Reducciones vía gadgets



UNIVERSIDAD DE CHILE

Arturo Merino

Plan

Recuerdo y plan

- **Recuerdo:**

Recuerdo y plan

- **Recuerdo:**
  - Cook–Levin: $\mathrm{SAT\text{-}CNF}$ es NP-completo.

Recuerdo y plan

- **Recuerdo:**
  - Cook–Levin: $\mathrm{SAT\text{-}CNF}$ es NP-completo.
  - Ya vimos $\mathrm{3SAT}$, IS, CLIQUE y VC.

Recuerdo y plan

- **Recuerdo:**
  - Cook–Levin: $\mathrm{SAT\text{-}CNF}$ es NP-completo.
  - Ya vimos $\mathrm{3SAT}$, IS, CLIQUE y VC.
  - B es NP-completo si $B \in \mathsf{NP}$ y $A \leq_{\mathsf{P}} B$ para algún A NP-completo.

Recuerdo y plan

- **Recuerdo:**
  - Cook–Levin: $\mathrm{SAT\text{-}CNF}$ es NP-completo.
  - Ya vimos $\mathrm{3SAT}$, IS, CLIQUE y VC.
  - B es NP-completo si $B \in \mathsf{NP}$ y $A \leq_{\mathsf{P}} B$ para algún A NP-completo.

- **Plan:**

Recuerdo y plan

- **Recuerdo:**
  - Cook–Levin: $\mathrm{SAT\text{-}CNF}$ es NP-completo.
  - Ya vimos $\mathrm{3SAT}$, IS, CLIQUE y VC.
  - B es NP-completo si $B \in \mathsf{NP}$ y $A \leq_{\mathsf{P}} B$ para algún A NP-completo.

- **Plan:**
  - Más reducciones de NP-completitud

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.
- ¿Features de 3-SAT?

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.
- ¿Features de 3-SAT?
  - Variables $\leftarrow$ gadgets de variables.

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.
- ¿Features de 3-SAT?
  - Variables $\leftarrow$ gadgets de variables.
  - Claus. $\leftarrow$ gadgets de claus.

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.
- ¿Features de 3-SAT?
  - Variables $\leftarrow$ gadgets de variables.
  - Claus. $\leftarrow$ gadgets de claus.
- ¿Cómo diseñar gadgets?

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.
- ¿Features de 3-SAT?
  - Variables $\leftarrow$ gadgets de variables.
  - Claus. $\leftarrow$ gadgets de claus.
- ¿Cómo diseñar gadgets?
  - Variables $\leftarrow$ decisiones binarias

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.
- ¿Features de 3-SAT?
  - Variables $\leftarrow$ gadgets de variables.
  - Claus. $\leftarrow$ gadgets de claus.
- ¿Cómo diseñar gadgets?
  - Variables $\leftarrow$ decisiones binarias
  - Claus. $\leftarrow$ satisfechas si alguna de los literales lo es

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.
- ¿Features de 3-SAT?
  - Variables $\leftarrow$ gadgets de variables.
  - Claus. $\leftarrow$ gadgets de claus.
- ¿Cómo diseñar gadgets?
  - Variables $\leftarrow$ decisiones binarias
  - Claus. $\leftarrow$ satisfechas si alguna de los literales lo es
  - Aceptar $\leftarrow$ todas las clausulas son satisfechas

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.
- ¿Features de 3-SAT?
  - Variables $\leftarrow$ gadgets de variables.
  - Claus. $\leftarrow$ gadgets de claus.
- ¿Cómo diseñar gadgets?
  - Variables $\leftarrow$ decisiones binarias
  - Claus. $\leftarrow$ satisfechas si alguna de los literales lo es
  - Aceptar $\leftarrow$ todas las clausulas son satisfechas
- **Hoy:** Reducciones desde 3SAT.

Reducciones vía gadgets

- Reducciones más complejas: *gadgets*.
- Un *gadget* es una pieza de la reducción que modela un feature del problema.
- ¿Features de 3-SAT?
  - Variables $\leftarrow$ gadgets de variables.
  - Claus. $\leftarrow$ gadgets de claus.
- ¿Cómo diseñar gadgets?
  - Variables $\leftarrow$ decisiones binarias
  - Claus. $\leftarrow$ satisfechas si alguna de los literales lo es
  - Aceptar $\leftarrow$ todas las clausulas son satisfechas
- **Hoy:** Reducciones desde 3SAT.
  - Gadgets es una técnica general.

Problemas NP-completos

Hamiltonian Cycle

- Planificar ruta de reparto

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Teo.

HAMCYCLE es NP-completo.

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Teo.

HAMCYCLE es NP-completo.

- ¿En NP?

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Teo.

HAMCYCLE es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

Hamiltonian Cycle

- Planificar ruta de reparto

Def.

Input: Un grafo dirigido G.

Output: ¿Hay un ciclo que visite cada vértice exactamente una vez?

$$\text{HAMCYCLE} = \{\langle G \rangle : G \text{ tiene un ciclo hamiltoniano}\}.$$

Teo.

HAMCYCLE es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De 3SAT a HAMCYCLE.

3-Coloring

- Modelar incompatibilidades

3-Coloring

- Modelar incompatibilidades

Def.

Input: Un grafo no dirigido G.

3-Coloring

- Modelar incompatibilidades

Def.

Input: Un grafo no dirigido G.

Output: ¿Se pueden colorear los vértices con a lo más tres colores, sin vecinos del mismo color?

3-Coloring

- Modelar incompatibilidades

Def.

Input: Un grafo no dirigido G.

Output: ¿Se pueden colorear los vértices con a lo más tres colores, sin vecinos del mismo color?

$$3\text{COLOR} = \{\langle G \rangle : G \text{ admite una 3-coloración}\}.$$

3-Coloring

- Modelar incompatibilidades

Def.

Input: Un grafo no dirigido G.

Output: ¿Se pueden colorear los vértices con a lo más tres colores, sin vecinos del mismo color?

$$3\text{COLOR} = \{\langle G \rangle : G \text{ admite una 3-coloración}\}.$$

Teo.

3COLOR es NP-completo.

3-Coloring

- Modelar incompatibilidades

Def.

Input: Un grafo no dirigido G.

Output: ¿Se pueden colorear los vértices con a lo más tres colores, sin vecinos del mismo color?

$$3\text{COLOR} = \{\langle G \rangle : G \text{ admite una 3-coloración}\}.$$

Teo.

3COLOR es NP-completo.

- ¿En NP?

3-Coloring

- Modelar incompatibilidades

Def.

Input: Un grafo no dirigido G.

Output: ¿Se pueden colorear los vértices con a lo más tres colores, sin vecinos del mismo color?

$$3\text{COLOR} = \{\langle G \rangle : G \text{ admite una 3-coloración}\}.$$

Teo.

3COLOR es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

3-Coloring

- Modelar incompatibilidades

Def.

Input: Un grafo no dirigido G.

Output: ¿Se pueden colorear los vértices con a lo más tres colores, sin vecinos del mismo color?

$$3\text{COLOR} = \{\langle G \rangle : G \text{ admite una 3-coloración}\}.$$

Teo.

3COLOR es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De 3SAT a 3COLOR.

3D-Matching

- Formar ternas que cubran todo sin repetir elementos.

3D-Matching

- Formar ternas que cubran todo sin repetir elementos.

Def.

Input: Conjuntos disjuntos X, Y, Z de tamaño n, y $T \subseteq X \times Y \times Z$.

3D-Matching

- Formar ternas que cubran todo sin repetir elementos.

Def.

Input: Conjuntos disjuntos X, Y, Z de tamaño n, y $T \subseteq X \times Y \times Z$.

Output: ¿Hay n ternas de T que cubran cada elemento exactamente una vez? Un conjunto así es un *3d-matching perfecto*.

3D-Matching

- Formar ternas que cubran todo sin repetir elementos.

Def.

Input: Conjuntos disjuntos X, Y, Z de tamaño n, y $T \subseteq X \times Y \times Z$.

Output: ¿Hay n ternas de T que cubran cada elemento exactamente una vez? Un conjunto así es un *3d-matching perfecto*.

$$\text{3DM} = \{\langle X, Y, Z, T\rangle : T \text{ admite un 3d-matching perfecto}\}.$$

3D-Matching

- Formar ternas que cubran todo sin repetir elementos.

Def.

Input: Conjuntos disjuntos X, Y, Z de tamaño n, y $T \subseteq X \times Y \times Z$.

Output: ¿Hay n ternas de T que cubran cada elemento exactamente una vez? Un conjunto así es un *3d-matching perfecto*.

$$\mathrm{3DM} = \{\langle X, Y, Z, T\rangle : T \text{ admite un 3d-matching perfecto}\}.$$

Teo.

$\mathrm{3DM}$ es NP-completo.

3D-Matching

- Formar ternas que cubran todo sin repetir elementos.

Def.

Input: Conjuntos disjuntos X, Y, Z de tamaño n, y $T \subseteq X \times Y \times Z$.

Output: ¿Hay n ternas de T que cubran cada elemento exactamente una vez? Un conjunto así es un *3d-matching perfecto*.

$$\mathrm{3DM} = \{\langle X, Y, Z, T\rangle : T \text{ admite un 3d-matching perfecto}\}.$$

Teo.

$\mathrm{3DM}$ es NP-completo.

- ¿En NP?

3D-Matching

- Formar ternas que cubran todo sin repetir elementos.

Def.

Input: Conjuntos disjuntos X, Y, Z de tamaño n, y $T \subseteq X \times Y \times Z$.

Output: ¿Hay n ternas de T que cubran cada elemento exactamente una vez? Un conjunto así es un *3d-matching perfecto*.

$$\mathrm{3DM} = \{\langle X, Y, Z, T\rangle : T \text{ admite un 3d-matching perfecto}\}.$$

Teo.

$\mathrm{3DM}$ es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

3D-Matching

- Formar ternas que cubran todo sin repetir elementos.

Def.

Input: Conjuntos disjuntos X, Y, Z de tamaño n, y $T \subseteq X \times Y \times Z$.

Output: ¿Hay n ternas de T que cubran cada elemento exactamente una vez? Un conjunto así es un *3d-matching perfecto*.

$$\mathrm{3DM} = \{\langle X, Y, Z, T\rangle : T \text{ admite un 3d-matching perfecto}\}.$$

Teo.

$\mathrm{3DM}$ es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De $\mathrm{3SAT}$ a $\mathrm{3DM}$.

Subset Sum

- Problemas aritmeticos

Subset Sum

- Problemas aritmeticos

Def.

Input: Números naturales $w_1, \ldots, w_n, W$ en binario.

Subset Sum

- Problemas aritmeticos

Def.

Input: Números naturales $w_1, \ldots, w_n, W$ en binario.

Output: ¿Hay un subconjunto de los w_i que sume W?

Subset Sum

- Problemas aritmeticos

Def.

Input: Números naturales $w_1, \ldots, w_n, W$ en binario.

Output: ¿Hay un subconjunto de los w_i que sume W?

$$\text{SUBSET-SUM} = \{\langle w_1, \ldots, w_n, W\rangle : \exists I \subseteq \{1, \ldots, n\},\ \textstyle\sum_{i \in I} w_i = W\}.$$

Subset Sum

- Problemas aritmeticos

Def.

Input: Números naturales $w_1, \ldots, w_n, W$ en binario.

Output: ¿Hay un subconjunto de los w_i que sume W?

$$\text{SUBSET-SUM} = \{\langle w_1, \ldots, w_n, W\rangle : \exists I \subseteq \{1, \ldots, n\},\ \textstyle\sum_{i \in I} w_i = W\}.$$

Teo.

SUBSET-SUM es NP-completo.

Subset Sum

- Problemas aritmeticos

Def.

Input: Números naturales $w_1, \ldots, w_n, W$ en binario.

Output: ¿Hay un subconjunto de los w_i que sume W?

$$\text{SUBSET-SUM} = \{\langle w_1, \ldots, w_n, W\rangle : \exists I \subseteq \{1, \ldots, n\},\ \textstyle\sum_{i \in I} w_i = W\}.$$

Teo.

SUBSET-SUM es NP-completo.

- ¿En NP?

Subset Sum

- Problemas aritmeticos

Def.

Input: Números naturales $w_1, \ldots, w_n, W$ en binario.

Output: ¿Hay un subconjunto de los w_i que sume W?

$$\text{SUBSET-SUM} = \{\langle w_1, \ldots, w_n, W\rangle : \exists I \subseteq \{1, \ldots, n\},\ \textstyle\sum_{i \in I} w_i = W\}.$$

Teo.

SUBSET-SUM es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?

Subset Sum

- Problemas aritmeticos

Def.

Input: Números naturales $w_1, \ldots, w_n, W$ en binario.

Output: ¿Hay un subconjunto de los w_i que sume W?

$$\text{SUBSET-SUM} = \{\langle w_1, \ldots, w_n, W\rangle : \exists I \subseteq \{1, \ldots, n\},\ \textstyle\sum_{i \in I} w_i = W\}.$$

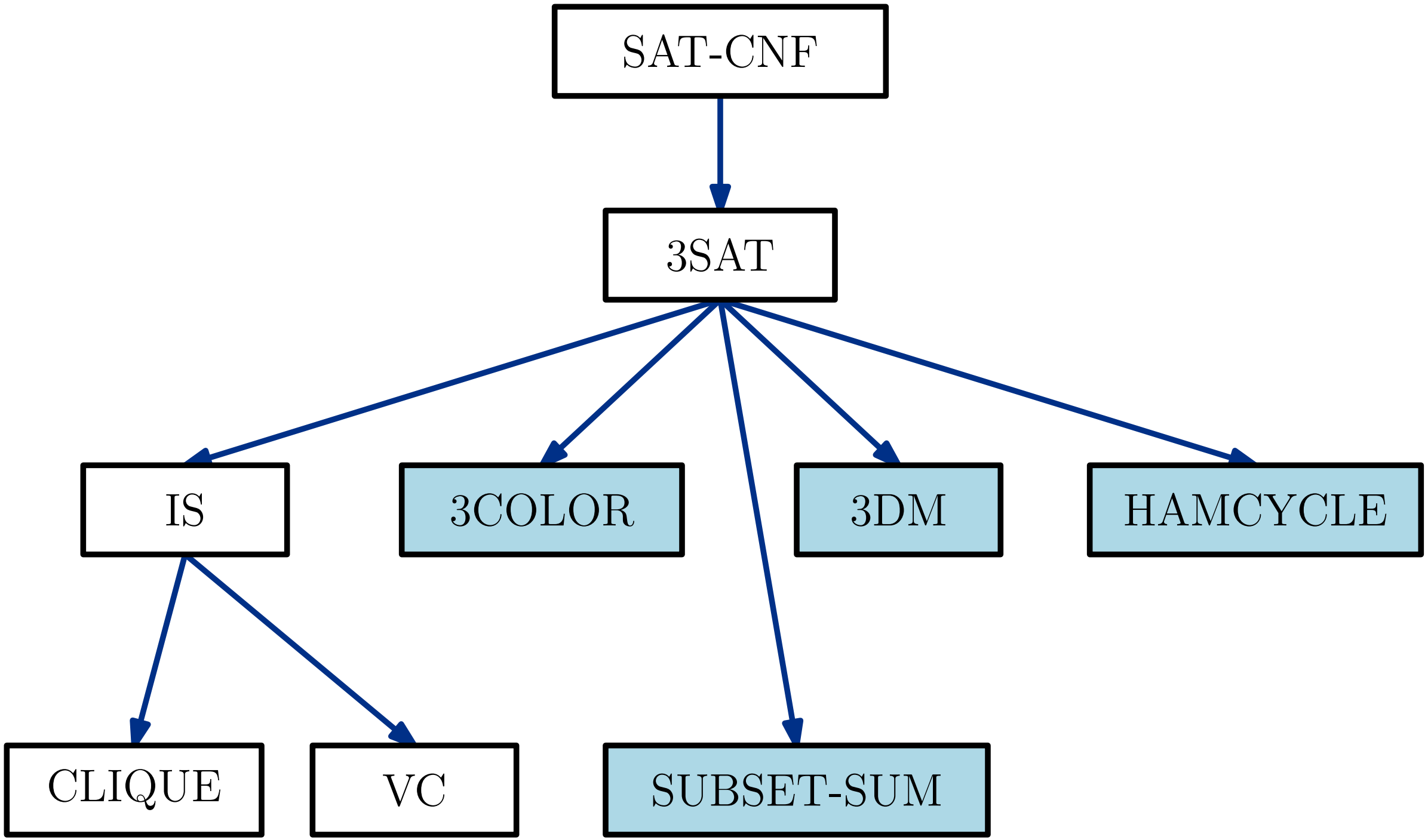
Teo.

SUBSET-SUM es NP-completo.

- ¿En NP?
- ¿Desde qué problema reducimos?
  - De 3SAT a SUBSET-SUM.

Red de reducciones y tipos de problemas

La red de reducciones



Problemas por tipo

- ¿Qué problema conocido se parece al que queremos resolver?

Problemas por tipo

- ¿Qué problema conocido se parece al que queremos resolver?

- **Empaquetamiento**
  Elegir objetos sin conflictos.

IS y CLIQUE.

Problemas por tipo

- ¿Qué problema conocido se parece al que queremos resolver?

- **Empaquetamiento**
  Elegir objetos sin conflictos.

  IS y CLIQUE.

- **Cubrimiento**
  Cubrir todos los requerimientos.

  VC.

Problemas por tipo

- ¿Qué problema conocido se parece al que queremos resolver?

- **Empaquetamiento**
  Elegir objetos sin conflictos.

  IS y CLIQUE.

- **Cubrimiento**
  Cubrir todos los requerimientos.

  VC.

- **Particionado**
  Dividir los objetos en grupos válidos.

  3DM y 3COLOR.

Problemas por tipo

- ¿Qué problema conocido se parece al que queremos resolver?

- **Empaquetamiento**
  Elegir objetos sin conflictos.

  IS y CLIQUE.

- **Cubrimiento**
  Cubrir todos los requerimientos.

  VC.

- **Particionado**
  Dividir los objetos en grupos válidos.

  3DM y 3COLOR.

- **Secuenciamiento**
  Ordenar objetos bajo restricciones.

  HAMCYCLE.

Problemas por tipo

- ¿Qué problema conocido se parece al que queremos resolver?

- **Empaquetamiento**
  Elegir objetos sin conflictos.

  IS y CLIQUE.

- **Cubrimiento**
  Cubrir todos los requerimientos.

  VC.

- **Particionado**
  Dividir los objetos en grupos válidos.

  3DM y 3COLOR.

- **Secuenciamiento**
  Ordenar objetos bajo restricciones.

  HAMCYCLE.

- **Problemas numéricos**
  Elegir objetos con cierta suma.

  SUBSET-SUM.

Problemas por tipo

- ¿Qué problema conocido se parece al que queremos resolver?

- **Empaquetamiento**
  Elegir objetos sin conflictos.

  IS y CLIQUE.

- **Cubrimiento**
  Cubrir todos los requerimientos.

  VC.

- **Particionado**
  Dividir los objetos en grupos válidos.

  3DM y 3COLOR.

- **Secuenciamiento**
  Ordenar objetos bajo restricciones.

  HAMCYCLE.

- **Problemas numéricos**
  Elegir objetos con cierta suma.

  SUBSET-SUM.

- **Satisfacción de restricciones**
  Asignar valores que cumplan las condiciones.

  SAT-CNF y 3SAT.

Tres reducciones más

Tres reducciones más

- **TSP. Input:** Distancias d entre n ciudades y una cota D.

Tres reducciones más

- **TSP. Input:** Distancias d entre n ciudades y una cota D.

  Output: ¿Hay un tour de largo $\leq D$ que visite cada ciudad una vez y regrese al inicio?

Tres reducciones más

- **TSP. Input:** Distancias d entre n ciudades y una cota D.

  Output: ¿Hay un tour de largo $\leq D$ que visite cada ciudad una vez y regrese al inicio?

  - De $\mathrm{HAMCYCLE}$ a TSP.

Tres reducciones más

- **TSP. Input:** Distancias d entre n ciudades y una cota D.

  Output: ¿Hay un tour de largo $\leq D$ que visite cada ciudad una vez y regrese al inicio?

  - De $\mathrm{HAMCYCLE}$ a TSP.

- **Set Cover. Input:** Una familia $\mathcal{S}$ de subconjuntos de U y una cota k.

Tres reducciones más

- **TSP. Input:** Distancias d entre n ciudades y una cota D.

  Output: ¿Hay un tour de largo $\leq D$ que visite cada ciudad una vez y regrese al inicio?

  - De $\mathrm{HAMCYCLE}$ a TSP.

- **Set Cover. Input:** Una familia $\mathcal{S}$ de subconjuntos de U y una cota k.

  Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

Tres reducciones más

- **TSP. Input:** Distancias d entre n ciudades y una cota D.

  Output: ¿Hay un tour de largo $\leq D$ que visite cada ciudad una vez y regrese al inicio?

  - De $\mathrm{HAMCYCLE}$ a TSP.

- **Set Cover. Input:** Una familia $\mathcal{S}$ de subconjuntos de U y una cota k.

  Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

  - De VC a SC.

Tres reducciones más

- **TSP. Input:** Distancias d entre n ciudades y una cota D.

  Output: ¿Hay un tour de largo $\leq D$ que visite cada ciudad una vez y regrese al inicio?

  - De $\mathrm{HAMCYCLE}$ a TSP.

- **Set Cover. Input:** Una familia $\mathcal{S}$ de subconjuntos de U y una cota k.

  Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

  - De VC a SC.

- **Knapsack. Input:** Pesos w_i, valores v_i $(i = 1, \ldots, n)$ y cotas W, V, todos naturales en binario.

Tres reducciones más

- **TSP. Input:** Distancias d entre n ciudades y una cota D.

  Output: ¿Hay un tour de largo $\leq D$ que visite cada ciudad una vez y regrese al inicio?

  - De $\mathrm{HAMCYCLE}$ a TSP.

- **Set Cover. Input:** Una familia $\mathcal{S}$ de subconjuntos de U y una cota k.

  Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

  - De VC a SC.

- **Knapsack. Input:** Pesos w_i, valores v_i $(i = 1, \ldots, n)$ y cotas W, V, todos naturales en binario.

  Output: ¿Hay $I \subseteq \{1, \ldots, n\}$ con $\sum_{i \in I} w_i \leq W$ y $\sum_{i \in I} v_i \geq V$?

Tres reducciones más

- **TSP. Input:** Distancias d entre n ciudades y una cota D.

  Output: ¿Hay un tour de largo $\leq D$ que visite cada ciudad una vez y regrese al inicio?

  - De $\mathrm{HAMCYCLE}$ a TSP.

- **Set Cover. Input:** Una familia $\mathcal{S}$ de subconjuntos de U y una cota k.

  Output: ¿Hay a lo más k conjuntos de $\mathcal{S}$ cuya unión sea U?

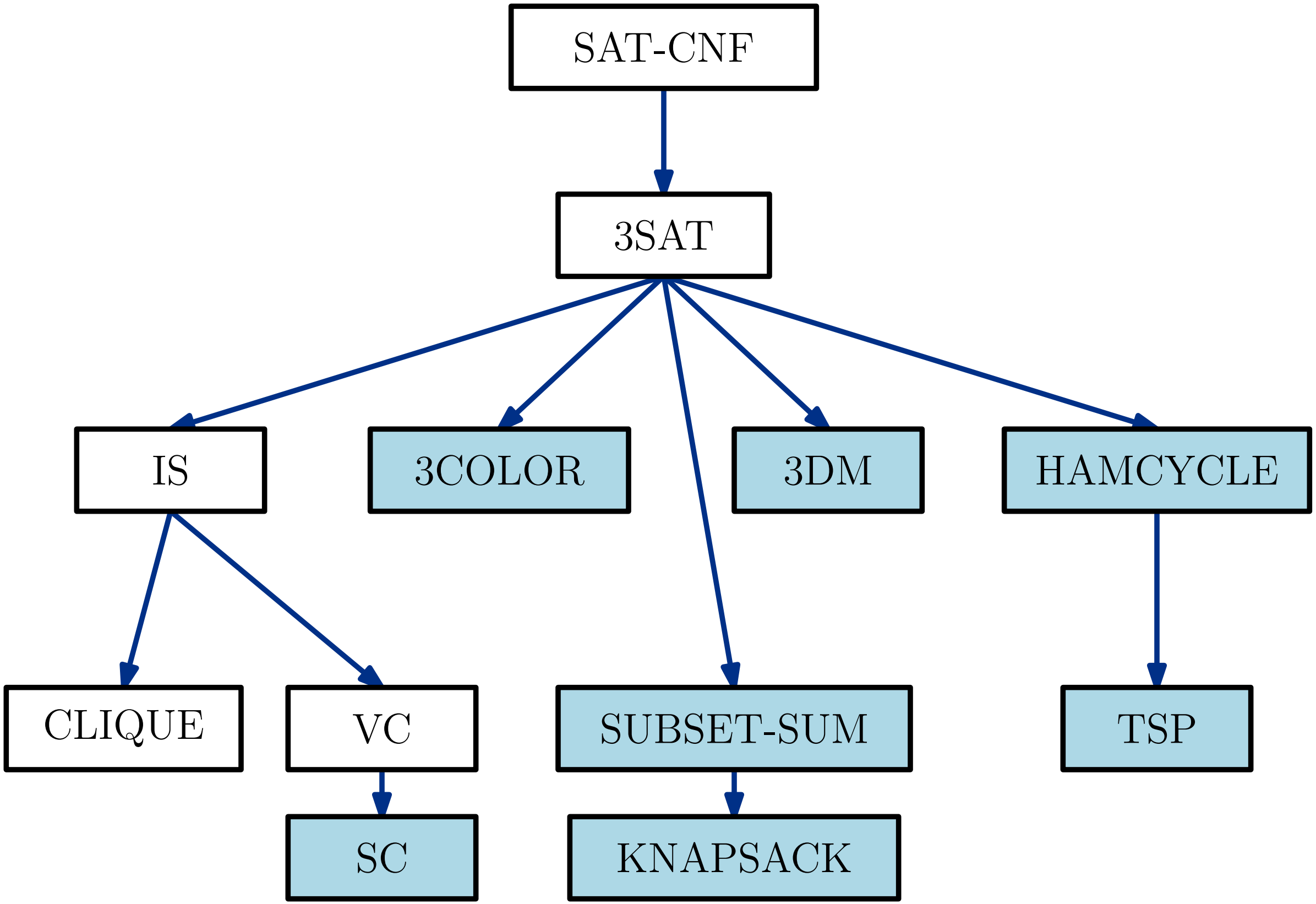
  - De VC a SC.

- **Knapsack. Input:** Pesos w_i, valores v_i $(i = 1, \ldots, n)$ y cotas W, V, todos naturales en binario.

  Output: ¿Hay $I \subseteq \{1, \ldots, n\}$ con $\sum_{i \in I} w_i \leq W$ y $\sum_{i \in I} v_i \geq V$?

  - De $\mathrm{SUBSET\text{-}SUM}$ a $\mathrm{KNAPSACK}$.

La red de reducciones, ampliada



Dudas/Preguntas