Sommario

Esempi di problemi NP-completi:

3-SAT

CLIQUE

VERTEX-COVER

INDEPENDENT-SET

3-COLORING

HamCycle

TSP

3SAT

Consideriamo formule booleane in forma normale congiuntiva (CNF) con esattamente 3 letterali per clausola e chiamiamo il loro insieme 3CNF.

Per esempio:

$\varphi = (X_1 \vee X_1 \vee X_2) \wedge (\neg X_1 \vee \neg X_2 \vee \neg X_2) \wedge (\neg X_1 \vee X_2 \vee X_2)$

brevemente chiamate formule in 3CNF

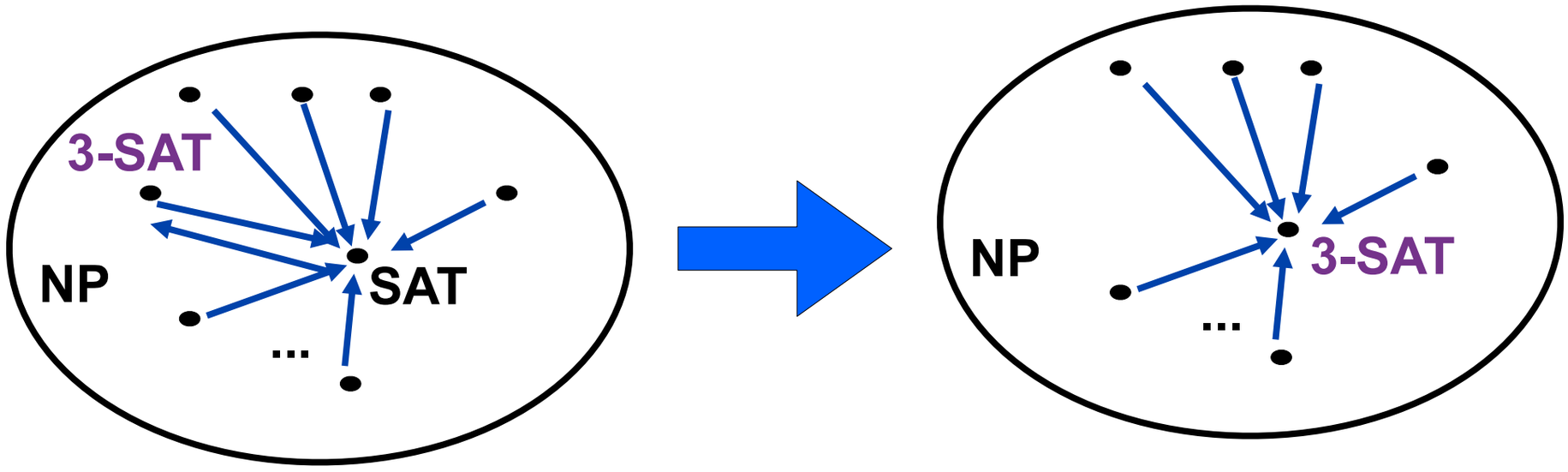
3-SAT = {φ | φ è in 3CNF e soddisfacibile}

Teorema. 3-SAT è NP-completo

3SAT

Teorema. 3-SAT è NP-completo

Proviamo che 3-SAT è NP-hard, per riduzione da SAT.



Sia $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_n$ una formula in CNF. Associeremo a φ una formula in 3CNF che sarà soddisfacibile se e solo se lo è φ. Trasformeremo le clausole di φ in formule in 3CNF, in tempo polinomiale.

3SAT è NP-completo

Sia quindi C una delle clausole, distinguiamo tre casi, qui X, Y, X_i sono letterali

$C = X$ prendiamo $C' = X \vee X \vee X$

$C = X \vee Y$, prendiamo $C' = X \vee X \vee Y$

$C = X_1 \vee X_2 \vee \ldots \vee X_n$, con n>3, allora introduciamo nuove variabili in modo da ottenere una formula in 3CNF:

$$\varphi_C =$$

$$(X_1 \vee X_2 \vee Z_1) \wedge \ldots \wedge (\neg Z_{i-2} \vee X_i \vee Z_{i-1}) \wedge \ldots \wedge (\neg Z_{n-3} \vee X_{n-1} \vee X_n)$$

(serve una variabile per le prime due e una per tutte le altre meno le ultime due)

3SAT è NP-completo, 2

$C = X_1 \vee X_2 \vee \ldots \vee X_n$ è soddisfacibile $\Leftrightarrow$ φ_C =

$(X_1 \vee X_2 \vee Z_1) \wedge \ldots \wedge (\neg Z_{i-2} \vee X_i \vee Z_{i-1}) \wedge \ldots \wedge (\neg Z_{n-3} \vee X_{n-1} \vee X_n)$
lo è.

$\Rightarrow$Sia C soddisfacibile allora esiste un i, $1 \leq i \leq n$, tale che il letterale X_i ha valore vero. Allora φ_C è soddisfatta ponendo Z_j a vero per ogni j= 1,..., i-2, e ponendo Z_j a falso per ogni j= i-1,...,n-3, (saranno tutti a falso se $i \leq 2$ e tutti a vero se $i \geq n-1$)

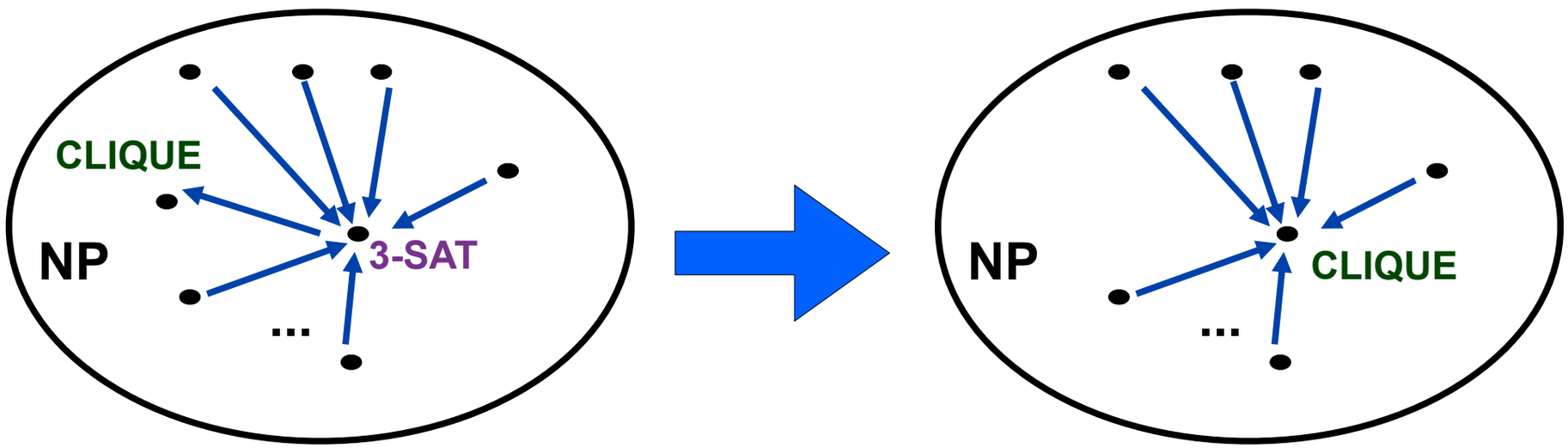
$\Leftarrow$Sia φ_C soddisfacibile, basta osservare che allora non tutti gli X_i possono essere a falso perchè altrimenti Z_1 dovrebbe essere vero e quindi Z_2 ,...,fino a Z_{n-3} dovrebbero essere vere ma allora l'ultima clausola sarebbe falsa.

CLIQUE è NP-completo

Il problema del CLIQUE
istanza (G,k) dove G è un grafo non diretto e k>0
istanza SI: G ha un k-clique

CLIQUE = {<G,k> | G è un grafo non diretto con un k-clique}

1. **CLIQUE è in NP**
2. **3-SAT $\leq_P$ CLIQUE.**

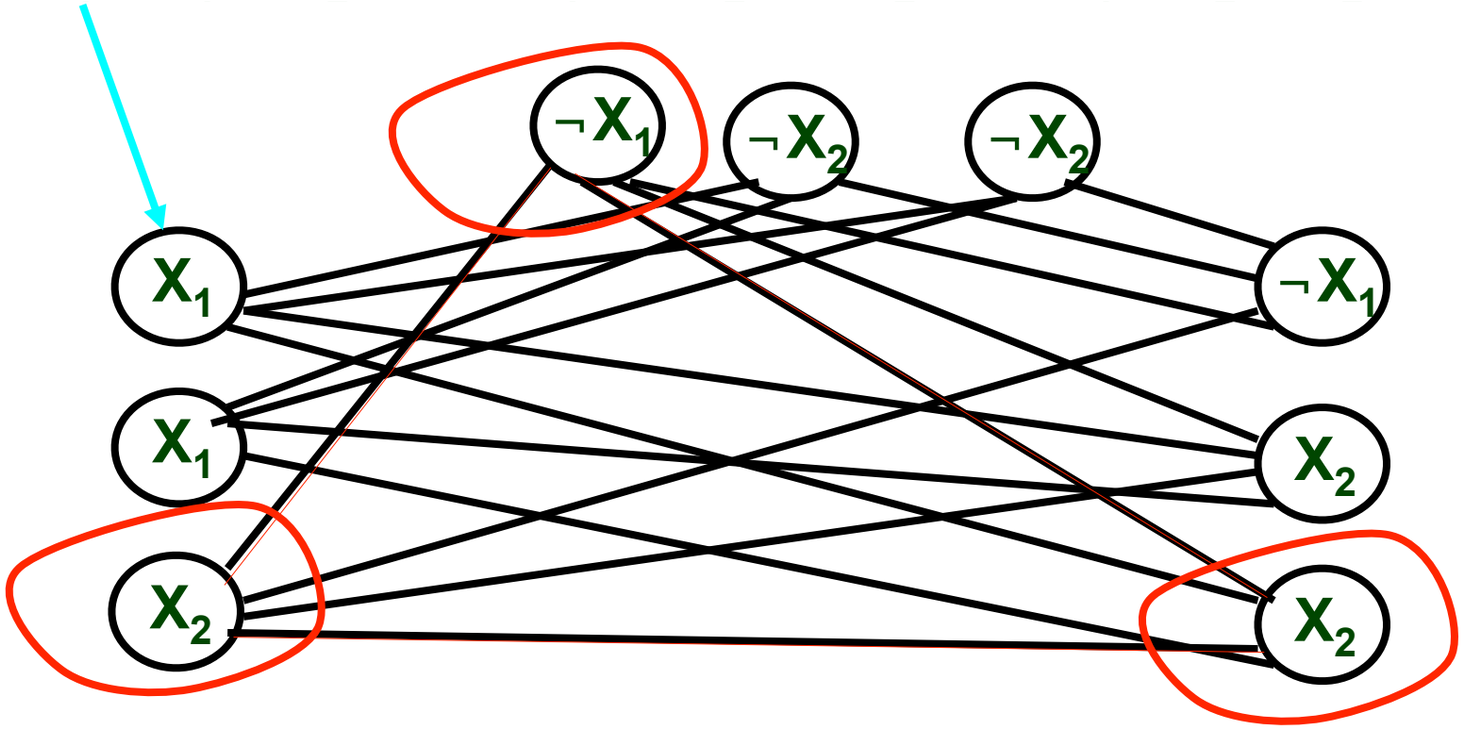


CLIQUE è NP-completo

Data una formula φ in 3CNF, con k clausole, costruiremo un grafo G_φ in modo tale che

φ è soddisfacibile $\Leftrightarrow$ G_φ ha un k-clique

$$\varphi = (X_1^{0} \vee X_1 \vee X_2^{1}) \wedge (\neg X_1 \vee \neg X_2 \vee \neg X_2) \wedge (\neg X_1 \vee X_2 \vee X_2)$$



CLIQUE è NP-completo

φ è soddisfacibile $\Longrightarrow$ G_φ ha un k-clique

Se φ è soddisfacibile in ognuna delle k clausole c'è un letterale vero, quei letterali formano un k clique, infatti i nodi corrispondenti sono connessi perchè scelti tra clausole differenti e certamente i letterali non sono uno il negato di un altro.

G_φ ha un k-clique $\Longrightarrow$ φ è soddisfacibile

Se G_φ ha un k-clique, i nodi provengono da clausole diverse. Assegnamo valore vero ai letterali corrispondenti; questo assegnamento non è contradditorio perchè tra i k letterali non ce ne sono due l'uno il negato dell'altro. La formula φ è soddisfatta perchè in ogni clausola c'è un letterale vero.

IndependentSET è NP-hard

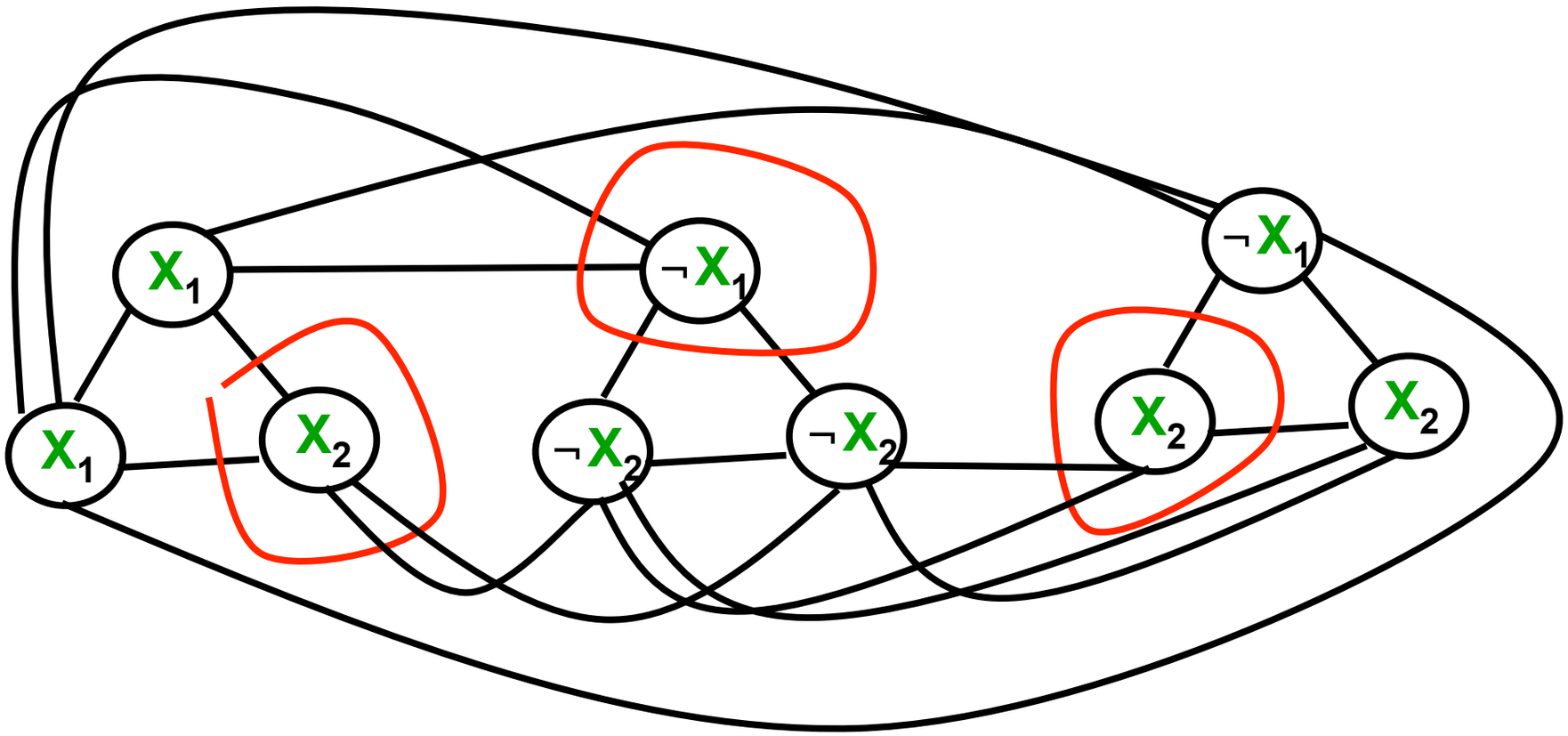
Proviamo che
IndependentSET = {<G,k> | G è un grafo non diretto con un sottoinsieme di k vertici non connessi tra loro} è NP-hard, per riduzione da 3-SAT.

Data una formula φ in 3CNF, con k clausole, costruiremo un grafo G_φ in modo tale che

φ è soddisfacibile $\Leftrightarrow$ G_φ ha un k-independentSET

IndependentSET è NP-hard

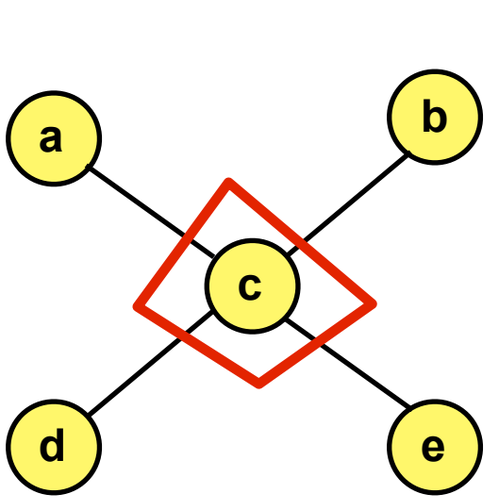
$$\varphi = (x_1 \vee x_1 \vee \overset{1}{x_2}) \wedge (\neg \overset{1}{x_1} \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee \overset{1}{x_2} \vee x_2)$$



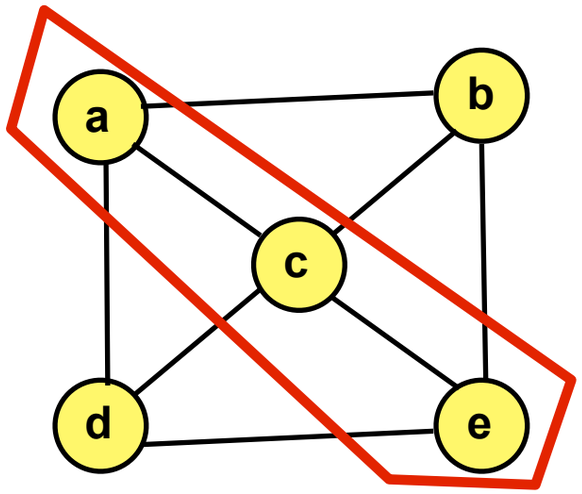
φ è soddisfacibile $\Leftrightarrow$ G_φ ha un k-independentSET

VERTEX-COVER è NP-completo

Un vertex cover in un grafo non diretto è un sottoinsieme dei vertici con la proprietà che ogni arco di G ne tocca uno. Un k-vertex cover è un vertex cover con k nodi.



1-vertex cover



3-vertex cover

VERTEX-COVER è NP-completo

Il problema del VERTEX-COVER
istanza (G,k) dove G è un grafo non diretto e k>0
istanza SI: G ha un k-vertex-cover

VERTEX-COVER =
{<G,k> | G è un grafo non diretto con un k-vertex cover}

Teorema. VERTEX-COVER è NP-completo

1. **VERTEX-COVER è in NP**
2. **3-SAT $\leq_P$ VERTEX-COVER.**

VERTEX-COVER è NP-completo

Il problema del VERTEX-COVER è in NP: il certificato è un sottoinsieme di k vertici di G.

Dimostriamo che VERTEX-COVER è NP-hard via una riduzione polinomiale da 3-SAT.

Data una formula φ in 3CNF, con c clausole e v variabili, costruiremo un grafo G_φ, con 3c+2v nodi, in modo tale che

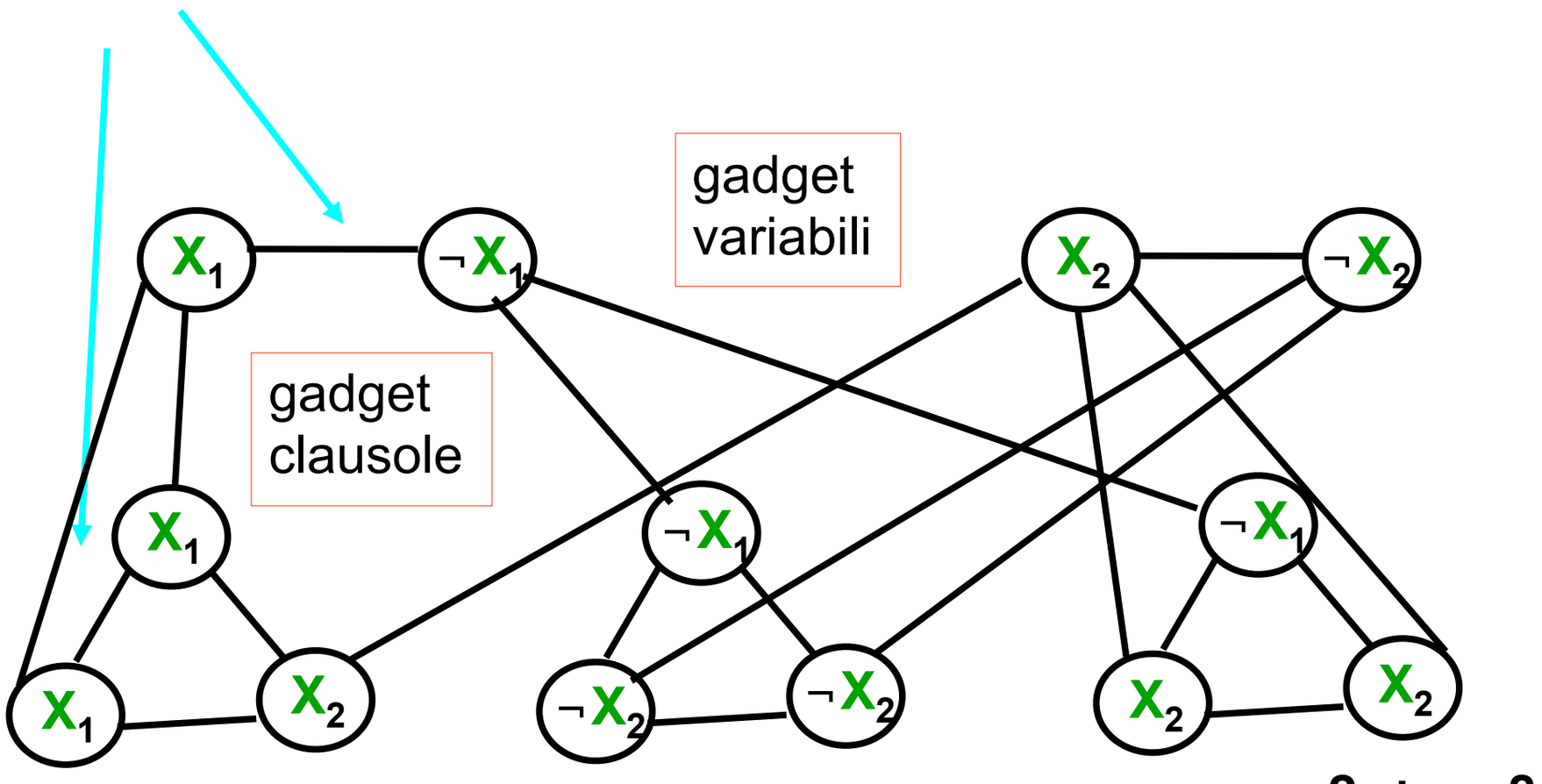
φ è soddisfacibile $\Leftrightarrow$ G_φ ha un 2c+v -vertex cover

VERTEX-COVER è NP-hard

costruzione di G_φ

c = 3 e v = 2

$$\varphi = (X_1 \vee X_1 \vee X_2) \wedge (\neg X_1 \vee \neg X_2 \vee \neg X_2) \wedge (\neg X_1 \vee X_2 \vee X_2)$$

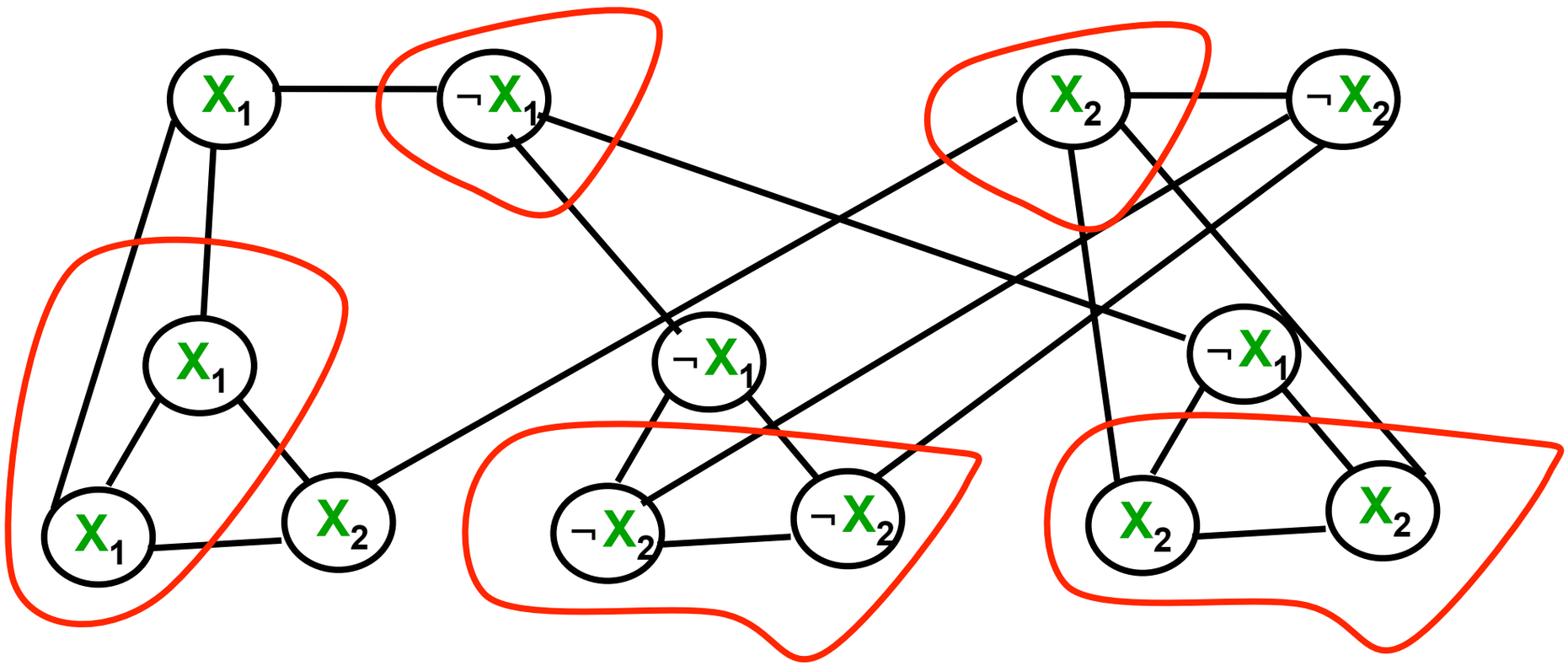


2c+v = 8

VERTEX-COVER è NP-hard

φ **è soddisfacibile** $\Longrightarrow$ G_φ **ha un 2c+v -vertex cover**

$$\varphi = (X_1 \vee X_1 \vee \overset{1}{X_2}) \wedge (\neg X_1 \vee \neg X_2 \vee \neg X_2) \wedge (\neg X_1 \vee X_2 \vee X_2)$$

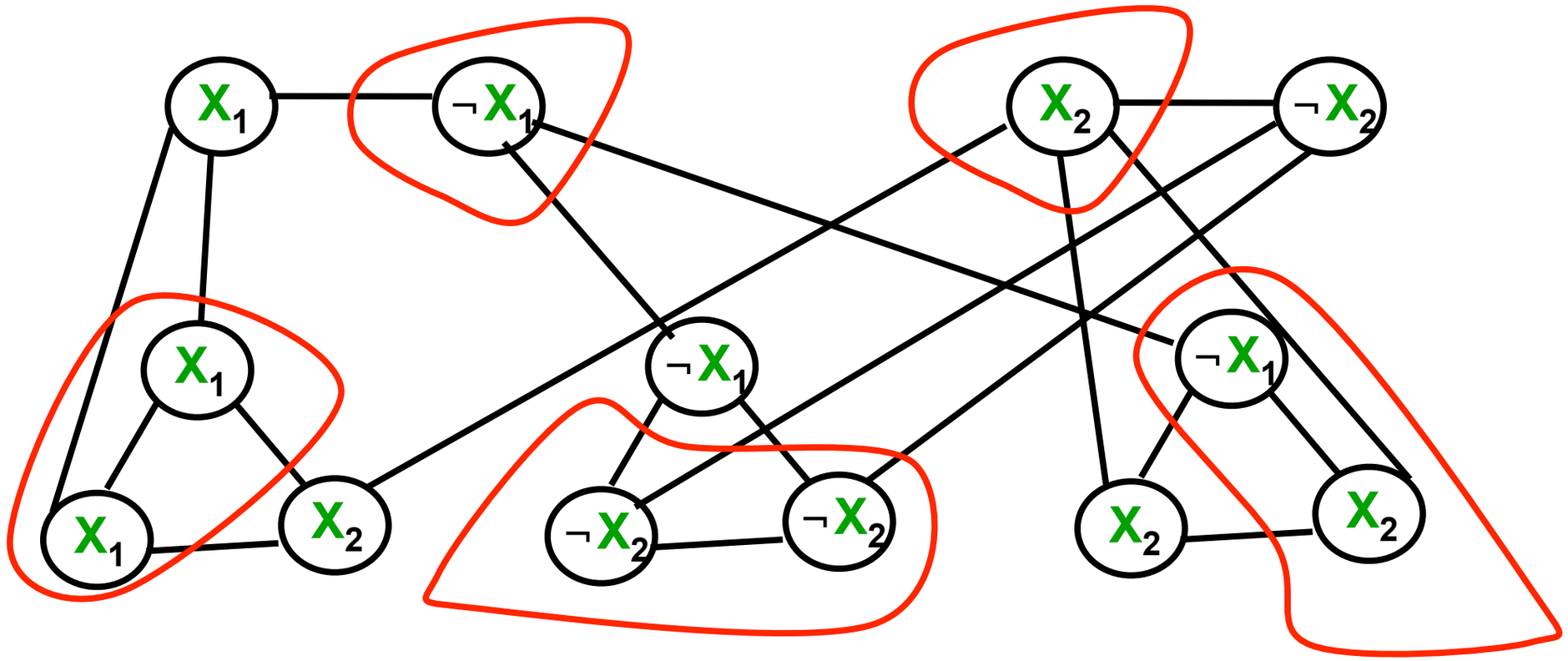


Prendiamo un nodo in ciascuno dei gadget delle variabili la cui etichetta sia vera nell'assegnamento e per ogni gadget di clausola i due nodi non connessi con quello scelto. Nell'esempio per aver preso X_2 vera nella prima clausola dobbiamo prendere X_1 due volte nel primo gadget clausola.

VERTEX-COVER è NP-hard

G_φ ha un 2c+v -vertex cover $\Longrightarrow$ φ è soddisfacibile

$$\varphi = (X_1 \vee X_1 \vee \overset{1}{X_2}) \wedge (\overset{1}{\neg X_1} \vee \neg X_2 \vee \neg X_2) \wedge (\neg X_1 \vee X_2 \vee \overset{1}{X_2})$$



In un 2c+v-vertex cover ci deve essere un vertice per ogni gadget delle variabili, e due dei vertici nei gadget delle clausole, e così sono 2c+v. Assegnamo valore vero alle variabili nei gadget delle variabili che sono scelti nel vertex cover.

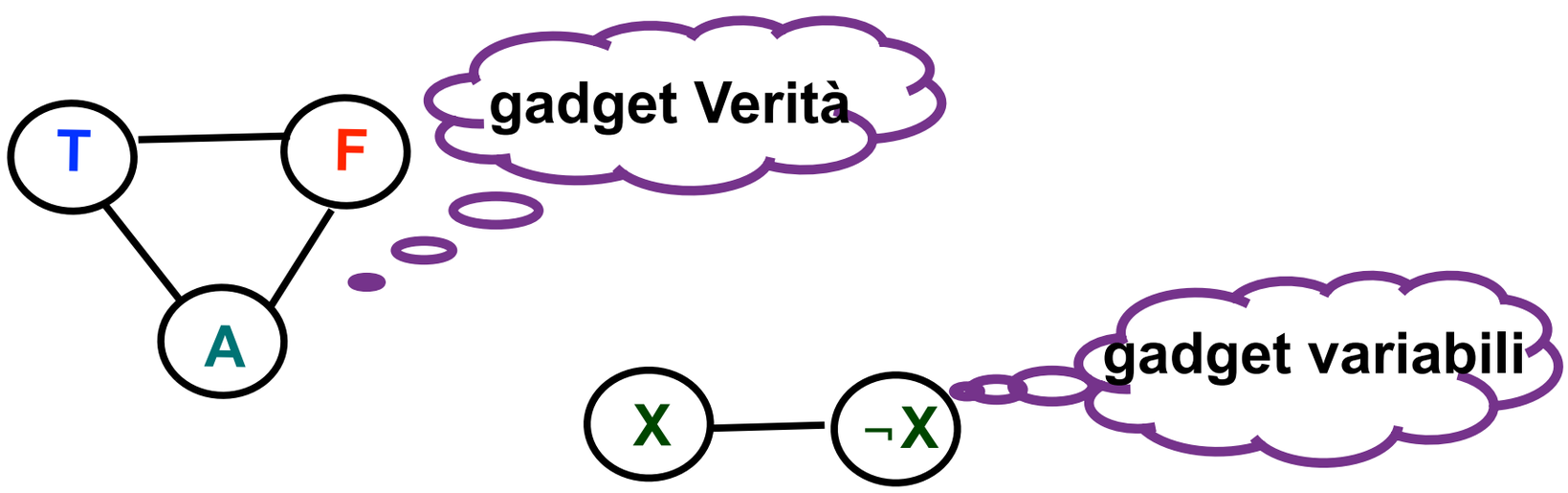
Per esercizio si mostri che CLIQUE $\leq_P$ INDEPENDENT-SET.

3-Coloring è NP-hard

Una colorazione di un grafo G=(V,E) è una funzione f : V → {1,...,n} tale che {u,v} ∈ E ⇒ f(u)≠f(v). Una 3-colorazione usa solo tre colori.

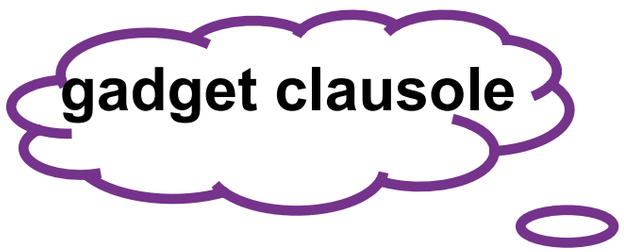
3-col = {<G> | G è un grafo non diretto che ha una 3-colorazione} è NP-hard

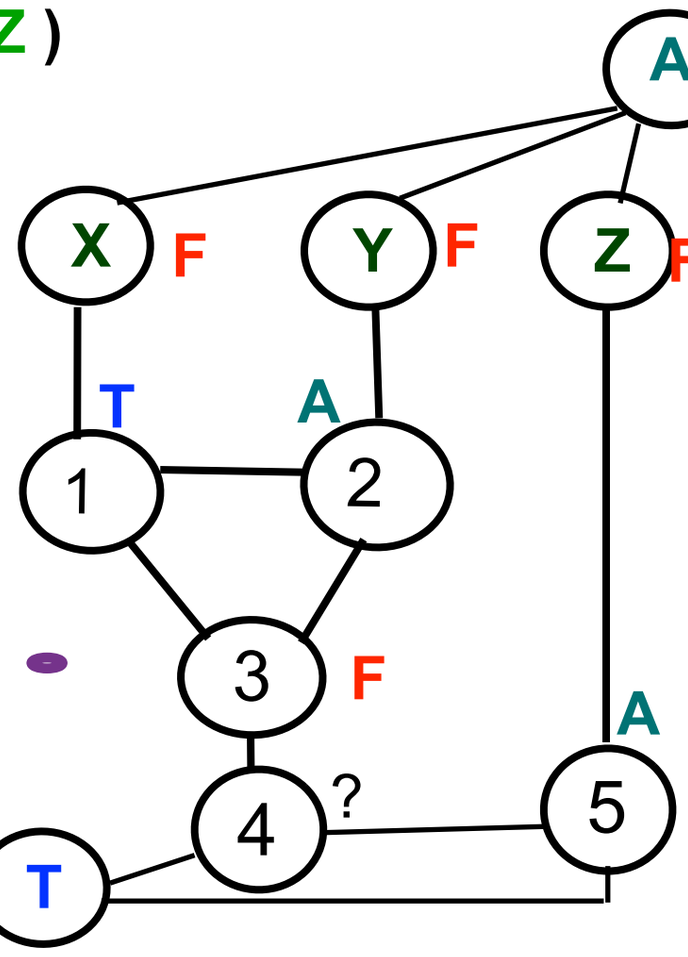
Per riduzione da 3-SAT



3-Coloring è NP-hard

Data la clausola (X v Y v Z)

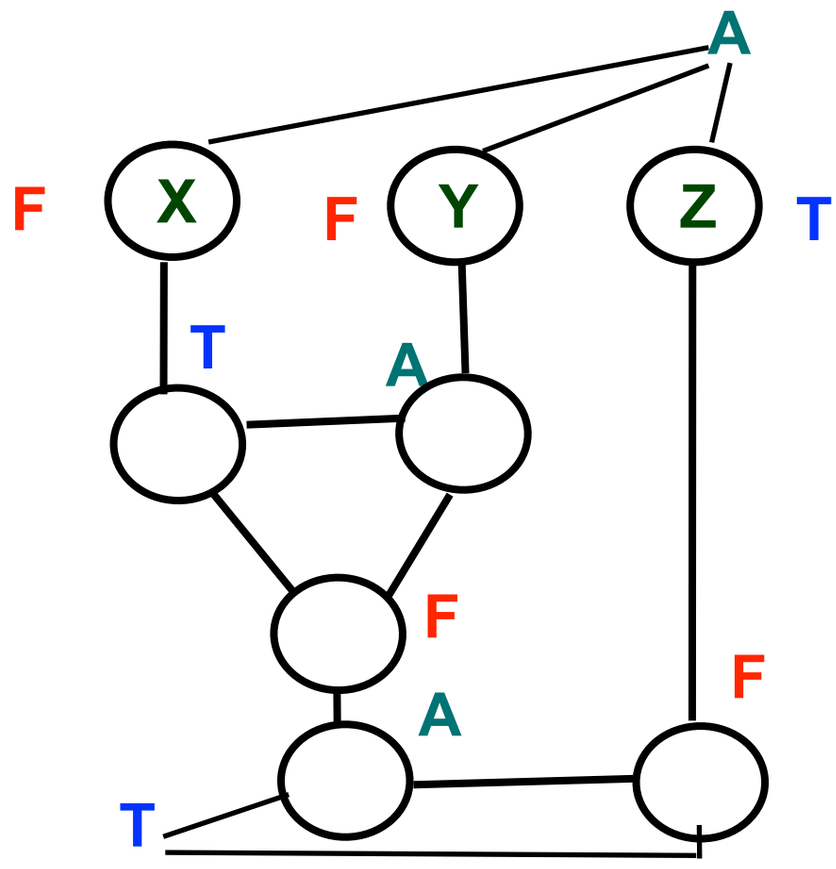




Il nodo 5 è connesso con un nodo colorato T e uno colorato F quindi non può che assumere il colore A. Ma il nodo 4 anche risulta connesso con un nodo colorato F, necessariamente, e un nodo colorato T, quindi anche 4 dovrebbe assumere il colore A.

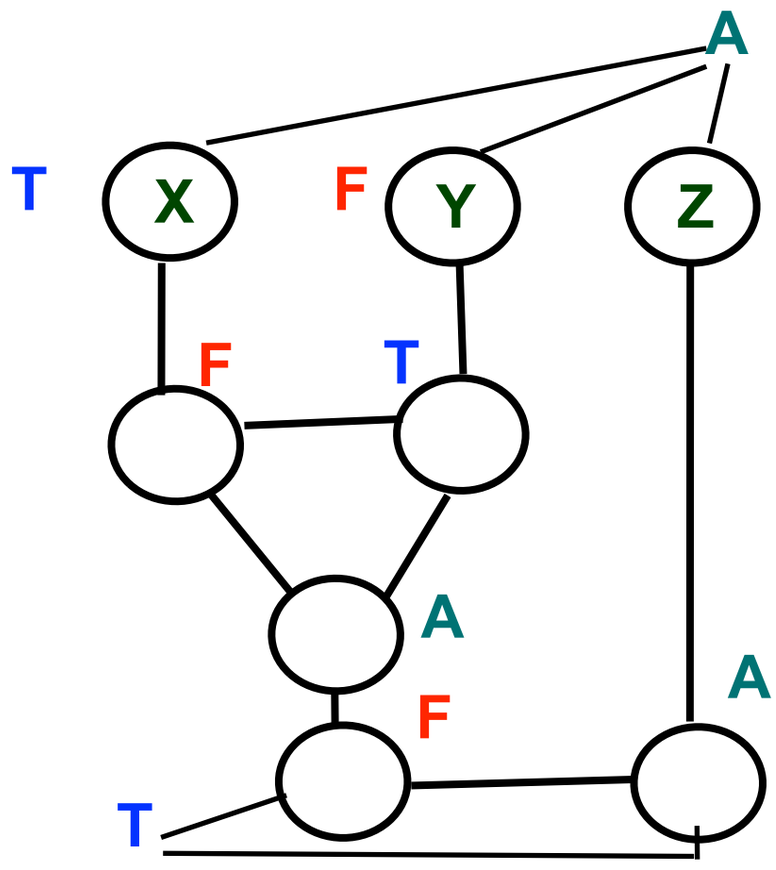
3-Coloring è NP-hard

Abbiamo visto che il gadget clausola non è colorabile se la clausola è falsa.
Basta attribuire valore vero a una variabile per ottenere la colorabilità



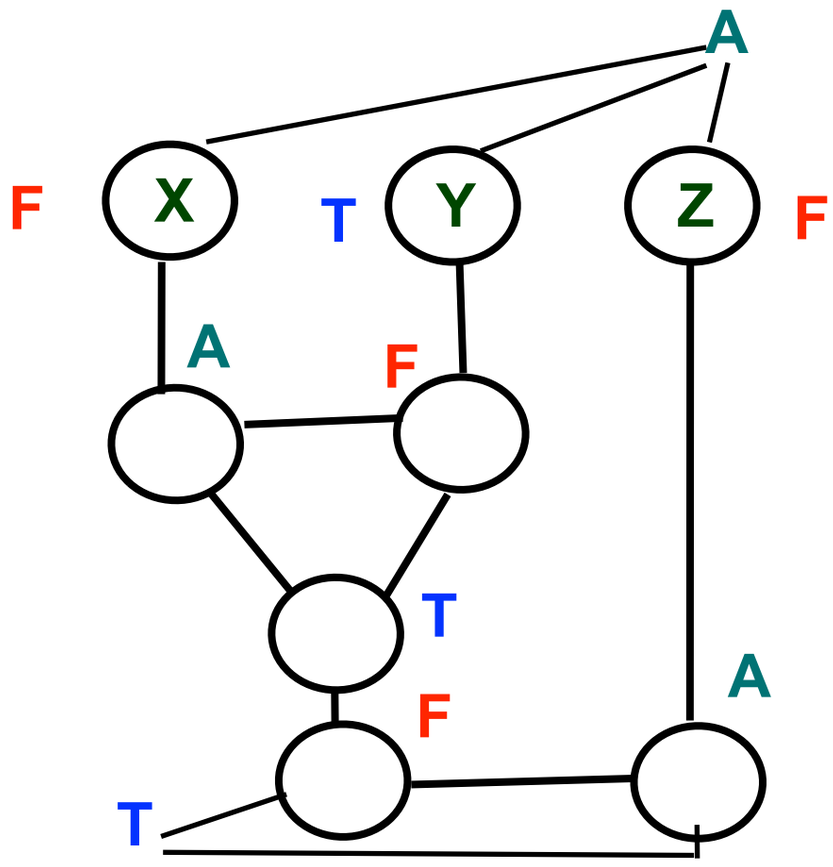
$\varphi = (X \vee Y \vee Z)$

3-Coloring è NP-hard



$\varphi = (X \vee Y \vee Z)$

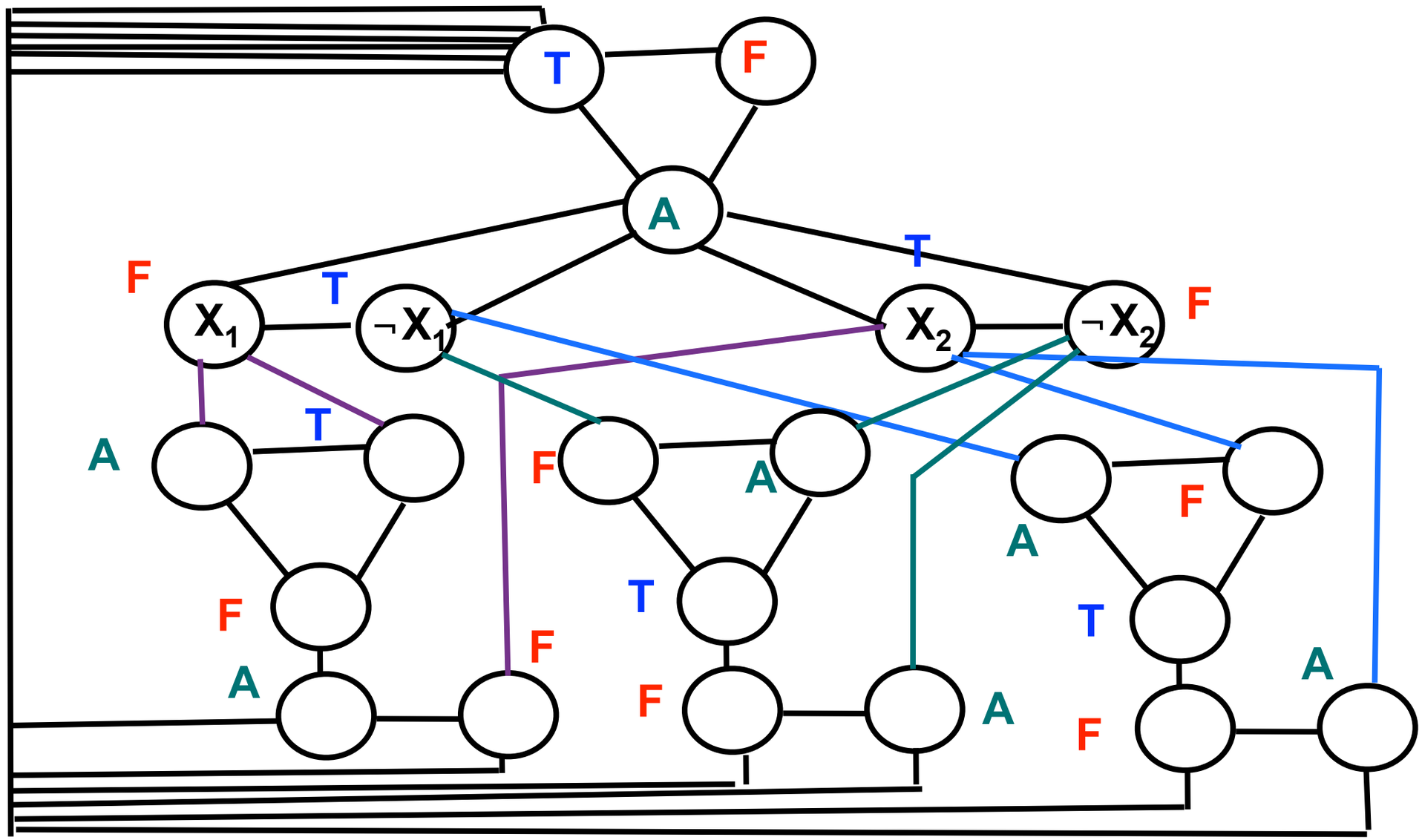
3-Coloring è NP-hard



$\varphi = (X \vee Y \vee Z)$

3-Coloring è NP-hard

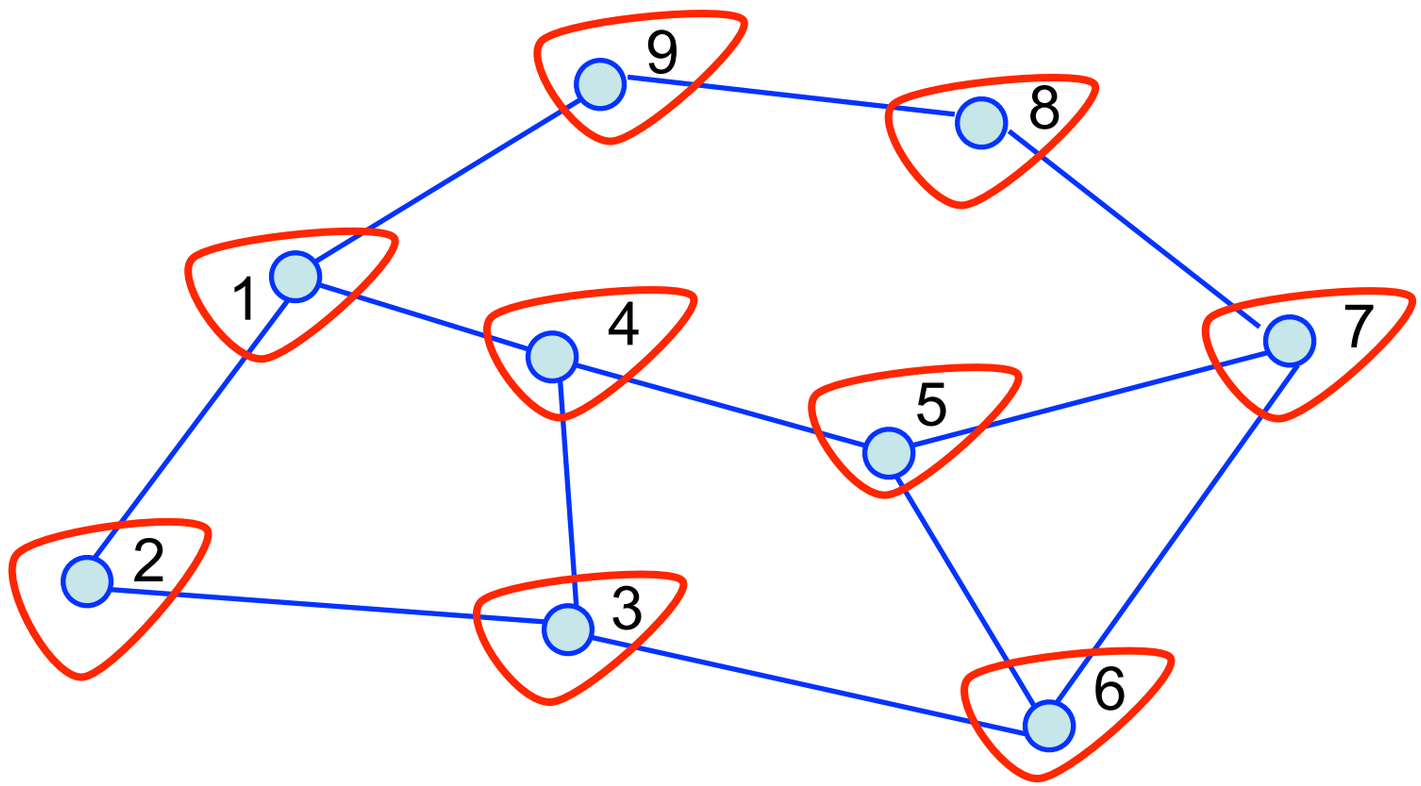
$\varphi = (X_1 \vee X_1 \vee X_2) \wedge (\neg X_1 \vee \neg X_2 \vee \neg X_2) \wedge (\neg X_1 \vee X_2 \vee X_2)$



HamCycle

HamCycle è il problema dell'esistenza di un ciclo semplice che contiene tutti i vertici, cioè un ciclo hamiltoniano, in un grafo non diretto.

HamCycle = {<G> | G è un grafo non diretto con un ciclo hamiltoniano}

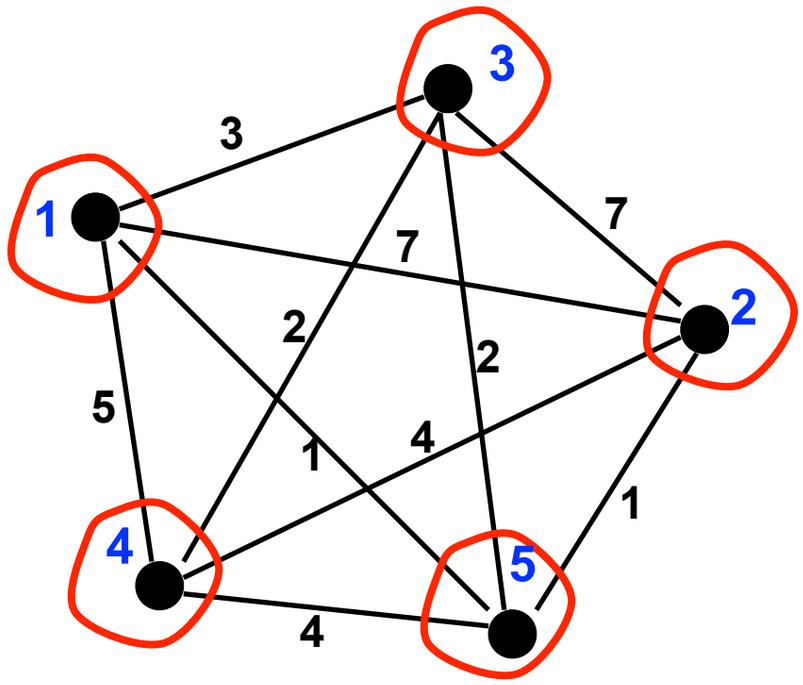


TSP

TSP è il problema decisionale associato al Traveling Salesman Problem (il problema del commesso viaggiatore):

dato un grafo G=(V,E) non diretto completo con una funzione costo a valori interi non negativi p : E → N e un intero non negativo B, si vuole sapere se esiste un ciclo hamiltoniano t su G di costo minore o uguale a B.

TSP = {<G=(V,E),p,B> | G è un grafo non diretto completo, p una funzione peso p : E → N, B in N e G ha un ciclo hamiltoniano di peso ≤ B}



B = 25

p(c)= 21

Hamcycle e TSP sono NP-completi

HamCycle = {<G> | G è un grafo non diretto con un ciclo hamiltoniano}

TSP = {<G=(V,E),p,B> | G è un grafo non diretto completo, p una funzione peso p : E → N, B in N e G ha un ciclo hamiltoniano di peso ≤ B}

Diamo per noto che Hamcycle sia NP-hard (vedi Sipser), sappiamo che è in NP, facciamo vedere che TSP è NP-completo.

Non è difficile dimostrare che TSP è in NP.

Dimostraiamo che HamCycle $\leq_p$ TSP

HamCycle $\leq_p$ TSP

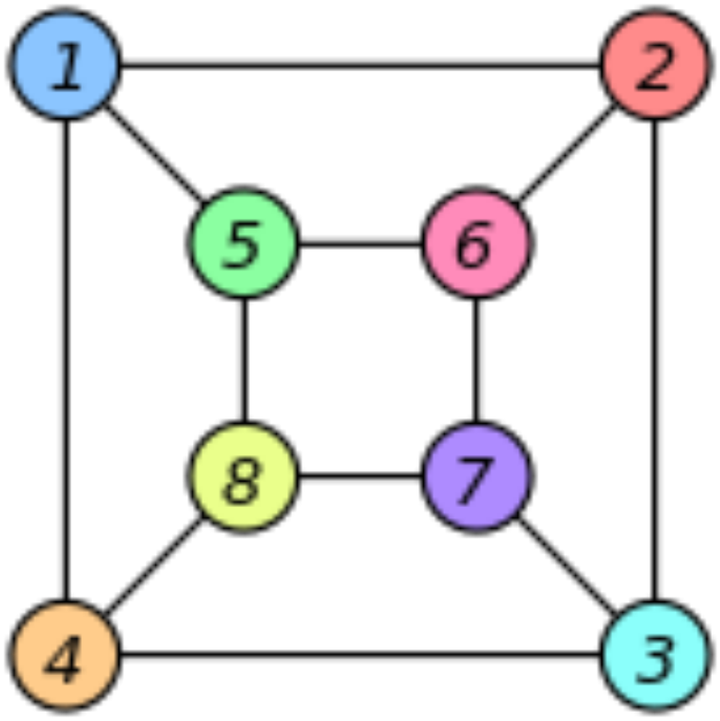
A un grafo non diretto G=(V,E) associamo un'istanza del TSP (G',p,B) in modo tale che G=(V,E) $\in$ HamCycle $\Leftrightarrow$ <G'=(V,E'),p,B> $\in$ TSP

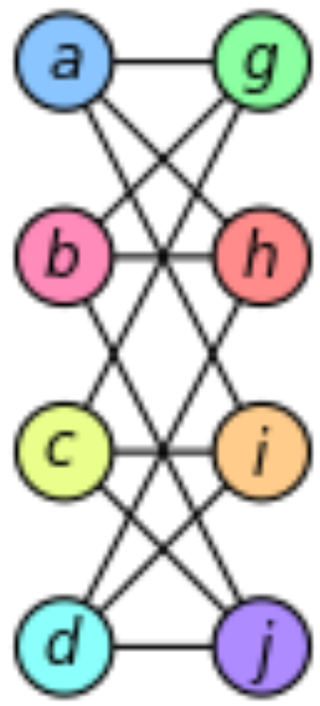
G'=(V,E') dove E'={{x,y} | x≠y $\in$ V},
p({x,y}) = 0 se {x,y} $\in$ E e p({x,y}) = 1 altrimenti e
B = 0

Isomorfismo tra grafi

GI è il problema di determinare se due grafi G=(V,E) e G'=(V',E') sono isomorfi, cioè se esiste una funzione biettiva f: V -> V' tale che se {u,v} è un arco di G, allora {f(u),f(v)} è un arco di G'. Informalmente il problema può porsi come il problema di stabilire se due grafi disegnati diversamente sono lo stesso grafo.

Esempio (preso da Wikipedia):





I due grafi sono isomorfi, basta associare vertici dell'uno a quelli dell'altro con lo stesso colore.

Isomorfismo tra grafi: un problema in NP

GI = {(G,G') l G e G' sono grafi non diretti e isomorfi tra loro} è in NP

Infatti non è difficile immaginare un verificatore A che prende in input due grafi G=(V,E) e G'=(V',E') e un potenziale isomorfimo. L'isomorfismo può essere rappresentato da una sequenza di n coppie in VxV', dove n è il numero dei vertici dei due grafi.
Si verifica se le coppie sono tutte distinte e senza ripetizioni, cioè se la funzione così rappresentata è una biezione. Si verifica se per ogni arco {u,v} in G c'è l'arco {u',v'} in G' dove (u,u') e (v,v') sono nella sequenza di coppie fornita in input. Se è così dopo la verifica che in G' non ci siano archi non controllati il verificatore può dare risposta positiva.
Questo lavoro è fatto in tempo polinomiale.

Isomorfismo tra grafi: algoritmi

L'algoritmo migliore, risultato recente, dicembre 2015 poi revisionato in gennaio 2016, e non ancora pubblicato di Làszlò Babai (premio Knuth 2015) è di complessità quasi polinomiale, cioè in $2^{O((\lg n)^c)}$, per una costante c>0 fissata.

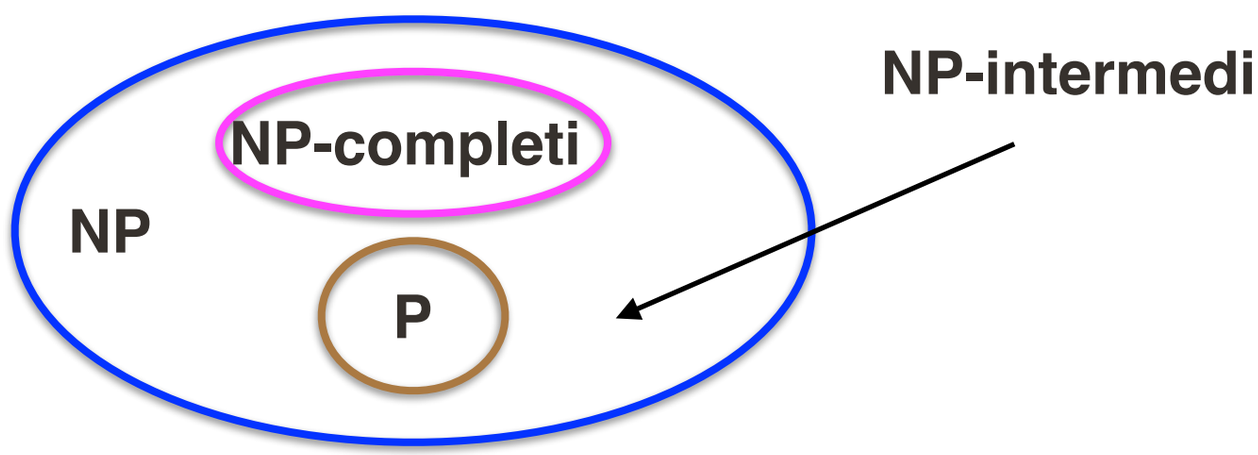
L'algoritmo migliore precedente, sempre di Babai, con Luks, è stato presentato al FOCS (IEEE Symposium on Foundations of Computer Science) nel 1983 e ha complessità in $2^{O(\sqrt{n} \lg n)}$

Per le applicazioni c'è un'euristica che lavora molto bene i cui autori sono Brendan McKay e Adolfo Piperno, che la descrivono in un articolo apparso sul Journal of Symbolic Computation, nel 2014.
Altre informazioni: http://pallini.di.uniroma1.it

Isomorfismo tra grafi è NP-completo?

Non si sa.
Quindi GI è un problema non noto essere in P e nemmeno NP-completo.

Se fosse un problema non in P e non NP-completo sarebbe un esempio concreto e "naturale" di un problema intermedio la cui esistenza è stata dimostrata da Ladner (pubblicato nel Journal of the ACM (JACM) nel 1975):
Se P≠NP allora esiste un problema in NP che non è in P né in NP-completo.
Questi problemi sono chiamati NP-intermedi. C'è una lista di problemi considerati buoni candidati a entrare nella classe degli NP-intermedi; tra questi problemi c'è stato PRIMES, fino al 2002, ora comprende per esempio sia GI che il problema della fattorizzazione. Ma ora ci si aspetta che GI esca dalla lista, per essere collocato in P.



coNP

La classe coNP contiene i complementi dei linguaggi in NP.

Esempi di problemi in coNP:

UnSAT = {ϕ | ϕ è falsa per ogni assegnamento alle variabili }

NoCLIQUE = {<G,k> | G è un grafo non diretto senza un k-clique}

coNP: altro esempio

Val = {ϕ | ϕ è vera per ogni assegnamento alle variabili }

Val è in coNP?

sì, perché unVAL è in NP (prova analoga a quella per SAT!)

Verificatore per unVAL:

Input <<φ>,c>

1. **verifica che c sia un assegnamento di valori di verità per tutte le variabili di φ**
2. **verifica se φ risulta falsa e in tal caso accetta, altrimenti rifiuta.**

PRIMES and COMPOSITES

E' facile provare che COMPOSITES è in NP.
COMPOSITES={x | x=p*q, per p,q interi e p,q≠1}
Basta dare in input a un verificatore gli interi x e p come istanza del problema e certificato.

not COMPOSITES =
PRIMES = {x | x è un numero primo}.
Quindi PRIMES è in coNP

La prova che PRIMES è in NP è meno semplice (teorema di PRATT, 1975)
Quindi PRIMES NP ∩ coNP .

Primes in P

PRIMES: Dato un intero positivo n, n è primo?

Agrawal-Kayal-Saxena hanno dimostrato nel 2002 che PRIMES si risolve con un algoritmo polinomiale. Più precisamente in $\tilde{O}(\log^{12} n)$, nel 2005 abbassato a $\tilde{O}(\log^{6} n)$ da Carl Pomerance and H. W. Lenstra, Jr..

$f(n) = \tilde{O}(g(n))$ è un'abbreviazione per
$f(n) = O(g(n) \log^{k} g(n))$ per qualche k

Factor in NP co-NP

FACTORIZE: Dato un intero x trova la sua fattorizzazione

FACTOR: Dati due interi x e U, x ha un fattore non banale minore di U?

FACTOR è in NP∩co-NP.

Un certificato per FACTOR e notFACTOR è dato da una sequenza di fattori primi.
Il verificatore controlla se si tratta di primi, in tempo polinomiale, e poi se si tratta della fattorizzazione di x.
Ora il verificatore per FACTOR risponde sì se uno dei fattori è minore di U, mentre quello per notFactor risponde sì se sono tutti maggiori di U.

P, NP, coNP e EXPTIME

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXPTIME}$$

dove

$$\mathbf{EXPTIME} = \bigcup \mathbf{TIME}(2^{n^k})$$

e

$$\mathbf{P} \subseteq \mathbf{coNP} \subseteq \mathbf{EXPTIME}$$

Prova P ⊆ coNP. Infatti P=coP (l'insieme dei complementi dei linguaggi in P) e P⊆NP ⇒ coP⊆coNP, infatti se P⊆NP allora L∈ P⇒ L ∈ NP⇒¬ L ∈ coNP, quindi P=coP⊆coNP!

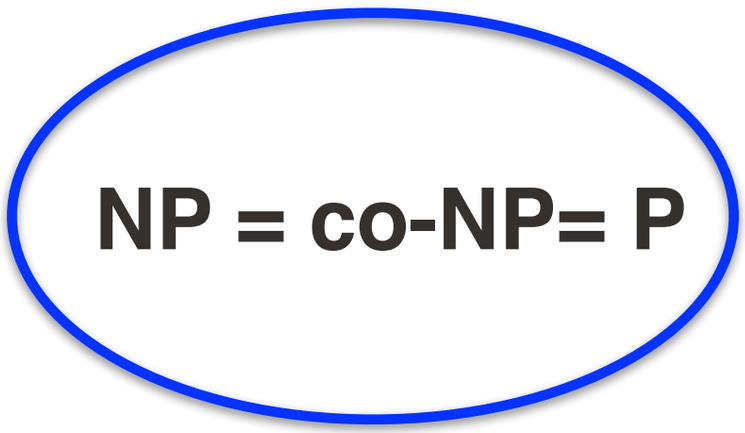
P, NP e coNP

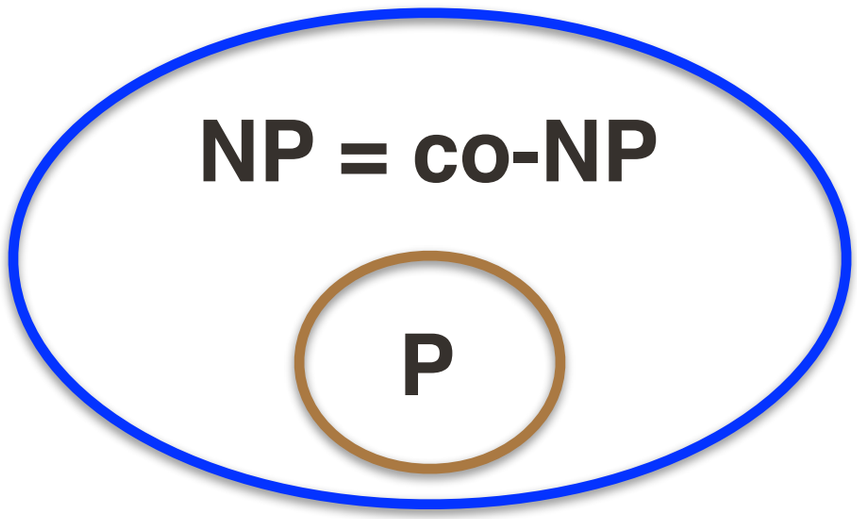
$P \subseteq NP \cap coNP$ ma $P = NP \cap coNP$?

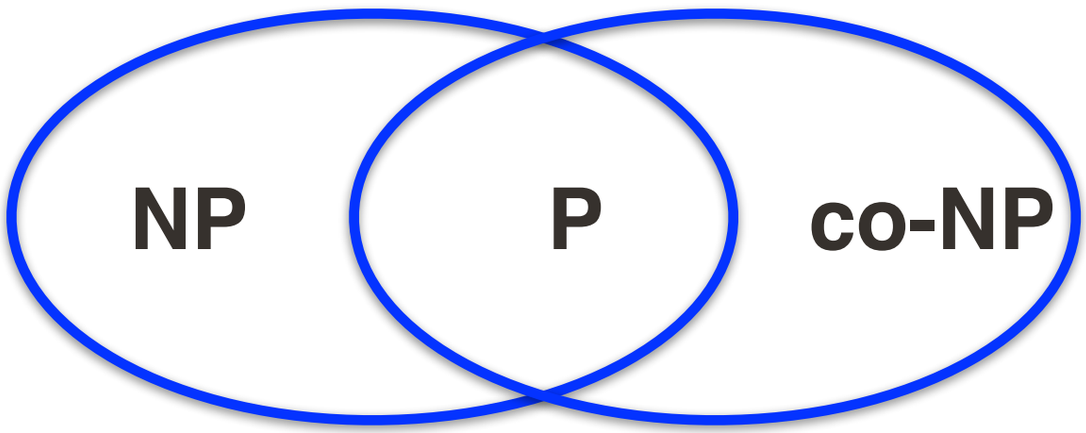
Non noto (si suppone di no)

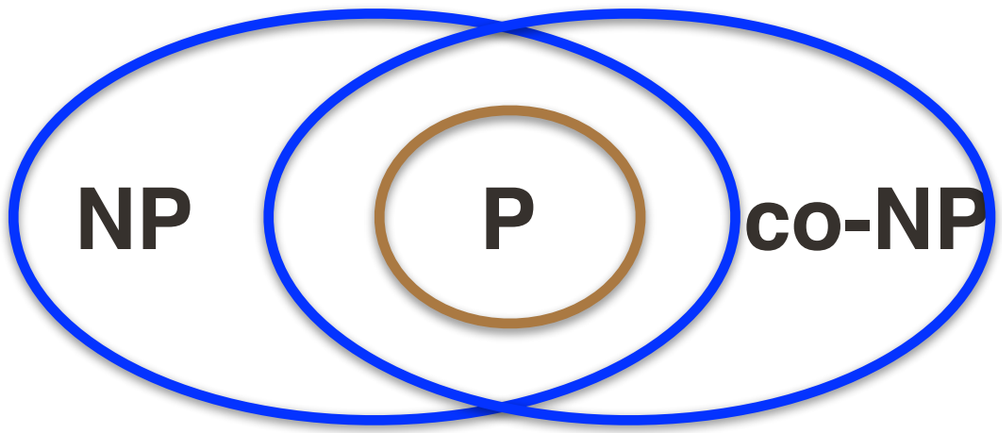
Per i problemi in $NP \cap coNP$ disponiamo di un verificatore polinomiale sia per le istanze sì che per le istanze no.
Quindi sono problemi per i quali può essere ragionevole aspettarsi l'appartenenza a P.
Tra questi c'è anche Factor.

P, NP e coNP : i possibili scenari









Factor in NP ∩co-NP

FACTORIZE: Dato un intero x trova la sua fattorizzazione

FACTOR: Dati due interi x e U, x ha un fattore non banale minore di U?

FACTOR è in NP∩co-NP.

Un certificato per FACTOR e notFACTOR è dato da una sequenza di fattori primi.
Il verificatore controlla se si tratta di primi, in tempo polinomiale, e poi se si tratta della fattorizzazione di x.
Ora il verificatore per FACTOR risponde sì se uno dei fattori è minore di U, mentre quello per notFactor risponde sì se sono tutti maggiori di U.

P, NP, coNP e EXPTIME

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXPTIME}$$

dove

$$\mathbf{EXPTIME} = \bigcup \mathbf{TIME}(2^{n^k})$$

e

$$\mathbf{P} \subseteq \mathbf{coNP} \subseteq \mathbf{EXPTIME}$$

Prova P ⊆ coNP. Infatti P=coP (l'insieme dei complementi dei linguaggi in P) e P⊆NP ⇒ coP⊆coNP, infatti se P⊆NP allora L∈ P⇒ L ∈ NP⇒¬ L ∈ coNP, quindi P=coP⊆coNP.

P, NP e coNP

$P \subseteq NP \cap coNP$ ma $P = NP \cap coNP$?

Non noto (si suppone di no)

Per i problemi in $NP \cap coNP$ disponiamo di un verificatore polinomiale sia per le istanze sì che per le istanze no.
Quindi sono problemi per i quali può essere ragionevole aspettarsi l'appartenenza a P.
Tra questi c'è anche Factor.

P, NP e coNP : i possibili scenari