

Capitolo 18

Problemi intrattabili

Teoria dell'NP-completezza

Introduzione

Con l'eccezione della sezione su backtrack, finora abbiamo considerato solo problemi con soluzioni in tempo polinomiale. Il tempo di esecuzione per queste soluzioni è $\mathcal{O}(n^k)$ per qualche k .

Esistono problemi che per essere risolti hanno bisogno di un tempo almeno esponenziale (che indicheremo con **EXPTIME**). Valutare una posizione di scacchi, dama, go; Oppure risolvere il problema delle Torri di Hanoi sono esempi di problemi con tempo almeno esponenziale. Esistono inoltre problemi per i quali non esiste alcuna soluzione. Come ad esempio il problema della terminazione (*halting problem*).

In questa lezione discuteremo di una classe di problemi per cui non è chiaro se esista un algoritmo polinomiale oppure no; questi problemi sono tutti nella stessa barca: potrebbero essere tutti risolti in tempo polinomiale, oppure nessuno di essi.

I problemi per il millennio (*Millennium problems*) sono sette problemi posti all'attenzione dei matematici dell'Istituto Clay. Per chi risolve uno di questi problemi c'è in palio un milione di dollari. Ad oggi, solo un problema è stato risolto (la Congettura di Poincaré).

Problema astratto Un 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.

Ad esempio prendiamo in considerazione il problema del cammino minimo (**SHORTEST-PATH**). Una sua istanza è una quadrupla (V, E, u, v) , e una sua soluzione è una sequenza ordinata di vertici v_1, \dots, v_k .

Nota. Possono esistere più soluzioni associate alla stessa istanza.

Tipologie di problemi

Esistono diverse tipologie di problemi: problemi di **ottimizzazione**, di **ricerca** o di **decisione**.

Nei **problemi di ottimizzazione** data un'istanza, dobbiamo trovare la “migliore” soluzione secondo criteri specifici. Ad esempio nel problema del cammino minimo dato un grafo G e due nodi u, v , dobbiamo trovare il cammino più breve fra essi.

Nei **problemi di ricerca** data un'istanza, dobbiamo trovare una possibile soluzione fra quelle esistenti. Ad esempio nel problema della ricerca di un cammino dato un grafo G e due nodi u, v , dobbiamo trovare un cammino fra essi.

Nei **problemi di decisione** data un'istanza, cerchiamo semplicemente di verificare se soddisfa o meno una data proprietà. La relazione in questo caso è una funzione $R : I \rightarrow \{0, 1\}$. Prendiamo di nuovo in considerazione il problema del percorso minimo ma stavolta ci poniamo una domanda. Dati un grafo G , due nodi u, v e un valore k , esiste un cammino tra u e v di lunghezza minore o uguale a k ?

Equivalenza fra problemi di ottimizzazione e decisione In quanto la versione decisionale dei problemi è più semplice da trattare matematicamente, ragioneremo in termini di problemi di decisione e non di ottimizzazione.

Difatti se posso risolvere efficientemente la versione di ottimizzazione, allora posso risolvere efficientemente la versione di decisione. È vero anche l'opposto. Ossia, se *non* posso risolvere efficientemente la versione di decisione, allora *non* posso risolvere efficientemente la versione di ottimizzazione.

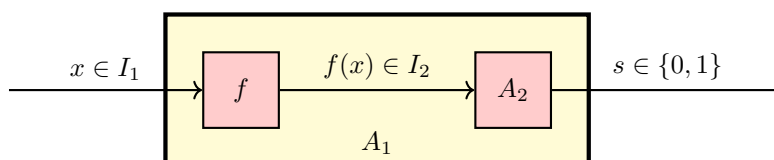
Ad esempio, riconducendoci sempre al problema del cammino più breve, se conosco il cammino più breve, posso rispondere alla domanda decisionale per qualunque valore di k .

18.1 Riduzioni

18.1.1 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 (si scrive matematicamente $R_1 \leq_p R_2$) se esiste una funzione $f : I_1 \rightarrow I_2$ con le seguenti proprietà:

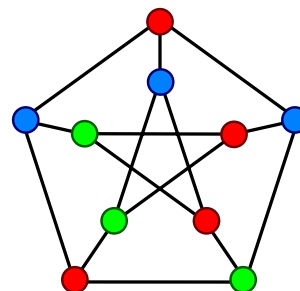
- f è calcolabile in tempo polinomiale;
- per ogni istanza x del problema R_1 e ogni soluzione $s \in \{0, 1\}$, $(x, s) \in R_1 \Leftrightarrow (f(x), s) \in R_2$.



18.1.2 Colorazione di grafi

Dati un grafo non orientato $G = (V, E)$ e un insieme di colori C , 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.

Possiamo formulare il problema sia sotto forma di problema di ottimizzazione, che decisionale. Nella forma decisionale dobbiamo determinare *se esiste*, dato un grafo non orientato $G = (V, E)$ e un valore k , una colorazione G con k colori; mentre nella forma di ottimizzazione dobbiamo restituire la colorazione che necessita del numero minimo di colori.



18.1.3 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.

Esistono due versioni di questo problema, entrambe espresse in forma decisionale.

Le prima versione è quella classica: 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.

Mentre nella seconda abbiamo qualche numero già presente: data una matrice $n^2 \times n^2$ elementi con alcuni numeri già presenti nella matrice, 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 $V = \{(x, y) : 1 \leq x \leq n^2, 1 \leq y \leq n^2\}$

$[(x, y), (x', y')] \in E \Leftrightarrow$

- $x = x'$; oppure,
- $y = y'$; oppure,
- $(\lceil x/n \rceil = \lceil x'/n \rceil) \wedge (\lceil y/n \rceil = \lceil y'/n \rceil)$

$$C = \{1, \dots, n\}.$$

Se abbiamo una soluzione per la colorazione, allora abbiamo una soluzione algoritmica per il Sudoku. Scriveremo quindi che $\text{SUDOKU} \leq_p \text{GRAPH-COLORING}$.

Troviamo diverse applicazioni. Come ad esempio nell'assegnamento delle radio frequenze in un insieme di torri cellulari. O nell'allocazione degli esami universitari.

18.1.4 Insieme indipendente

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

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

Possiamo formulare il problema sia sotto forma di problema di ottimizzazione, che decisionale. Nella forma decisionale dobbiamo determinare *se esiste*, dato un grafo non orientato $G = (V, E)$ e un valore k , un insieme indipendente di dimensione k ; mentre nella forma di ottimizzazione dobbiamo restituire il più grande insieme indipendente presente nel grafo.

Packing problem 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. Il problema dello zaino ne è l'esempio principe.

18.1.5 Copertura di vertici

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

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

Possiamo formulare il problema sia sotto forma di problema di ottimizzazione, che decisionale. Nella forma decisionale dobbiamo determinare *se esiste*, dato un grafo non orientato $G = (V, E)$ e un valore k , una copertura di vertici di dimensione al massimo k ; mentre nella forma di ottimizzazione dobbiamo restituire la copertura dei vertici di dimensione minima.

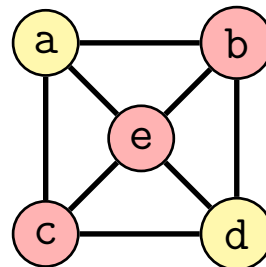
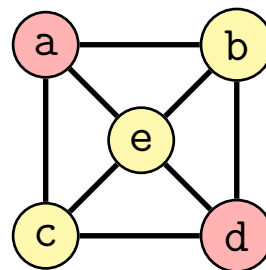
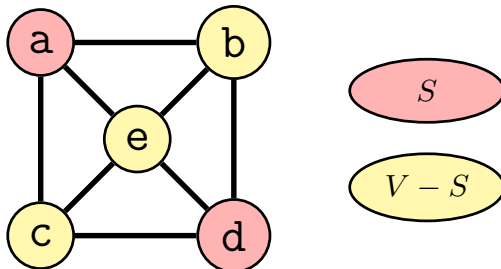
Covering problem Questo è un esempio di covering problem, un problema in cui si cerca di ottenere il più piccolo insieme in grado di coprire un insieme arbitrario di oggetti con il più piccolo sottoinsieme di questi oggetti.

18.1.6 Riduzione per problemi duali

Se $S \subseteq V$ è un insieme indipendente, allora $V - S$ è una copertura di vertici

Se S è un insieme indipendente:

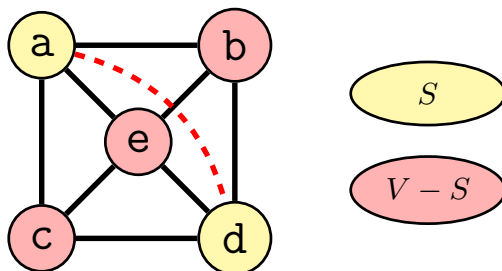
- ogni arco (x, y) non può avere entrambi gli estremi in S ;
- quindi almeno uno dei due deve essere in $V - S$.



Se $V - S$ è una copertura di vertici, allora $S \subseteq V$ è un insieme indipendente

Supponiamo per assurdo che S non sia un insieme indipendente:

- 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 una copertura di vertici, il che è assurdo.



18.1.7 Equivalenza dei problemi

Se il problema della copertura di vertici è riducibile al problema dell'insieme indipendente.

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

E il problema dell'insieme indipendente è riducibile al problema della copertura di vertici.

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

Abbiamo dimostrato che i due problemi sono equivalenti.

18.1.8 Soddisfacibilità di formule booleane

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.

Ad esempio se prendiamo l'espressione $(x \vee \bar{y} \vee z) \wedge (\bar{x} \vee w) \wedge y$ e assegniamo ad ogni variabile il valore di verità **true**, allora possiamo riscrivere l'espressione come $(\text{true} \vee \text{false} \vee \text{false}) \wedge (\text{false} \vee \text{true}) \wedge \text{true}$, semplificando ulteriormente otteniamo $\text{true} \wedge \text{true} \wedge \text{true}$, ed infine **true**. Possiamo dedurre quindi che la formula booleana iniziale è soddisfacibile, in quanto esiste almeno un'interpretazione delle variabili che la rende vera.

Riduciamo il problema generale ad una forma che include solo tre letterali. Chiameremo questo problema 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 il problema 3-SAT è riducibile polinomialmente a quello dell'insieme indipendente. Ossia

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

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 (che chiameremo *arco di conflitto*).

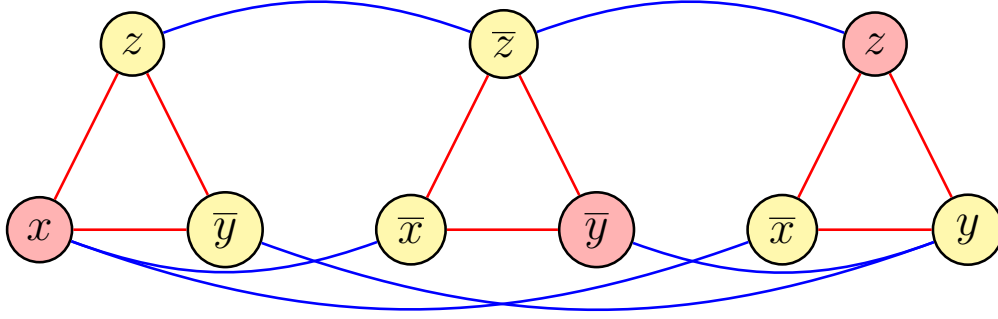


Figura 18.1: La formula 3-SAT è soddisfacibile se e solo è possibile trovare un insieme indipendente di dimensione esattamente k .

Riduzione da problema particolare a problema generale

Ovviamente il problema di soddisfacibilità delle formule booleane con sole tre variabili è riducibile polinomialmente al problema più generale. Scriviamo quindi che $3\text{-SAT} \leq_p \text{SAT}$.

Possiamo dimostrare l'inverso? Ossia che $\text{SAT} \leq_p 3\text{-SAT}$?

È 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”:

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

18.1.9 Proprietà transitiva della riduzione polinomiale

È facile intuire che la nozione di riducibilità polinomiale gode della proprietà transitiva.

$$\text{SAT} \leq_p \text{INDIPENDENT-SET} \leq_p \text{VERTEX-COVER}$$

18.2 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:

- $\text{TIME}(f(n))$ l'insieme dei problemi decisionali risolvibili da un algoritmo che lavora in tempo $\mathcal{O}(f(n))$;
- $\text{SPACE}(f(n))$ gli insiemi dei problemi decisionali risolvibili da un algoritmo che lavora in spazio $\mathcal{O}(f(n))$.

La classe \mathbb{P} è la classe dei problemi decisionali risolvibili in tempo polinomiale nella dimensione n dell'istanza di ingresso:

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

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

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

Nota. $\mathbb{P} \subseteq \text{PSPACE}$.

18.3 Classe NP

18.3.1 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 $(x, \mathbf{true}) \in R$.

Un'assegnamento di verità alle variabili della formula è un certificato per il problema di soddisfacibilità delle formule booleane. Un'associazione nodo-colore $f : V \rightarrow \{1, \dots, k\}$ è un certificato per il problema di colorazione dei grafi. Un sottoinsieme di V di k elementi è un certificato per il problema degli insiemi indipendenti. Tutti questi “certificati” hanno dimensione polinomiale nella dimensione dell'input.

I certificati possono essere verificati in tempo polinomiale:

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

Nota. La classe NP è l'insieme di tutti i problemi che ammettono un certificato verificabile in tempo polinomiale.

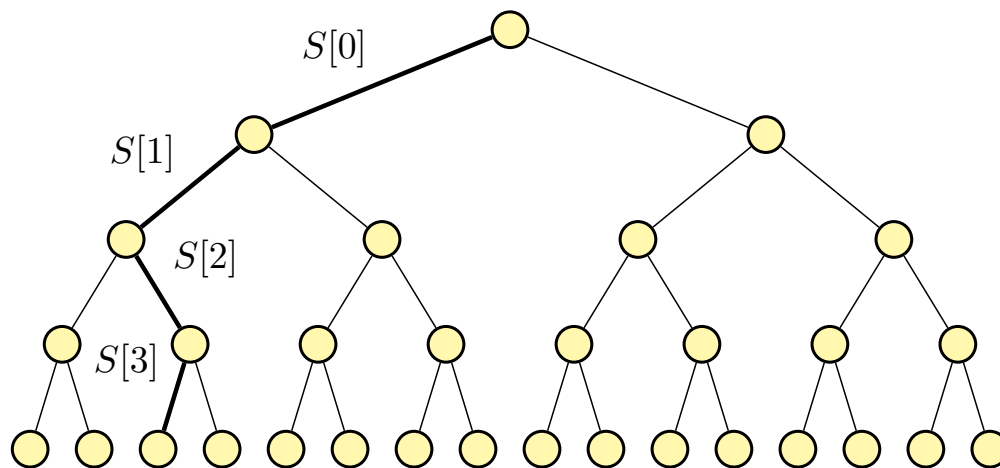
Certificati non polinomiali Esistono dei certificati che non possono essere verificati in tempo polinomiale. Ad esempio il problema *Quantified Boolean Formula (QBF)* è una generalizzazione del problema SAT nel quale ad ogni variabile possono essere applicati quantificatori universali ed esistenziali. Un esempio è la formula $\forall x \exists y \exists z ((x \vee z) \wedge y)$. Si ritiene che un certificato del genere non esista.

18.3.2 Definizione basata su non determinismo

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

In maniera *molto* informale: dato uno stato ed un elemento di input, una macchina non deterministica può andare in un insieme finito di altri stati. Esistono due interpretazioni per la macchina non deterministica. Può essere una macchina che “azzecca” sempre la scelta giusta, oppure può essere una macchina non deterministica che si divide in un insieme finito di copie, una per scelta possibile.

Prendiamo ad esempio il problema SAT con quattro variabili.



18.3.3 Relazioni fra problemi

Lemma 1. Se $R_1 \leq_p R_2$ e $R_2 \in \mathbb{P}$, allora anche R_1 è contenuto in \mathbb{P} .

Dimostrazione. Sia $T_f(n) = \mathcal{O}(n^{k_f})$ il tempo necessario per trasformare un input di R_1 , in input di R_2 , tramite una funzione f . Sia $T_2(n) = \mathcal{O}(n^{k_2})$ il tempo necessario per risolvere R_2 . Qual è la complessità di R_1 ?

La funzione f può prendere un input di dimensione n e trasformarlo in un input di dimensione $\mathcal{O}(n^{k_f})$ per R_2 . Il tempo per risolvere R_1 sarà quindi $T_1(n) = \mathcal{O}(n^{k_f} n^{k_2})$.

Possiamo dedurre che $T_1(n)$ è polinomiale. □

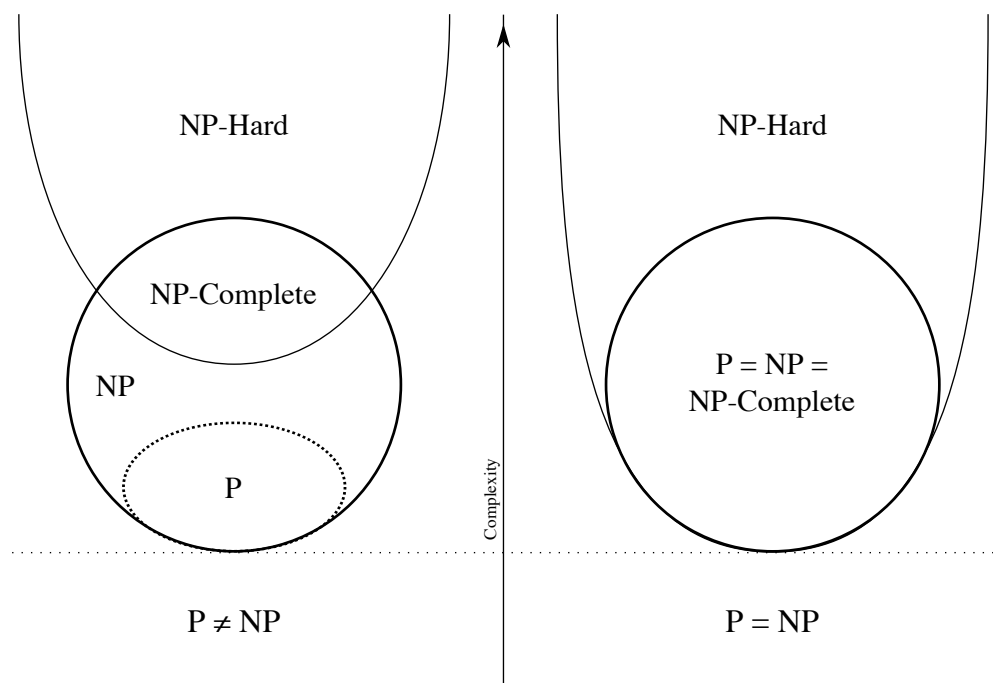
18.4 Problemi NP-completi

Introduciamo alcune 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 NP ed è NP-arduo.

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



Dimostrare che un problema è contenuto in NP è semplice. Dimostrare che un problema è NP-completo richiede una dimostrazione difficile, apparentemente impossibile: tutti i problemi in NP sono riducibili polinomialmente a tale problema, anche quelli che non conosciamo!

Nel 1973 Leonid Levin ha dimostrato in maniera indipendente il seguente teorema.

Teorema (Teorema di Cook-Levin). *SAT* è NP-completo.

La dimostrazione del Teorema è complessa, è basata sugli stati della macchina di Turing.

Problemi introdotti oggi

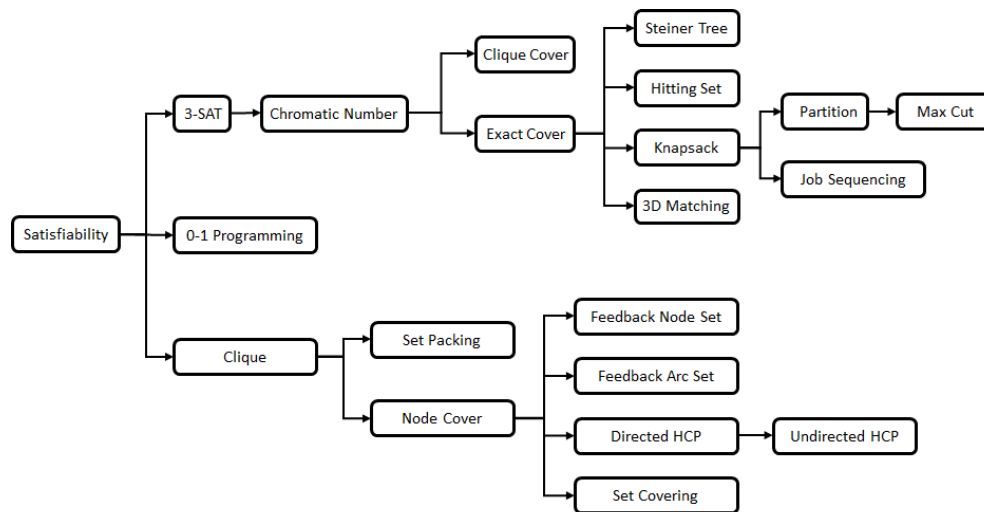
Partendo dalle riduzioni viste oggi e utilizzando il Teorema di Cook-Levin, otteniamo:

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

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

In “*Reducibility Among Combinatorial Problems*” Richard Karp ha stilato una lista di 21 problemi NP-completi.

I 21 problemi NP-completi di Karp

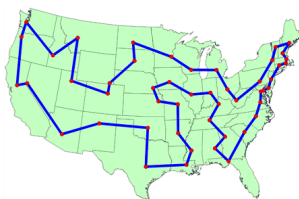
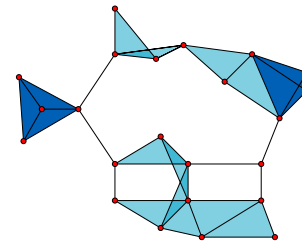


18.4.1 Problemi NP-Completi “Classici”

Cricca (CLIQUE)

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

Questo problema trova applicazioni in bioinformatica, ingegneria elettronica e chimica.



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 non superiore a k ?

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 $Ax \leq b$?

Ad esempio

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

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

$$x_1 + x_3 + x_4 \geq 1$$

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

Copertura esatta di insiemi

Dato un insieme X e una collezione $\mathcal{Y} = \{Y_1, \dots, Y_n\}$ di sottoinsiemi di X , esiste una sottocollezione $\mathcal{Z} \subseteq \mathcal{Y}$ che partizioni X ?

Supponiamo che ad esempio l'insieme X sia $\{1, 2, 3, 4, 5, 6, 7\}$ e la collezione $\mathcal{Y} = \{A, B, C, D, E, F\}$ sia così composta:

$$A = \{1, 4, 7\} \quad B = \{1, 4\} \quad C = \{4, 5, 7\} \quad D = \{3, 5, 6\} \quad E = \{2, 3, 6, 7\} \quad F = \{2, 7\}$$

La sottocollezione che partiziona X è $\mathcal{Z} = \{B, D, F\}$.

Partizione (PARTITION)

Dato un vettore A contenente n interi positivi, esiste un sottoinsieme $S \subseteq \{1 \dots n\}$ tale che $\sum_{i \in S} A[i] = \sum_{i \notin S} A[i]$?

Somma di sottoinsieme (SUBSET-SUM)

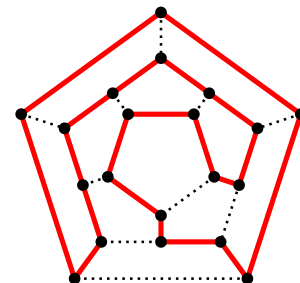
Dati un vettore A contenente n interi positivi ed un intero positivo k , esiste un sottoinsieme $S \subseteq \{1 \dots n\}$ tale che $\sum_{i \in S} a[i] = k$?

Zaino

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" $p[i] \in \mathbb{Z}^+$ e da un peso $w(i) \in \mathbb{Z}^+$. 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 ?

Circuito hamiltoniano

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



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? È un problema NP-completo.

Circuito euleriano Dato un grafo non orientato G , esiste un circuito che attraversi ogni *arco* una e una sola volta? È un problema in P.

Cammini massimi Dato un grafo $G = (V, E)$ e una funzione di peso w sugli archi, trovare il cammino con il peso *massimo*. È un problema NP-completo.

Cammini minimi Dato un grafo $G = (V, E)$ e una funzione di peso w sugli archi, trovare il cammino con il peso *minimo*. È un problema in P.

18.4.2 Problemi aperti

Isomorfismo fra grafi Il problema dell'isomorfismo fra grafi richiede di determinare se due grafi finiti sono isomorfi. Non sappiamo ancora se il problema sia NP -completo o sia in \mathbb{P} . Il dibattito è ancora in corso.

Primalità Dato un numero n , determinare se n è primo. Questo problema è incluso in \mathbb{P} .

Fattorizzazione Dato un numero n , individuare i fattori primi che lo compongono. Questo problema è sicuramente contenuto in NP , si presume che non sia contenuto né in \mathbb{P} , né che sia NP -completo.

Spunti di lettura

Potete approfondire l'argomento nei seguenti testi:

- Jeff Erickson, Algorithm. Cap. 12, NP-Hardness <http://jeffe.cs.illinois.edu/teaching/algorithms/>;
- Sanjeev Arora, Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009. <http://theory.cs.princeton.edu/complexity/>.

