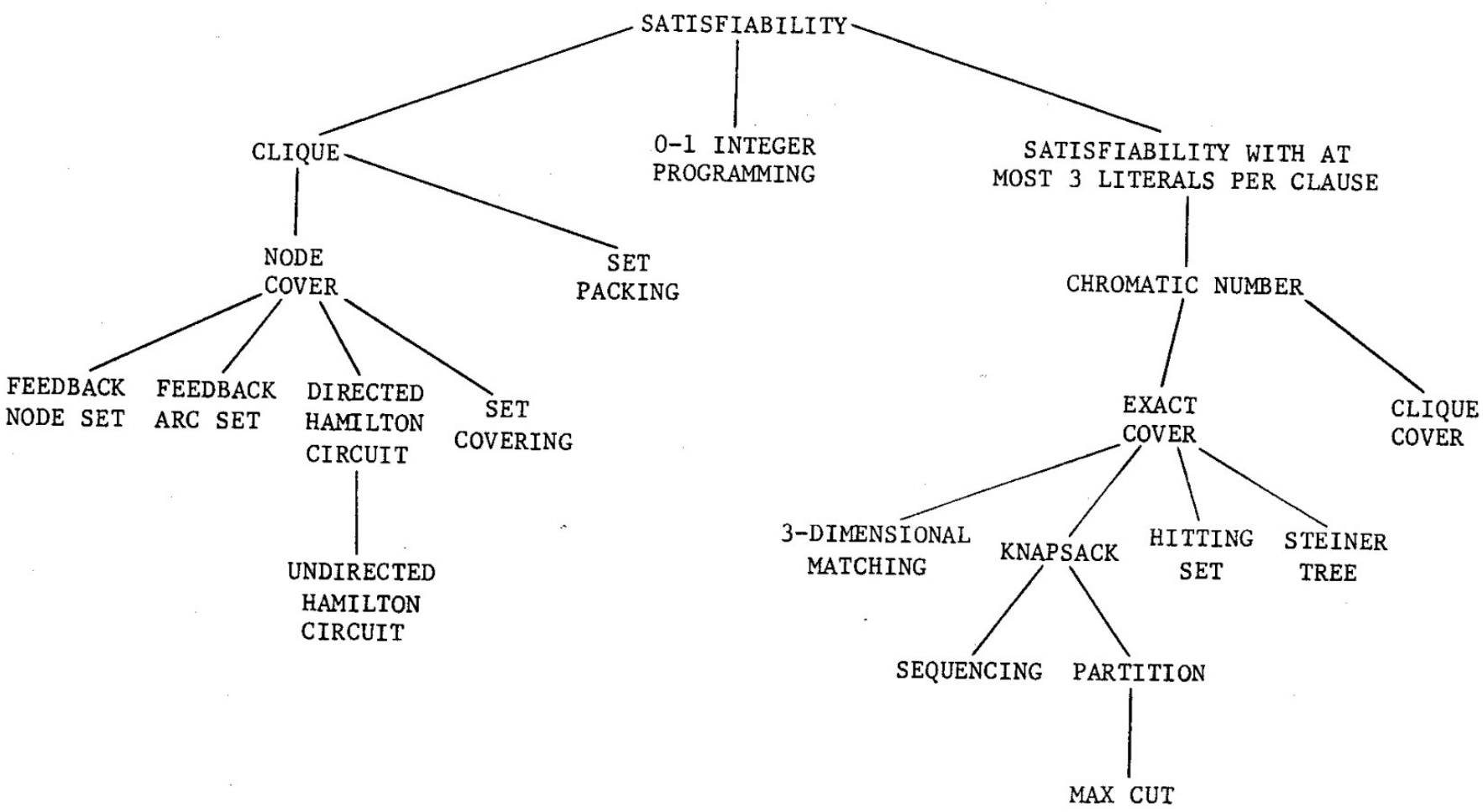
Toma un problema demostrado como NP-C y lo reduce a X

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

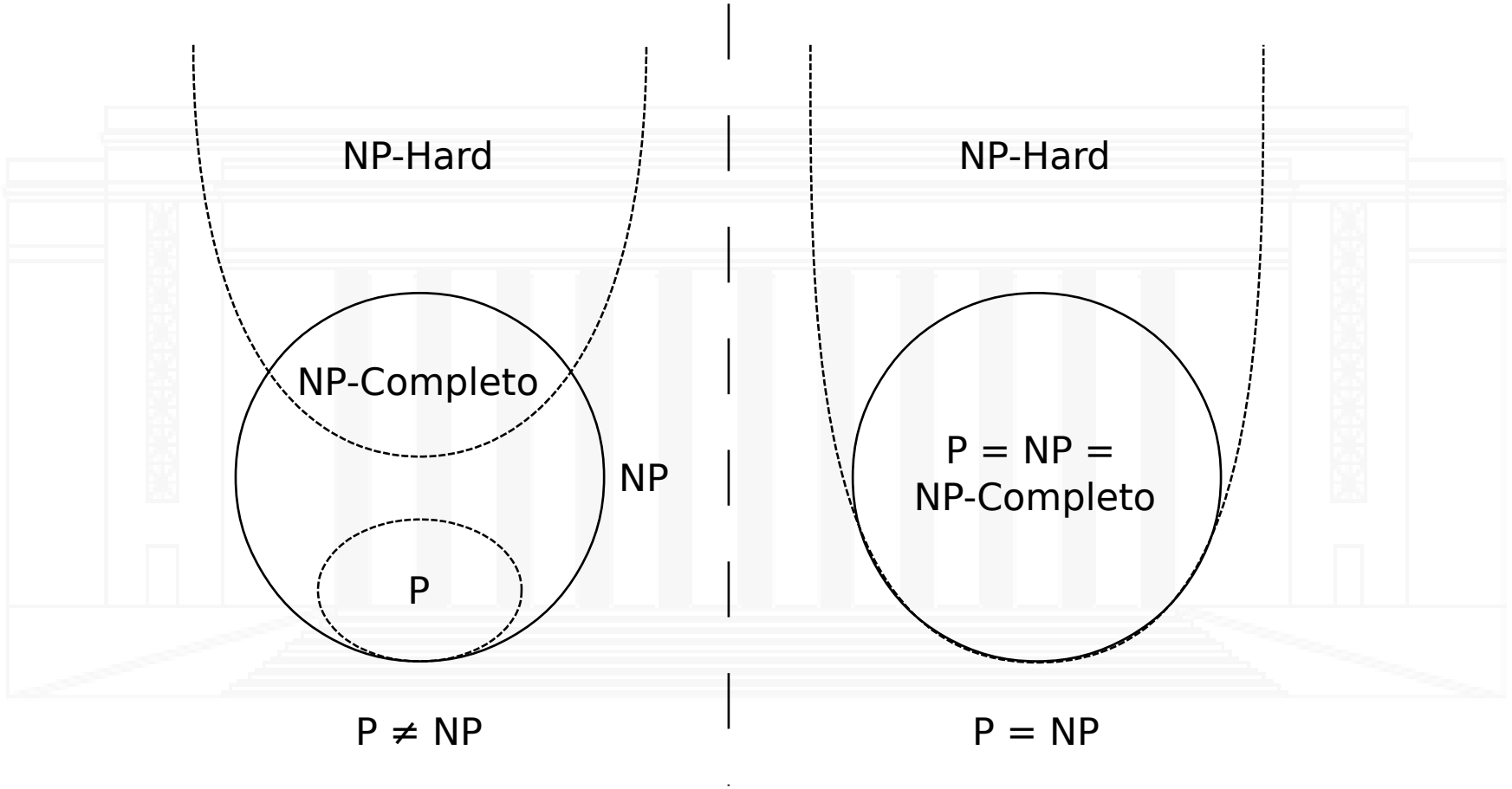
Definir el certificador eficiente

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

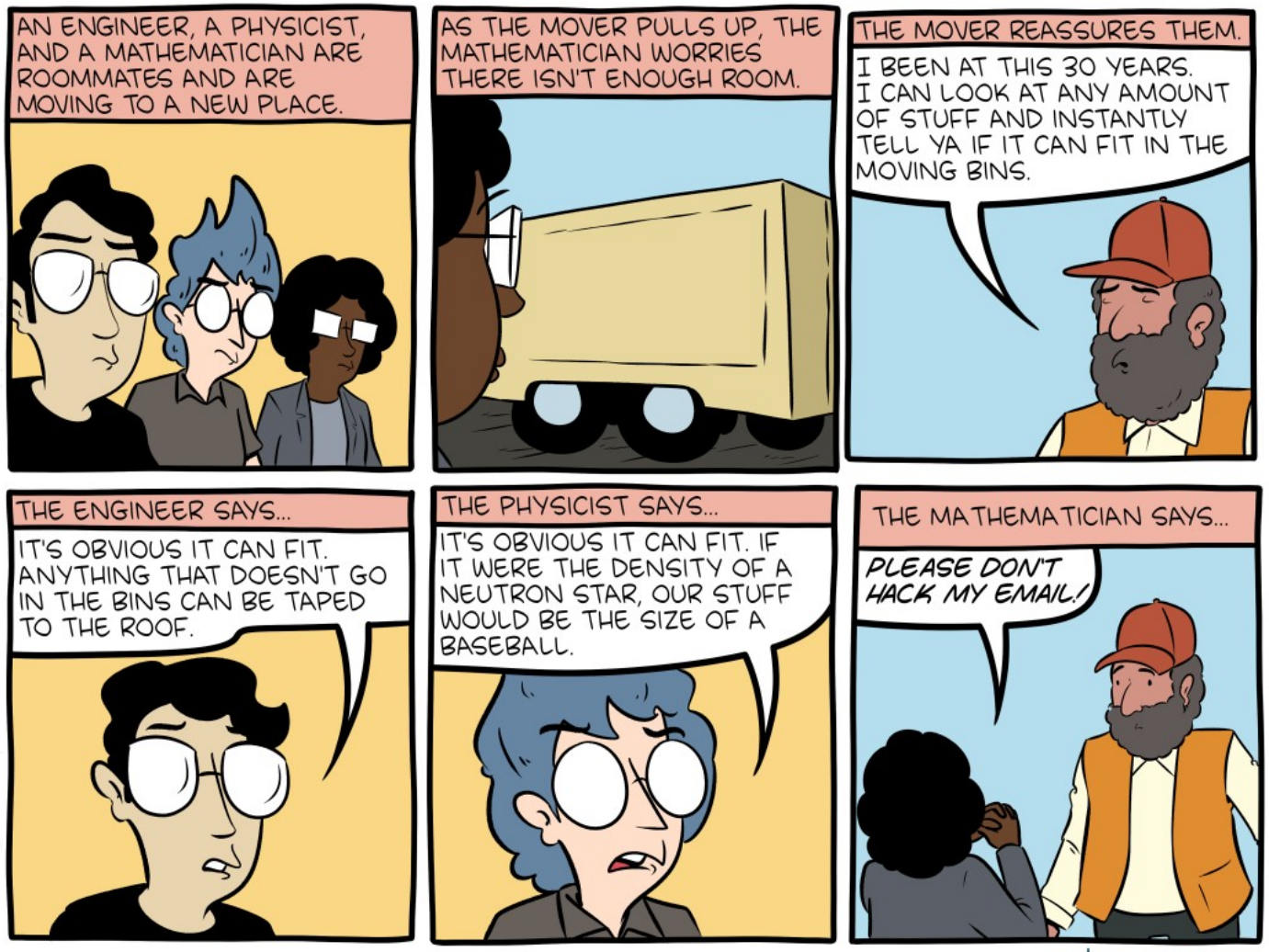
Todo problema NP-C se puede reducir entre si con igual complejidad

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