Problemi decisionali

- La teoria della complessità computazionale è definita principalmente in termini di problemi di decisione
  - Essendo la risposta binaria, non ci si deve preoccupare del tempo richiesto per restituire la soluzione e tutto il tempo è speso esclusivamente per il calcolo

Esempi

- Un esempio interessante, utilizzato ampiamente nella teoria della complessità:
  - *Soddisfacibilità di formule booleane*

Definizioni

- Insieme V di variabili Booleane
  - Letterale: variabile o sua negazione
  - Clausola: disgiunzione (OR) di letterali
- Un'espressione Booleana su V si dice in forma normale congiuntiva (FNC) se è espressa come congiunzione di clausole (AND di OR)

Esempi

$$V = \{x, y, z, w\}$$

$$FNC : (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee w) \wedge y$$

SAT

- Data una espressione in *forma normale congiuntiva*, il problema della soddisfacibilità (SAT) richiede di verificare se esiste una assegnazione di valori di verità alle variabili che rende l'espressione vera

Esempi

- La formula

$$(x \vee \overline{y} \vee z) \wedge (\overline{x} \vee w) \wedge y$$

è soddisfatta dall'assegnazione

$$x = 1 \quad y = 1 \quad z = 0 \quad w = 1$$

Certificato

- Certificato per SAT?
  - Un'assegnazione di verità alle variabili che renda vera l'espressione

$$SAT \in NP$$

Problemi NP-completi

- Caratterizzano i problemi più difficili all'interno della classe NP
  - Se esistesse un algoritmo polinomiale per risolvere uno solo di questi problemi, allora tutti i problemi in NP potrebbero essere risolti in tempo polinomiale, e P = NP
  - Quindi: o tutti i problemi NP-completi sono risolvibili deterministicamente in tempo polinomiale o nessuno di essi lo è

Riduzioni polinomiali

- Dati due problemi decisionali, Π_1 e Π_2.
- Siano I_1 e I_2 gli insiemi delle istanze di input di Π_1 e Π_2

Π_1 *si riduce in tempo polinomiale a* Π_2

$$\Pi_1 \leq_p \Pi_2$$

se esiste una *funzione* f: $I_1 \rightarrow I_2$ t.c.

- f è *calcolabile in tempo polinomiale*
- per ogni istanza x di Π_1 e ogni soluzione s $\in$ {0,1}

$$\Pi_1(x) = s \Leftrightarrow \Pi_2((f(x)) = s$$

Riduzioni polinomiali

In altri termini, f ***trasforma*** un' istanza di Π_1 in un' istanza di Π_2 in modo tale che

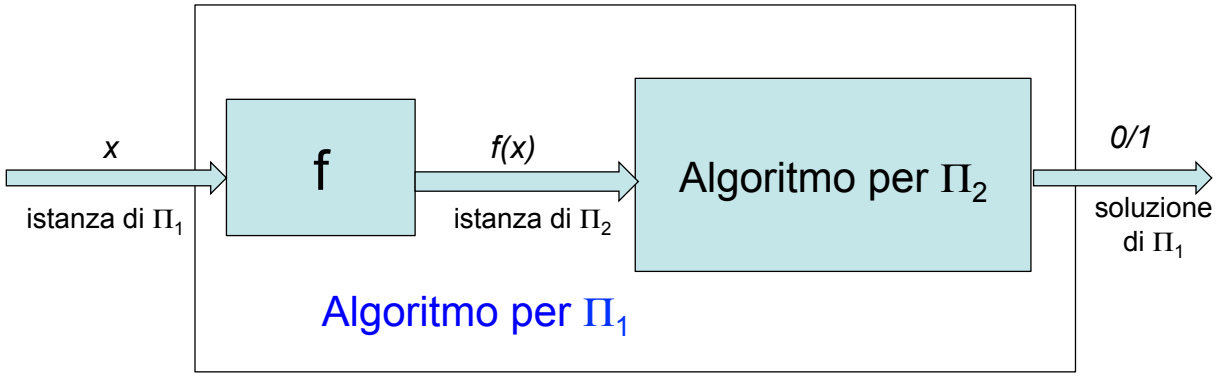
- istanze positive di Π_1 risultino in istanze positive di Π_2
- istanze negative di Π_1 risultino in istanze negative di Π_2

Riduzioni polinomiali

- Intuitivamente, un problema di decisione Π_1 si riduce polinomialmente a un altro problema di decisione Π_2 se
  - un algoritmo di risuoluzione polinomiale per Π_2 implica l'esistenza di un tale algoritmo anche per Π_1.

Riduzioni polinomiali

Se esistesse un algoritmo per risolvere Π_2, potremmo utilizzarlo per risolvere Π_1:



Riduzioni polinomiali

$$\Pi_1 \leq_p \Pi_2 \quad \text{e} \quad \Pi_2 \in P \Rightarrow \Pi_1 \in P$$

Problemi NP ardui

Un problema decisionale Π si dice NP-arduo se

per ogni $\Pi' \in$ NP, $\Pi' \leq_p \Pi$

Problemi NP completi

- Un problema decisionale Π si dice NP-completo se
  - $\Pi \in$ NP
  - Π è NP-arduo
- I problemi NPC sono i più "difficili" in NP
- Se si scoprisse un algoritmo polinomiale per risolvere un problema NPC, allora P = NP.

Problemi NP completi

- Dimostrare che un problema è in NP può essere facile
  - Esibire un certificato polinomiale
- Non è altrettanto facile dimostrare che un problema Π è NP-arduo
  - Bisogna dimostrare che TUTTI i problemi in NP si riducono polinomialmente a Π!
  - In realtà la prima dimostrazione di NP-completezza aggira il problema

Teorema di Cook

SAT

problema della soddisfacibilità di una espressione booleane in forma normale congiuntiva (FNC):

Teorema

SAT è NP completo

Teorema di Cook (idea)

- *Cook ha mostrato un algoritmo che*

  dati un qualunque problema Π ed una qualunque istanza x per Π,

  costruisce una espressione Booleana in forma normale congiuntiva che descrive il calcolo di un algoritmo per risolvere Π su x

- *L'espressione è vera se e solo se l'algoritmo restituisce 1*

Dimostrazioni di NP-completezza

- Sfruttano la transitività delle riduzione polinomiale

Se $\Pi_1 \leq_p \Pi_2$ e $\Pi_2 \leq_p \Pi_3$, allora $\Pi_1 \leq_p \Pi_3$

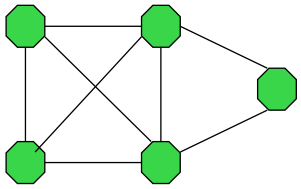
Problemi NP completi

- *Un problema decisionale Π è NP completo se*
  - $\Pi \in$ NP
  - SAT $\leq_p \Pi$

    (o un qualsiasi altro problema NPC)

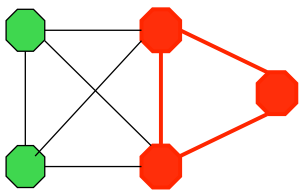
CLIQUE

- Dato un grafo G = (V,E) e un intero k > 0, stabilire se G contiene un sottografo completo di k nodi



CLIQUE

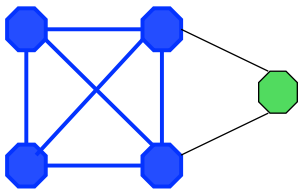
- Dato un grafo G = (V,E) e un intero k > 0, stabilire se G contiene un sottografo completo di k nodi



Clique di 3 nodi

CLIQUE

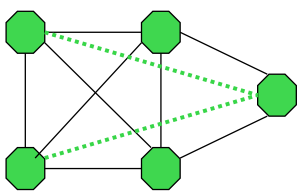
- Dato un grafo G = (V,E) e un intero k > 0, stabilire se G contiene un sottografo completo di k nodi



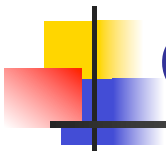
Clique di 4 nodi

CLIQUE

- Dato un grafo G = (V,E) e un intero k > 0, stabilire se G contiene un sottografo completo di k nodi



Non contiene
clique di 5 nodi



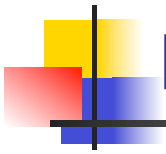
CLIQUE è NP completo

$SAT \leq_p CLIQUE$

data una espressione booleana F in forma normale congiuntiva con k clausole

costruire in tempo polinomiale

un grafo G che contiene una **clique di k vertici se e solo se F è soddisfacibile**.



Riduzione: vertici

- Ad ogni letterale in ciascuna clausola di F corrisponde un vertice in G.

Esempio:

$F = (a \vee b) \wedge (!a \vee !b \vee c) \wedge !c$

$G = (V, E)$,

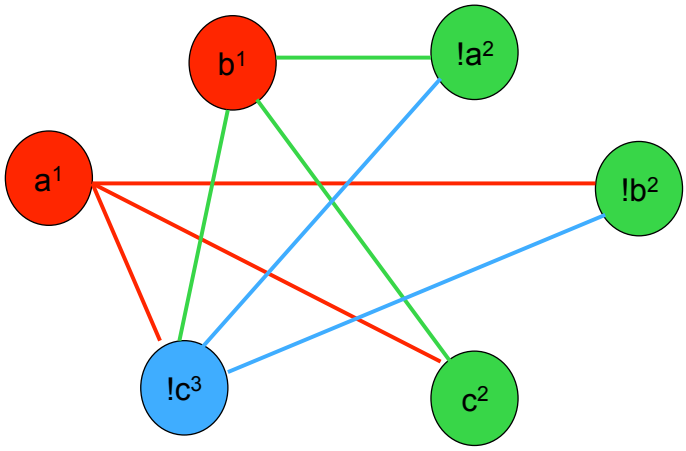
$V = \{ a^1, b^1, !a^2, !b^2, c^2, !c^3 \}$

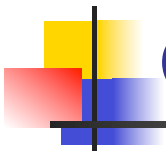
Riduzione: archi

$$(x^i, y^j) \in E \iff i \neq j \quad e \quad x \neq !y$$

Due letterali sono adiacenti in G se e solo se appartengono a clausole diverse e possono essere veri contemporaneamente.

F = (a ∨ b) ∧ (!a ∨ !b ∨ c) ∧ !c





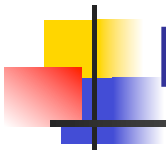
Clique in G

- *composta da k nodi*

  uno per ogni clausola di F

- *non può contenere due nodi della stessa clausola*

  perché non sono adiacenti.

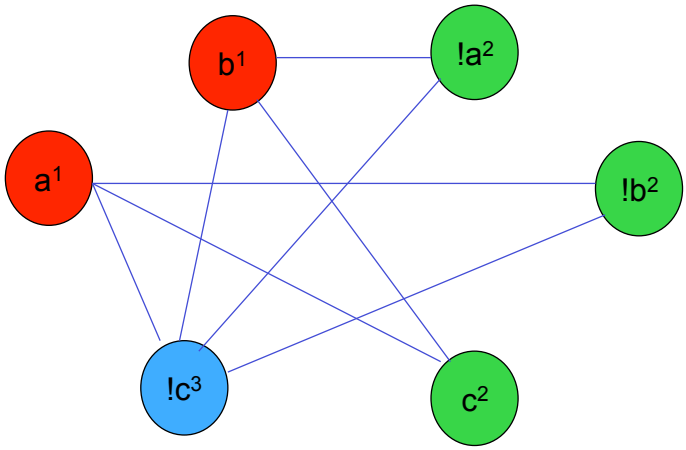


Riduzione

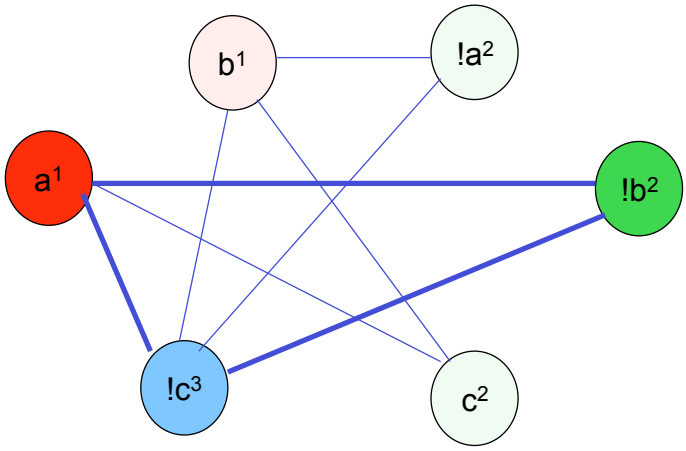
G contiene una clique ⇒ F è soddisfacibile

- si dà valore **1** (true) ai k letterali che corrispondono ai nodi della clique
- tutte la clausole corrispondenti diventano di valore **1** (true)
- F = **1** (true), soddisfacibile.

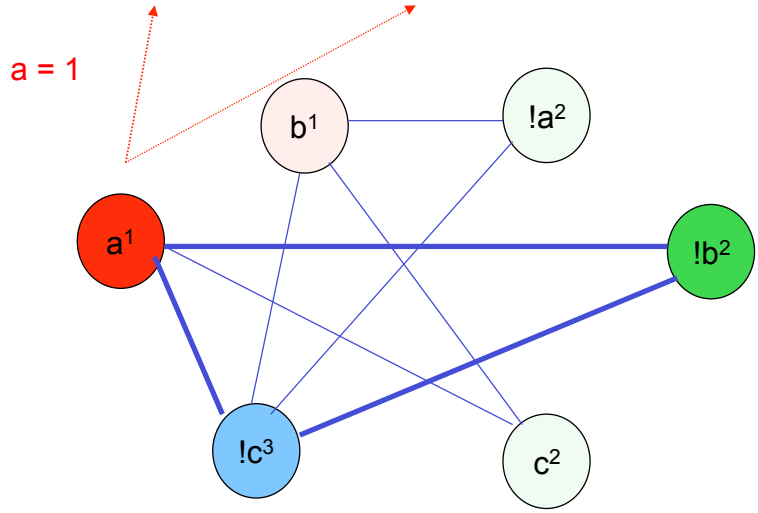
F = (a ∨ b) ∧ (!a ∨ !b ∨ c) ∧ !c



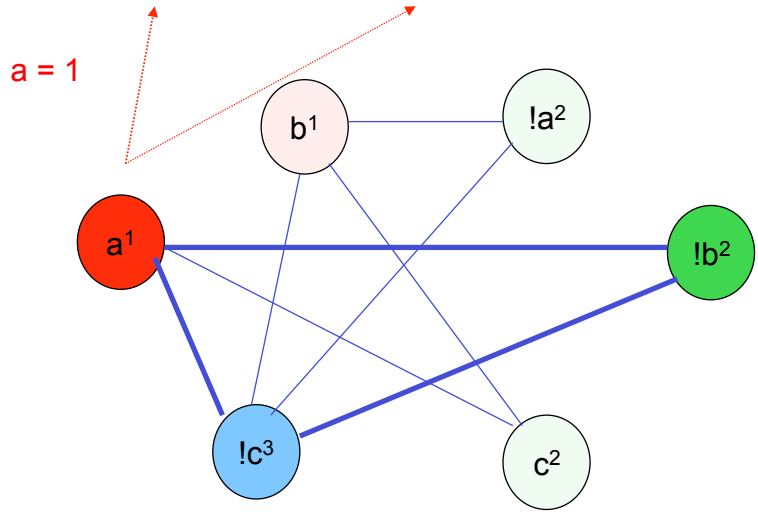
F = (a ∨ b) ∧ (!a ∨ !b ∨ c) ∧ !c



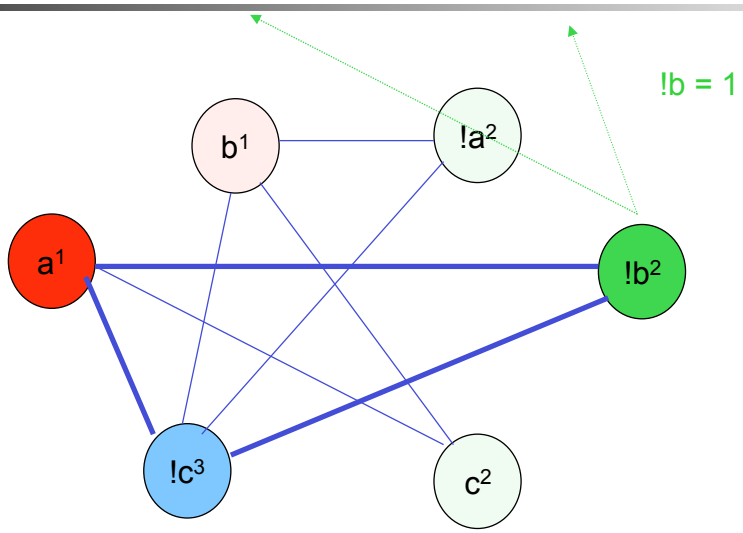
F = (a ∨ b) ∧ (!a ∨ !b ∨ c) ∧ !c



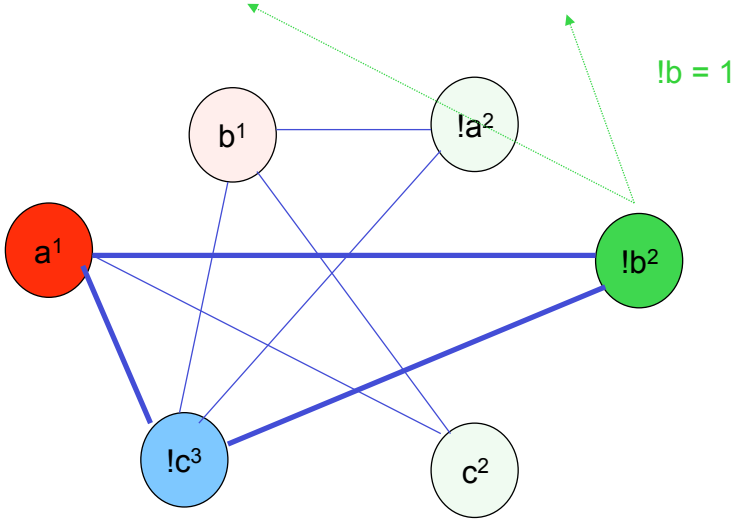
F = (1 ∨ b) ∧ (0 ∨ !b ∨ c) ∧ !c



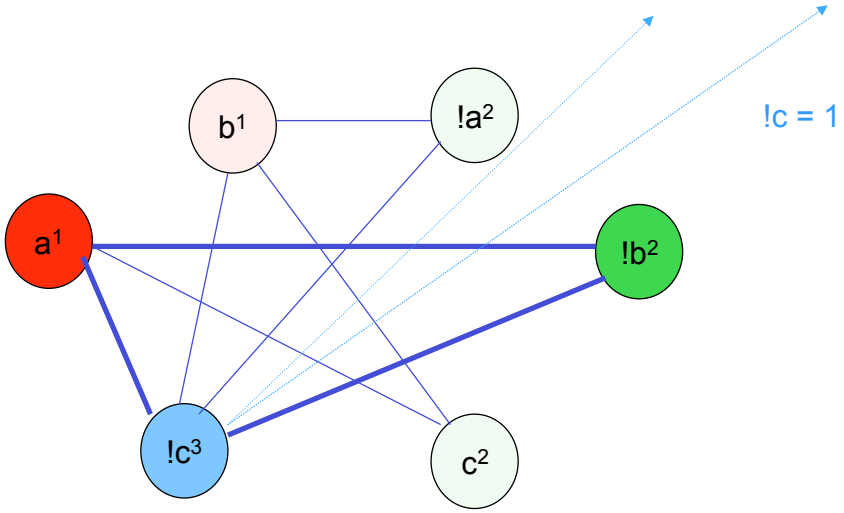
F = (1 ∨ b) ∧ (0 ∨ !b ∨ c) ∧ !c



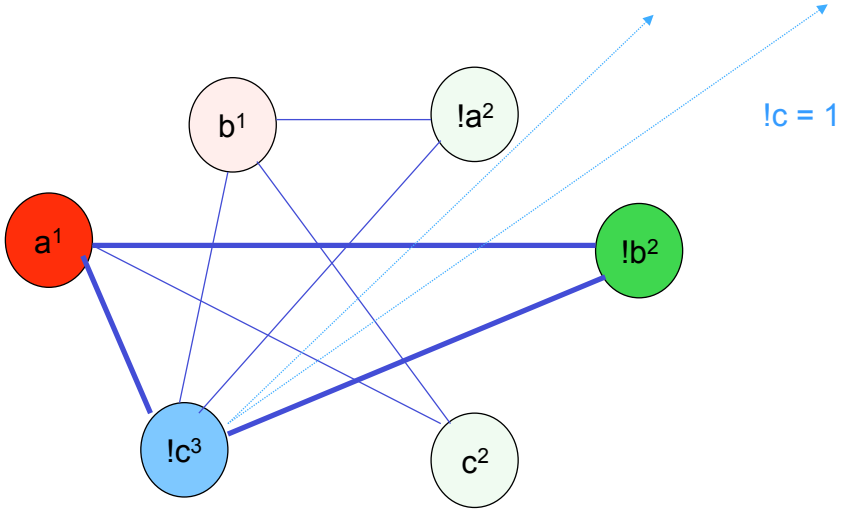
F = (1 ∨ 0) ∧ (0 ∨ 1 ∨ c) ∧ !c



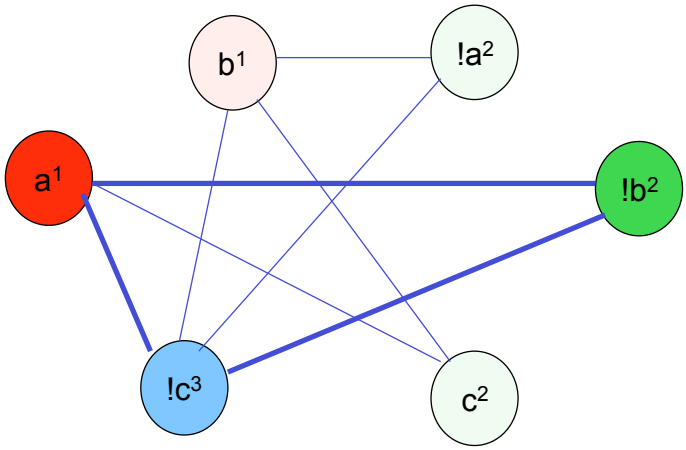
F = (1 ∨ 0) ∧ (0 ∨ 1 ∨ c) ∧ !c



F = (1 ∨ 0) ∧ (0 ∨ 1 ∨ 0) ∧ 1



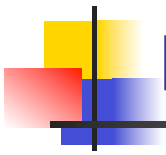
F = (1) ∧ (1) ∧ 1 = 1



Riduzione

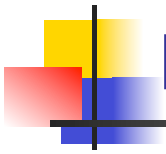
F è soddisfacibile ⇒ G contiene una clique

- esiste almeno un letterale vero per ogni clausola
- i corrispondenti vertici in G formano una clique.



Riduzione

- *La riduzione da F a G = (V,E) si esegue in tempo polinomiale:*
  - *n* = # variabili
  - *k* = # clausole
  - $|V| \leq n\,k$
  - l'esistenza di un arco si stabilisce in tempo costante
  - $|E| \leq O((n\,k)^2)$



Problemi NP equivalenti

- $SAT \leq_p CLIQUE$ $\Rightarrow$ CLIQUE è NP completo
- SAT è NP completo $\Rightarrow$ $CLIQUE \leq_p SAT$
- *SAT e CLIQUE sono* ***NP equivalenti***.
- *Tutti i problemi NP completi sono tra loro* ***NP equivalenti***.

Gerarchia delle classi aggiornata