

# Algoritmi e Strutture Dati

## Problemi intrattabili Teoria dell'NP-completezza

Alberto Montresor

Università di Trento

2018/11/07

This work is licensed under a Creative Commons  
Attribution-ShareAlike 4.0 International License.



# Sommario

- 1 Introduzione
- 2 Riduzioni
- 3 Classi  $P$ ,  $PSPACE$
- 4 Classe  $NP$
- 5 Problemi  $NP$ -Completi

# Introduzione

Con l'eccezione della sezione su backtrack, finora abbiamo considerato solo problemi con soluzioni in tempo polinomiale

- Il tempo di esecuzione è  $O(n^k)$  per qualche  $k$ ;

Sono gli unici problemi interessanti?

- Esistono problemi che per essere risolti hanno bisogno di un tempo almeno esponenziale (**EXPTIME**)
  - Valutare una posizione di scacchi, dama, go;
  - Risolvere il problema delle Torri di Hanoi
- Esistono problemi per i quali non esiste alcuna soluzione
  - Halting problem

# Introduzione

## Obiettivo della lezione

- Discuteremo di un classe di problemi per cui non è chiaro se esiste un algoritmo polinomiale oppure no
- Vedremo che questi problemi sono tutti sulla stessa barca: possono essere tutti risolti in tempo polinomiale, oppure nessuno
- Questa classe contiene un numero enorme di problemi interessanti!

## Millenium Prize Problems, Clay Institute

- Premio istituito nel 2000
- Sette problemi matematici aperti
- 1M€ per chi li risolve
- Ad oggi, un solo problema risolto (Congettura di Poincaré)

# Alcune definizioni

## Problema astratto

Una **problema astratto** è una relazione binaria  $R \subseteq I \times S$  tra un insieme  $I$  di istanze del problema e un insieme  $S$  di soluzioni

## Esempio

Per il problema SHORTEST-PATH:

- Un'istanza è una quadrupla  $(V, E, u, v)$
- Una soluzione è una sequenza ordinata di vertici
- Nota: possono esistere più soluzioni associate alla stessa istanza

# Tipologia di problemi

## Problemi di **ottimizzazione**

- Data un'istanza, trovare la "migliore" soluzione secondo criteri specifici
- **SHORTEST-PATH (Opt.)**: Dati un grafo  $G$  e due nodi  $u, v$ , trovare il cammino più breve fra essi

## Problemi di **ricerca**

- Data un'istanza, trovare una possibile soluzione fra quelle esistenti
- **PATH**: Dati un grafo  $G$  e due nodi  $u, v$ , trovare un cammino fra essi

## Problemi di **decisione**

- Data un'istanza, verificare se soddisfa o meno una data proprietà  
La relazione  $R$  è una funzione  $R : I \rightarrow \{0, 1\}$
- **SHORTEST-PATH (Dec.)**: Dati un grafo  $G$ , due nodi  $u, v$  e un valore  $k$ , esiste un cammino tra  $u$  e  $v$  di lunghezza maggiore o uguale a  $k$ ?

# Equivalenza problemi di ottimizzazione / decisione

I concetti che vedremo sono discussi in termini di problemi di decisione

- La versione decisionale è più semplice da trattare matematicamente
- Se posso risolvere efficientemente la versione di ottimizzazione  $\Rightarrow$  posso risolvere efficientemente la versione di decisione
- Se *non* posso risolvere efficientemente la versione di decisione  $\Rightarrow$  *non* posso risolvere efficientemente la versione di ottimizzazione

Esempio – Shortest path

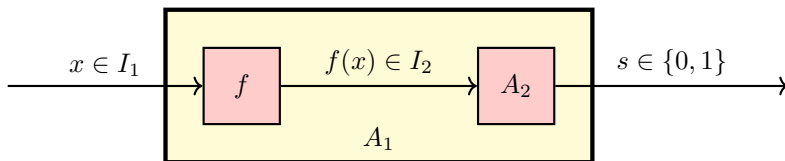
- Se conosco il cammino più breve, posso rispondere alla domanda decisionale per qualunque valore di  $k$

# Riduzione polinomiale

## Riduzione polinomiale

Dati due problemi decisionali  $R_1 \subseteq I_1 \times \{0, 1\}$  e  $R_2 \subseteq I_2 \times \{0, 1\}$ ,  $R_1$  è **riducibile polinomialmente** a  $R_2$  ( $R_1 \leq_p R_2$ ) se esiste una funzione  $f : I_1 \rightarrow I_2$  con le seguenti proprietà:

- $f$  è calcolabile in tempo polinomiale;
- per ogni istanza  $x$  del problema  $R_1$  e ogni soluzione  $s \in \{0, 1\}$ ,  $(x, s) \in R_1$  se e solo se  $(f(x), s) \in R_2$ .



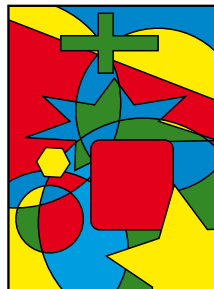
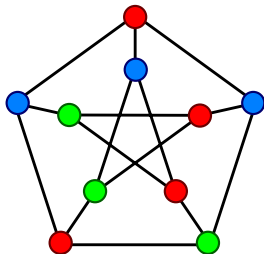


# Colorazione di grafi

## Colorazione di grafi (GRAPH-COLORING)

Dato un grafo non orientato  $G = (V, E)$  e un insieme di colori  $C$  di dimensione  $k$ , una **colorazione** dei vertici è un assegnamento  $f : V \rightarrow C$

- che “colora” ogni nodo con uno dei valori in  $C$ ,
- tale per cui nessuna coppia di nodi adiacenti ha lo stesso colore.



# Colorazione di grafi

## Colorazione, ottimizzazione

Dato un grafo non orientato  $G = (V, E)$ , restituire la colorazione che necessita del numero minimo di colori

## Colorazione, decisionale

Dato un grafo non orientato  $G = (V, E)$  e un valore  $k$ , determinare se esiste una colorazione di  $G$  con  $k$  colori

# Sudoku

## SUDOKU

Il problema generale del **Sudoku** richiede di inserire dei numeri fra 1 e  $n^2$  in una matrice di  $n^2 \times n^2$  elementi suddivisa in  $n \times n$  sottomatrici di dimensione  $n \times n$ , in modo tale che nessun numero compaia più di una volta in ogni riga, colonna e sottomatrice

## SUDOKU, decisionale

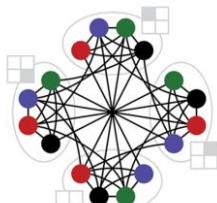
Data una matrice  $n^2 \times n^2$  elementi, determinare se esiste un modo per assegnare i numeri in modo da rispettare le regole del Sudoku

# Riduzione da problema particolare a problema generale

## Riduzione in tempo polinomiale

- $V = \{(x, y) : 1 \leq x \leq n^2, 1 \leq y \leq n^2\}$
- $[(x, y), (x', y')] \in E$  se e solo se:
  - $x = x'$ ; oppure,
  - $y = y'$ ; oppure,
  - $(\lceil x/n \rceil = \lceil x'/n \rceil) \wedge (\lceil y/n \rceil = \lceil y'/n \rceil)$
- L'insieme di colori è  $\{1, \dots, n\}$ .

1	2	3	4
3	4	1	2
2	3	4	1
4	1	2	3



SUDOKU

 $\leq_p$ 

GRAPH-COLORING

Se abbiamo una soluzione per la colorazione, allora abbiamo una soluzione algoritmica per il Sudoku

## Applicazioni

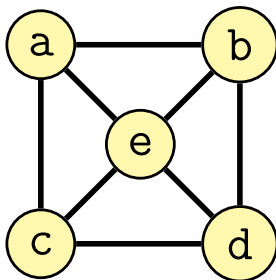
- Assegnamento radio frequenze in un insieme di torri cellulari
- Allocazione degli esami universitari

# Insieme indipendente

## Insieme Indipendente (INDEPENDENT-SET)

Sia dato un grafo non orientato  $G = (V, E)$ ; un insieme  $S \subseteq V$  è un **insieme indipendente** se solo se nessun arco unisce due nodi in  $S$

$$\forall (x, y) \in E : x \notin S \vee y \notin S$$

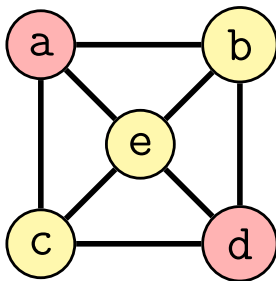


# Insieme indipendente

## Insieme Indipendente (INDEPENDENT-SET)

Sia dato un grafo non orientato  $G = (V, E)$ ; un insieme  $S \subseteq V$  è un **insieme indipendente** se solo se nessun arco unisce due nodi in  $S$

$$\forall (x, y) \in E : x \notin S \vee y \notin S$$



# Insieme indipendente

## INDEPENDENT-SET, ottimizzazione

Dato un grafo non orientato  $G = (V, E)$ , restituire il più grande insieme indipendente presente nel grafo

## INDEPENDENT-SET, decisionale

Dato un grafo non orientato  $G = (V, E)$  e un valore  $k$ , determinare se esiste un insieme indipendente di dimensione almeno  $k$

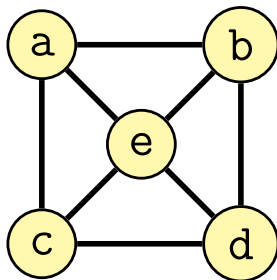
Questo è un esempio di **packing problem**, un problema in cui si cerca di selezionare il maggior numero di oggetti, la cui scelta è però soggetta a vincoli di esclusione

# Copertura di vertici

## Copertura di vertici (VERTEX-COVER)

Dato un grafo non orientato  $G = (V, E)$ , un insieme  $S \subseteq V$  è un **copertura di vertici** se e solo se ogni arco ha almeno un estremo in  $S$

$$\forall (x, y) \in E : x \in S \vee y \in S$$



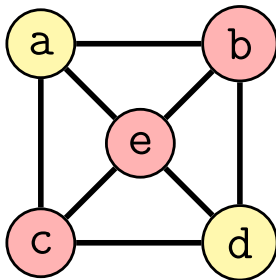


# Copertura di vertici

## Copertura di vertici (VERTEX-COVER)

Dato un grafo non orientato  $G = (V, E)$ , un insieme  $S \subseteq V$  è un **copertura di vertici** se e solo se ogni arco ha almeno un estremo in  $S$

$$\forall (x, y) \in E : x \in S \vee y \in S$$



## Copertura di vertici

VERTEX-COVER, ottimizzazione

Dato un grafo non orientato  $G = (V, E)$ , restituire la copertura dei vertici di dimensione minima.

VERTEX-COVER, decisionale

Dato un grafo non orientato  $G = (V, E)$  e un valore  $k$ , determinare se esiste una copertura di vertici di dimensione al massimo  $k$

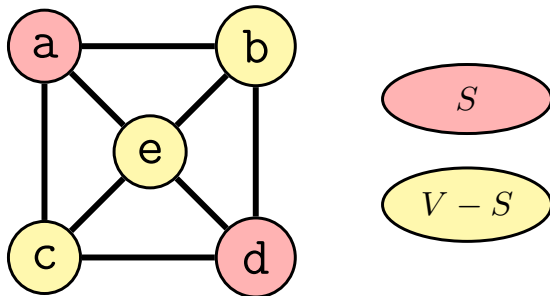
Questo è un esempio di **covering problem**, un problema in cui si cerca di ottenere il più piccolo insieme in grado di coprire un insieme arbitrario di oggetti con il più piccolo sottoinsieme di questi oggetti.

# Riduzione per problemi duali

$S \subseteq V$  è un independent set  $\Rightarrow V - S$  è un vertex cover

Se  $S$  è un insieme indipendente:

- ogni arco  $(x, y)$  non può avere entrambi gli estremi in  $S$ ;
- quindi almeno uno dei due deve essere in  $V - S$ , CVD

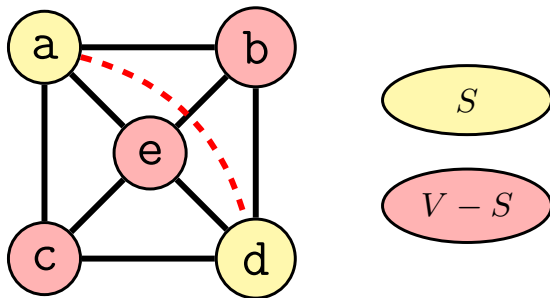


# Riduzione per problemi duali

$V - S$  è un vertex cover  $\Rightarrow S \subseteq V$  è un independent set

Supponiamo per assurdo che  $S$  non sia un independent set:

- allora esiste un arco  $(x, y)$  che unisce due nodi in  $S$
- nessuno dei due estremi sta in  $V - S$
- questo implica che  $V - S$  non è un vertex cover, assurdo



# Riduzione per problemi duali

$\text{VERTEX-COVER} \leq_p \text{INDEPENDENT-SET}$

$\text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER}$

Abbiamo quindi dimostrato che i due problemi sono equivalenti.

# SAT

## Formule booleane in forma normale congiuntiva

Dato un insieme  $V$  contenente  $n$  **variabili**

- un **letterale** è una variabile  $v$  oppure la sua negazione  $\bar{v}$
- una **clausola** è una disgiunzione di letterali (letterali separati da **or**)
- una **formula in forma normale congiuntiva** è una congiunzione di clausole (clausole separate da **and**)

Esempio

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

# SAT

## Soddisfattibilità di formule booleane ((SATISFIABILITY))

Data un'espressione in forma normale congiuntiva, il problema della **soddisfacibilità** consiste nel decidere se esiste una assegnazione di valori di verità alle variabili che rende l'espressione vera.

### Esempio

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

$$x = \mathbf{true}, y = \mathbf{true}, w = \mathbf{true}, z = \mathbf{true}\}$$

$$(\mathbf{true} \vee \mathbf{false} \vee \mathbf{true}) \wedge (\mathbf{false} \vee \mathbf{true}) \wedge \mathbf{true}$$

# 3-SAT

## 3-SAT

Data una espressione in forma normale congiuntiva in cui le clausole **hanno esattamente 3 letterali**, il problema della soddisfacibilità consiste nel decidere se esiste una assegnazione di valori di verità alle variabili che rende l'espressione vera.

Vogliamo dimostrare che:

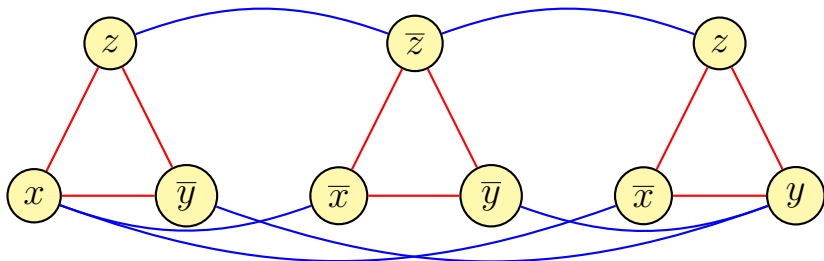
$$3\text{-SAT} \leq_p \text{INDEPENDENT-SET}$$



# Riduzione tramite "gadget"

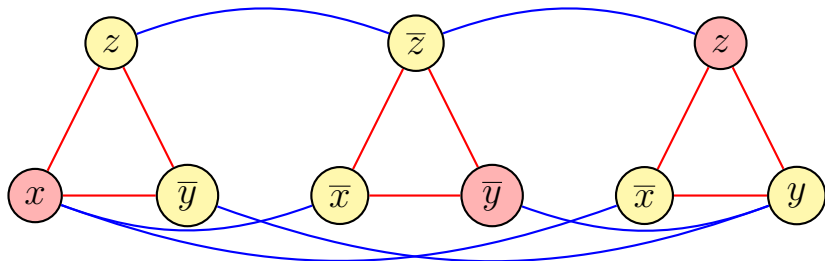
Data una formula 3-SAT, costruiamo un grafo nel modo seguente:

- Per ogni clausola, aggiungiamo un terzetto di nodi, collegati fra di loro da archi
- Per ogni letterale che compare in modo normale e in modo negato, aggiungere un arco fra di essi (arco di conflitto).



## Riduzione tramite "gadget"

La formula 3-SAT è soddisfacibile se e solo è possibile trovare un independent set di dimensione esattamente  $k$ .



[https://en.wikipedia.org/wiki/Gadget\\_\(computer\\_science\)](https://en.wikipedia.org/wiki/Gadget_(computer_science))

# Riduzione da problema generale a problema particolare

Ovviamente:  $3\text{-SAT} \leq_p \text{SAT}$

E' possibile dimostrare che  $\text{SAT} \leq_p 3\text{-SAT}$ ?

E' possibile trasformare una formula SAT in una formula 3-SAT usando due semplici trucchi:

- Se la clausola è più lunga di tre elementi, si introduce una nuova variabile e si divide la clausola in due:

$$(a \vee b \vee c \vee d) \equiv (a \vee b \vee z) \wedge (\bar{z} \vee c \vee d)$$

- Se la clausola è più corta di tre elementi, si fa "padding"

$$(a \vee b) \equiv (a \vee a \vee b)$$

# Transitività delle riduzioni

E' facile intuire che la nozione di riducibilità polinomiale gode della proprietà transitiva

$$\text{SAT} \leq_p \text{3-SAT} \leq_p \text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER}$$

# Classi P, PSPACE

## Algoritmo

Dati un problema di decisione  $R$  e un algoritmo  $A$  (scritto in un modello di calcolo Turing-equivalente) che lavora in tempo  $f_t(n)$  e spazio  $f_s(n)$ , diciamo che  $A$  risolve  $R$  se  $A$  restituisce  $s$  su un'istanza  $x$  se e solo se  $(x, s) \in R$ .

## Classi di complessità

Data una qualunque funzione  $f(n)$ , chiamiamo:

- **TIME**( $f(n)$ ) l'insieme dei problemi decisionali risolvibili da un algoritmo che lavora in tempo  $O(f(n))$
- **SPACE**( $f(n)$ ) gli insiemi dei problemi decisionali risolvibili da un algoritmo che lavora in spazio  $O(f(n))$ .

# Classi P, PSPACE

## Classe P

La **classe P** è la classe dei problemi decisionali risolvibili in tempo polinomiale nella dimensione  $n$  dell'istanza di ingresso:

$$P = \cup_{c=0}^{\infty} \text{TIME}(n^c)$$

## Classe PSPACE

La **classe PSPACE** è la classe dei problemi decisionali risolvibili in spazio polinomiale nella dimensione  $n$  dell'istanza di ingresso:

$$\text{PSPACE} = \cup_{c=0}^{\infty} \text{SPACE}(n^c)$$

### Note

- $P \subseteq \text{PSPACE}$

# Classe NP

## Certificato

Dato un problema decisionale  $R$  e un'istanza di input  $x$  tale che  $(x, \mathbf{true}) \in R$ , un **certificato** è un insieme di informazioni che permette di provare che  $(x, \mathbf{true}) \in R$

### Esempi:

- SAT: un assegnamento di verità alle variabili della formula
- GRAPH-COLORING: un'associazione nodo-colore  $f : V \rightarrow \{1, \dots, k\}$
- INDEPENDENT-SET: un sottoinsieme di  $V$  di  $k$  elementi

Tutti questi “certificati” hanno dimensione polinomiale nella dimensione dell'input.

## Classe NP – Definizione "moderna"

I certificati suddetti possono essere verificati in tempo polinomiale:

- SAT: si calcola il valore di verità della formula a partire dall'assegnamento di verità delle variabili in tempo  $O(n)$
- GRAPH-COLORING: si verifica che nodi adiacenti non abbiano lo stesso colore in  $O(m + n)$
- INDEPENDENT-SET: si verifica che nodi in  $V$  non abbiano nodi adiacenti in  $V$  in tempo  $O(m + n)$

### Classe NP

L'insieme di tutti i problemi che ammettono un certificato verificabile in tempo polinomiale.



# Certificati non polinomiali

## Quantified Boolean Formula (QBF)

Il problema QBF è una generalizzazione del problema SAT nel quale ad ogni variabile possono essere applicati quantificatori universali e esistenziali

### Esempio

$$\forall x \exists y \exists z ((x \vee z) \wedge y)$$

### Esercizio...

Progettate un certificato per QCB che possa essere verificato in tempo polinomiale

Si ritiene che un certificato del genere non esista

# Classe NP – Definizione basata su non determinismo

## Classe NP

NP è l'insieme di problemi decisionali che possono essere risolti da una Macchina di Turing non deterministica in tempo polinomiale

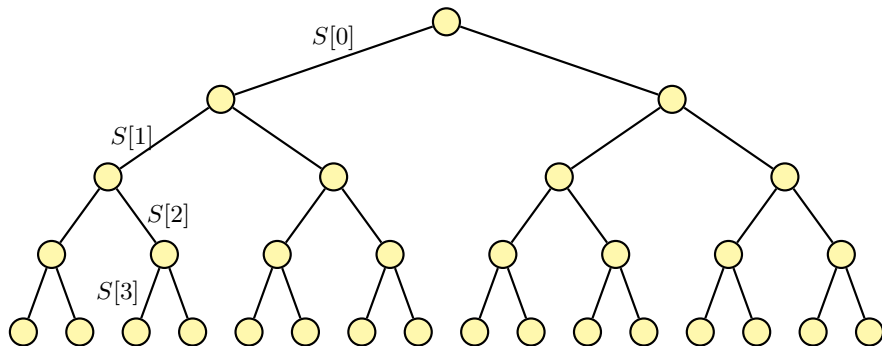
In maniera mooolto informale...

- Dato uno stato ed un elemento di input, una macchina non deterministica può andare in un insieme finito di altri stati
- Due interpretazioni per la macchina non deterministica:
  - Macchina non deterministica che "azzecca" sempre la scelta giusta
  - Macchina non deterministica che si divide in un insieme finito di copie, una per scelta possibile

# Classe NP – Definizione basata su non determinismo

## Esempio

SAT con quattro variabili



# Relazioni fra problemi

## Lemma

*Se  $R_1 \leq_p R_2$  e  $R_2 \in \mathbb{P}$ , allora anche  $R_1$  è contenuto in  $\mathbb{P}$ .*

### Dimostrazione

- Sia  $T_f(n) = O(n^{k_f})$  il tempo necessario per trasformare un input di  $R_1$  in input di  $R_2$
- Sia  $T_2(n) = O(n^{k_2})$  il tempo necessario per risolvere  $R_2$
- Qual è la complessità per risolvere  $R_1$ ?
- La funzione di trasformazione può prendere un input di dimensione  $n$  e trasformarlo in un input di dimensione  $O(n^{k_f})$  per  $R_2$
- Il tempo per risolvere  $R_1$  sarà quindi  $T_1(n) = O(n^{k_f k_2})$
- $T_1(n)$  è polinomiale

# Definizioni

## Problema NP-arduo (NP-hard)

Un problema decisionale  $R$  si dice **NP-arduo** se ogni problema  $Q \in \text{NP}$  è riducibile polinomialmente a  $R$  ( $Q \leq_p R$ )

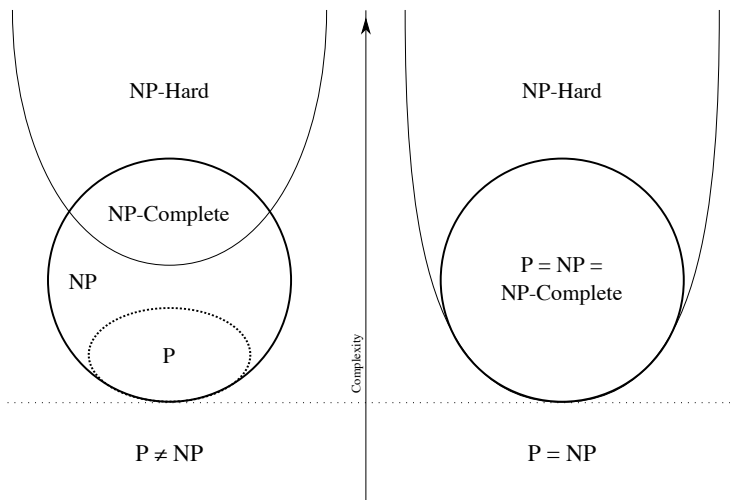
## Problema NP-completo (NP-complete)

Un problema decisionale  $R$  si dice **NP-completo** se appartiene alla classe  $\text{NP}$  ed è NP-hard.

## $\mathbb{P} = \text{NP}$ ?

Se un qualunque problema decisionale NP-completo appartenesse a  $\mathbb{P}$ , allora risulterebbe  $\mathbb{P} = \text{NP}$ .

## P vs NP



[https://en.wikipedia.org/wiki/NP\\_\(complexity\)#/media/File:P\\_np\\_np-complete\\_np-hard.svg](https://en.wikipedia.org/wiki/NP_(complexity)#/media/File:P_np_np-complete_np-hard.svg)

# Problemi NP-completi: esistono?

## Problema

- Dimostrare che un problema è contenuto in  $\text{NP}$  è semplice
- Dimostrare che un problema è  $\text{NP}$ -completo richiede una dimostrazione difficile, apparantemente impossibile:

*Tutti i problemi in  $\text{NP}$  sono riducibili polinomialmente a tale problema - anche quelli che non conosciamo!*

Fra i problemi visti oggi, quale potrebbe essere un buon candidato?

# Teorema di Cook-Levin

Stephen Cook (1971). "The Complexity of Theorem Proving Procedures". Proceedings of the third annual ACM Symposium on Theory of computing (STOC). pp. 151–158. ACM.

## Teorema di Cook-Levin

SAT è NP-completo

- Dimostrazione complessa, basata sugli stati della macchina di Turing
- Dimostrato nel 1973 da Leonid Levin (URSS) in maniera indipendente



[https://it.wikipedia.org/wiki/Stephen\\_Cook#/media/File:Prof.Cook.jpg](https://it.wikipedia.org/wiki/Stephen_Cook#/media/File:Prof.Cook.jpg)



## Problemi introdotti oggi

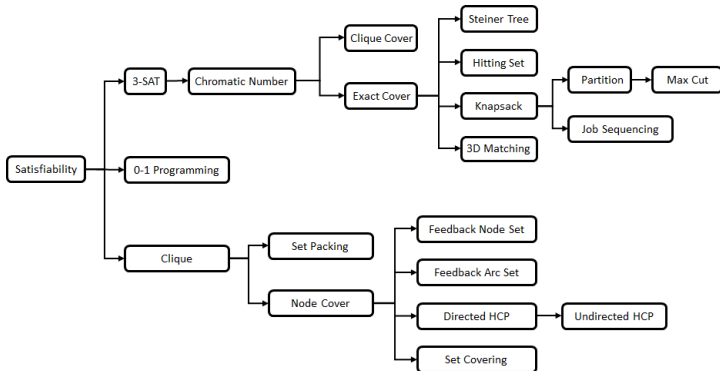
Partendo dalle riduzioni viste oggi e utilizzando il Teorema di Cook-Levin, otteniamo:

$$\text{SAT} \leq_p \text{3-SAT} \leq_p \text{INDEPENDENT-SET} \leq_p \text{VERTEX-COVER} \leq_p \text{SAT}$$

In altre parole, 3-SAT, INDEPENDENT-SET, VERTEX-COVER sono NP-completi.

# I 21 problemi NP-completi di Karp

Richard Karp (1972). "Reducibility Among Combinatorial Problems". In R. Miller and J. Thatcher (eds). Proc. of Complexity of Computer Computations, pp. 85–103 (Plenum).



# NP Compendium

## A compendium of NP optimization problems

Editors:

[Pierluigi Crescenzi](#), and [Viggo Kann](#)

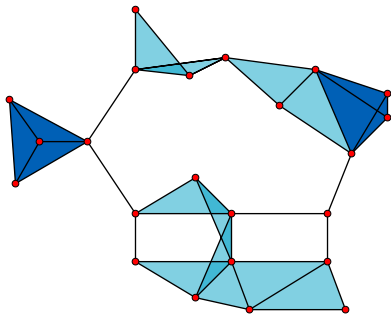
- 
- [Introduction](#)
  - [Graph Theory](#)
    - [Covering and Partitioning](#)
    - [Subgraphs and Supergraphs](#)
    - [Vertex Ordering](#)
    - [Iso- and Other Morphisms](#)
    - [Miscellaneous](#)
  - [Network Design](#)
    - [Spanning Trees](#)
    - [Cuts and Connectivity](#)
    - [Routing Problems](#)
    - [Flow Problems](#)
    - [Miscellaneous](#)

<https://www.nada.kth.se/~viggo/problemlist/compendium.html>

# Problemi NP-Completi "Classici"

## Cricca (Clique)

Dati un grafo non orientato ed un intero  $k$ , esiste un sottoinsieme di almeno  $k$  nodi tutti mutuamente adiacenti?



### Applicazioni

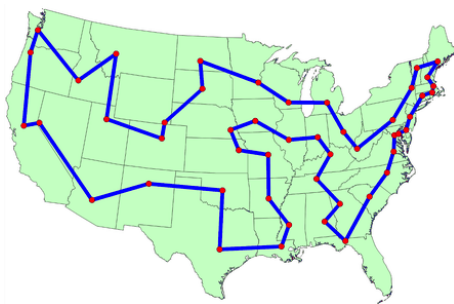
- Bioinformatica
- Ingegneria elettronica
- Chimica

[https://en.wikipedia.org/wiki/Clique\\_\(graph\\_theory\)#/media/File:VR\\_complex.svg](https://en.wikipedia.org/wiki/Clique_(graph_theory)#/media/File:VR_complex.svg)

# Problemi NP-Completi "Classici"

## Commesso viaggiatore (Traveling salesperson - TSP)

Date  $n$  città, le distanze tra esse, ed un intero  $k$ , è possibile partire da una città, attraversare ogni città esattamente una volta tornando alla città di partenza, percorrendo una distanza totale non superiore a  $k$ ?



<https://optimization.mccormick.northwestern.edu/images/e/ea/48StatesTSP.png>

# Problemi NP-Completi "Classici"

## Programmazione lineare 0/1

Data una matrice  $A$  di elementi interi e di dimensione  $m \times n$ , ed un vettore  $b$  di  $m$  elementi interi, esiste un vettore  $x$  di  $n$  elementi 0/1 tale che  $Ax \leq b$ ?

### Esempio

$$x_1 + x_2 + x_3 + x_4 \geq 2$$

$$x_1 - x_2 - x_3 + x_4 \geq 0$$

$$x_1 + x_3 + x_4 \leq 1$$

Il sistema è verificato per  $x_1 = x_2 = 1$  ed  $x_3 = x_4 = 0$ .

# Problemi NP-Completi "Classici"

## Copertura esatta di insiemi (Exact cover)

Dato un insieme  $X$  e una collezione di suoi sottoinsiemi

$\mathcal{Y} = \{Y_1, \dots, Y_n\}$ , esiste una sottocollezione  $\mathcal{Z}$  di  $\mathcal{Y}$  che partizioni  $X$ ?

$$X = \{1, 2, 3, 4, 5, 6, 7\}$$

$$\mathcal{Z} = \{B, D, F\}$$

$$\mathcal{Y} = \{A, B, C, D, E, F\}$$

$$A = \{1, 4, 7\}$$

$$B = \{1, 4\}$$

$$C = \{4, 5, 7\}$$

$$D = \{3, 5, 6\}$$

$$E = \{2, 3, 6, 7\}$$

$$F = \{2, 7\}$$

# Problemi NP-Completi "Classici"

## Partizione (PARTITION)

Dato un insieme  $A = \{a_1, \dots, a_n\}$  di interi positivi, esiste un sottoinsieme  $S$  di  $\{1, \dots, n\}$  tale che  $\sum_{i \in S} a_i = \sum_{i \notin S} a_i$ ?

## Somma di sottoinsieme (SUBSET-SUM)

Dati un insieme  $A = \{a_1, \dots, a_n\}$  di interi positivi ed un intero positivo  $k$ , esiste un sottoinsieme  $S$  di indici in  $\{1, \dots, n\}$  tale che  $\sum_{i \in S} a_i = k$ ?

## Zaino (KNAPSACK)

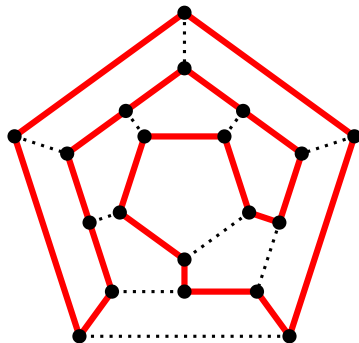
Dati un intero positivo  $C$  (la capacità dello zaino) e un insieme di  $n$  oggetti, tali che l'oggetto  $i$  è caratterizzato da un "profitto"  $p_i \in \mathbf{Z}^+$  e da un peso  $w_i \in \mathbf{Z}^+$ , esiste un sottoinsieme  $S \subseteq \{1, \dots, n\}$  tale che il peso totale  $w(S) = \sum_{i \in S} w_i \leq C$  e il profitto totale  $p(S) = \sum_{i \in S} p_i$  è maggiore o uguale a  $k$ ?



# Problemi NP-Completi "Classici"

## Circuito hamiltoniano (HAMILTONIAN-CIRCUIT)

Dato un grafo non orientato  $G$ , esiste un circuito che attraversi ogni nodo una e una sola volta?



[https://en.wikipedia.org/wiki/Hamiltonian\\_path#/media/File:Hamiltonian\\_path.svg](https://en.wikipedia.org/wiki/Hamiltonian_path#/media/File:Hamiltonian_path.svg)

# La complessità si nasconde dove non te l'aspetti...

## Circuito hamiltoniano

Dato un grafo non orientato  $G$ , esiste un circuito che attraversi ogni **nodo** una e una sola volta?

NP-completo

## Circuito euleriano

Dato un grafo non orientato  $G$ , esiste un circuito che attraversi ogni **arco** una e una sola volta?

Risolvibile in tempo polinomiale

## Cammini massimi

Dato un grafo  $G = (V, E)$  e una funzione di peso  $w$  sugli archi, trovare il cammino con peso **massimo**

NP-completo

## Cammini minimi

Dato un grafo  $G = (V, E)$  e una funzione di peso  $w$  sugli archi, trovare il cammino con peso **minimo**

Risolvibile in tempo polinomiale

# Problemi aperti

## Isomorfismo fra grafi

Il problema dell'isomorfismo fra grafi richiede di determinare se due grafi finiti sono isomorfi.

- Non sappiamo ancora.... NP-completo?  $\mathbb{P}$ ?
- Dibattito ancora in corso (Ottobre 2017)

## Primalità

Dato un numero  $n$ , determinare se  $n$  è primo.

- Incluso in  $\mathbb{P}$
- AKS primality test  
(2002-2005):  $\tilde{O}(\log^6 n)$

## Fattorizzazione

Dato un numero  $n$ , individuare i fattori primi che lo compongono.

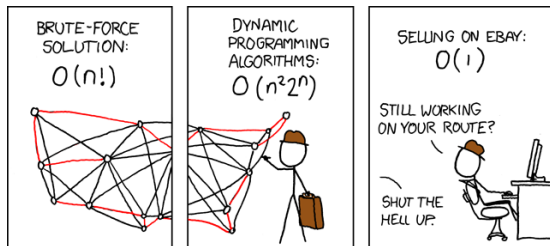
- Sicuramente in NP, si presume non in  $\mathbb{P}$ , non NP-completo

# Spunti di lettura

Sanjeev Arora, Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009

<http://theory.cs.princeton.edu/complexity/>

Lance Fortnow. The Golden Ticket. P, NP, and the Search for the Impossible. Princeton University Press, 2013.



<https://xkcd.com/399/>