Appunti lezione – Capitolo 18
Problemi intrattibili

Alberto Montresor

04 Maggio, 2018

1 Introduzione

Finora, abbiamo analizzato e accettato solo algoritmi con tempo polinomoniale nella dimensione dell'input :

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

E' naturale chiedersi se tutti i problemi possono essere risolti in tempo polinomiale. La risposta è no:

- esistono problemi che per essere risolti hanno bisogno di un tempo almeno esponenziale;
- ci sono problemi per cui è stato dimostrato che non esiste alcuna soluzione, come il problema della terminazione;

Qui ci occuperemo di una classe di problemi per per cui non è chiaro se esiste un algoritmo polinomiale oppure no. E' una domanda aperta dal 1971; esiste un premio di un milione dollari per la soluzione (Clay Institute, 7 Millenium Problems [1], fra cui una congettura di Poincaré del 1904 [4], "risolta" positivamente da Grigori Perelman [3], a cui nel 2006 è stata assegnata la medaglia Fields [2], poi rifiutata). Tutti pensano che la risposta sia negativa, visto che:

- tutti questi problemi sono sulla stessa barca; o possono essere risolti in tempo polinomiale, oppure no;
- da trent'anni e oltre si cercano soluzioni a tutti questi problemi, senza successo

2 Problemi: caratterizzazione

Per poter definire i concetti successivi, dobbiamo inserire alcuni formalismi.

Definizione 1 (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.

Per esempio, un'istanza del problema SHORTEST-PATH è una quadrupla (V, E, u, v), mentre una soluzione è una sequenza ordinata di vertici. Più soluzioni posso essere associate alla stessa istanza.

3 Problemi: tipologia

Possiamo (sommariamente) considerare due grandi classi di problemi:

- **Problemi di decisione**: problemi che richiedono una risposta binaria (*decisione*), ovvero verificano se una certa istanza soddisfa o meno una proprietà. Esempio: connessione di un grafo.
- **Problemi di ottimizzazione**: problemi che, dato un'istanza x, richiedono di calcolare il costo della migliore soluzione fra tutte le soluzioni $x, s \in R$, e ovviamente riportare la soluzione. La soluzione viene valutata secondo criteri specifici, come il costo del cammino nel problema dei camminimi minimi.

I concetti che vedremo sono valutati in termini di problemi di decisione.

Perché?: Perché per tali problemi non dobbiamo preoccuparci del tempo necessario per restituire la risposta, ma solo del tempo di calcolo. Stesso ragionamento per lo spazio di memoria.

Da ottimizzazione a decisione: Purtroppo, i problemi "interessanti" sono spesso di ottimizzazione; questo non si accorda con la nostra scelta di utilizzare problemi di decisione.
Tuttavia, è possibile "trasformare" un problema di ottimizzazione in un problema equivalente di decisione, e viceversa.

Esempio 0.1 (Trasformazione ottimizzazione-decisione)**.**

- Il problema di ottimizzazione dei cammini minimi richiede di trovare la lunghezza del cammino più corto fra una coppia di nodi u, v.
- Il problema di decisione dei cammini minimi richiede di dire se, data una coppia u, v, esiste un cammino fra essi di lunghezza minore o uguale ad un valore k.

I due problemi sono equivalenti; infatti, se risolvo il problema di ottimizzazione, ho risolto il problema di decisione per tutti i valori di k. Se invece ho risolto il problema di decisione, posso risolvere il problema di ottimizzazione semplicemente trovando il più piccolo valore k per cui il problema di decisione ritorna vero. I valori k da provare sono tutti quelli compresi fra 1 e $|V| - 1$.

3.1 Riducibilità polinomiale

Dati due problemi $R_1 \subseteq I_1 \times \{0, 1\}$ e $R_2 \subseteq I_2 \times \{0, 1\}$, diremo che R_1 è **riducibile polinomialmente** a R_2 (e scriveremo $R_1 \leq_p R_2$) se esiste una funzione $f : I_1 \to I_2$ 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 $(f(x), s) \in R_2$.

Quindi, se esiste un algoritmo per risolvere R_2, potremmo utilizzarlo per risolvere in tempo polinomiale R_1.

4 Alcuni esempi di riduzione

4.1 Riduzione da problema particolare a problema generale

Esempio 0.2 (Colorazione di grafi)**.** Sia dato un grafo non orientato $G = (V, E)$ e un insieme di colori C di dimensione k; una colorazione dei vertici è un assegnamento $f : V \to C$ sui nodi che "colora" ogni nodo con uno dei valori in C. Un assegnamento valido è tale per cui nessuna coppia di nodi adiacenti ha lo stesso colore.

Esempio 0.3 (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.

Questo problema può essere espresso come un problema di colorazione di grafi. Consideriamo $n = 3, n^2 = 9$, ovvero la forma standard. Il grafo in questione ha 81 vertici, uno per ogni elemento della matrice. I vertici sono etichettati (x, y), dove $y = 1 \dots 9$ è la riga e $x = 1 \dots 9$ è la colonna. Due vertici sono uniti da un arco se e solo se:

- $x = x'$; oppure,
- $y = y'$; oppure,
- $(\lceil x/3 \rceil = \lceil x'/3 \rceil) \wedge (\lceil y/3 \rceil = \lceil y'/3 \rceil)$

L'insieme di colori è dato dall'insieme $\{1, \ldots, 9\}$. E' facile vedere che la costruzione di un grafo del Sudoku è ottenibile in tempo polinomiale.
Questo è un esempio di riduzione. Quello che ci dice la riduzione è che il problema della colorazione è almeno tanto difficile quanto il Sudoku; ovvero, se abbiamo una soluzione per la colorazione, allora abbiamo una soluzione (algoritmica) per il Sudoku. Se trovo una soluzione per il primo, trovo una soluzione anche per il secondo.

4.2 Riduzione tramite problemi contrapposti

Esempio 0.4 (Insieme Indipendente)**.** Sia dato un grafo non orientato $G = (V, E)$; diciamo che un insieme $S \subseteq V$ è indipendente se e solo se non esistono archi che uniscono due nodi di S. Formalmente, $\forall (x, y) \in E : x \notin S \vee y \notin S$.

Esempio 0.5 (Insieme Indipendente massimo, decisionale)**.** Sia dato un grafo non orientato $G = (V, E)$ e un valore k; esiste un insieme indipendente di dimensione almeno k?

Questo è un esempio di "packing problem", un problema in cui si cerca di inserire il maggior numero di vertici, la cui scelta è però soggetta a vincoli.

Esempio 0.6 (Copertura di Vertici)**.** Sia dato un grafo non orientato $G = (V, E)$; diciamo che un insieme $S \subseteq V$ è un vertex cover se e solo se ogni arco ha almeno un estremo in S. Formalmente, $\forall (x, y) \in E : x \in S \vee y \in S$.

Esempio 0.7 (Vertex Cover minimo, decisionale)**.** Sia dato un grafo non orientato $G = (V, E)$ e un valore k; esiste un vertex cover 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.
E' possibile dimostrare che un insieme S è indipendente se $V - S$ è un vertex cover. Infatti:

- 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 $V - S$, quindi $V - S$ è un vertex cover.
- se $V - S$ è un vertex cover, vogliamo dimostrare che S è un insieme indipendente. Supponiamo per assurdo che non lo sia, ed esista un arco (x, y) che unisce due nodi in S. Allora nessuno degli estremi sta in $V - S$, il che implica che $V - S$ non è un vertex cover.

Abbiamo quindi dimostrato che Vertex Cover $\leq_p$ Independent Set e che Independent Set $\leq_p$ Vertex Cover.

4.3 Riduzione tramite "gadget"

Definizione 2 (Formule booleane in forma normale congiuntiva)**.** Dato un insieme V di *variabili*, $|V| = n$, chiamiamo *letterale* una variabile v o la sua negazione $\overline{v}$; chiamiamo *clausola* una disgiunzione di letterali (letterali separati da **or**); un formula in forma normale congiuntiva è una congiunzione di clausole (clausole separate da and). Esempio:

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

SODDISFATTIBILITÀ DI FORMULE BOOLEANE (SAT). *Data una espressione in forma normale congiuntiva, il problema della soddisfattibilità consiste nel decidere se esiste una assegnazione di valori di verità alle variabili che rende l'espressione vera.*

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

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

- Per ogni clausola, aggiungiamo un terzetto di nodi, collegati fra di loro in modo completo;

- Per ogni letterale che compare in modo normale e in modo negato, aggiungere un arco fra di essi (arco di conflitto).

A questo punto, se ci sono k clausole, ci domandiamo se è possibile trovare un Independent Set di dimensione almeno k (in realtà, per la costruzione dei triangoli, avrà dimensione esattamente k).
Abbiamo quindi dimostrato che 3-SAT $\leq_p$ Independent Set.

5 Transitività delle Riduzioni

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

$$\text{3-SAT} \leq_p \text{Independent Set} \leq_p \text{Vertex Cover}$$

6 Le classi $\mathbb{P}$, $\mathbb{PSPACE}$

Definizione 3. Dati un problema di decisione R e un algoritmo A 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 $(x, s) \in R$.

Definizione 4. Data una qualunque funzione $f(n)$, chiamiamo $\mathbb{TIME}(f(n))$ e $\mathbb{SPACE}(f(n))$ gli insiemi dei problemi decisionali che possono essere risolti rispettivamente in tempo e spazio $O(f(n))$.

Ad esempio, il problema di cercare un elemento in un dizionario di n elementi organizzato come un albero RB è $\mathbb{TIME}(\log n)$ e $\mathbb{SPACE}(n)$. Tuttavia, queste definizioni sono troppo dettagliate, quindi definiamo alcune classi più ampie.

Definizione 5 (Classe $\mathbb{P}$)**.** La classe $\mathbb{P}$ è la classe dei problemi decisionali risolvibili in tempo polinomiale nella dimensione n dell'istanza di ingresso:

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

Definizione 6 (Classe $\mathbb{PSPACE}$)**.** La classe $\mathbb{PSPACE}$ è la classe dei problemi decisionali risolvibili in spazio polinomiale nella dimensione n dell'istanza di ingresso:

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

7 La classe NP

Consideriamo la classe dei problemi decisionali; data una qualunque istanza, è possibile provare che la risposta associata all'istanza è **true** fornendo una descrizione (**certificato**) della possibile soluzione che soddisfa i requisiti del problema. Ad esempio,

- nel caso del problema SAT, un certificato è dato da un assegnamento di verità alle variabili della formula;
- nel caso del problema Colorazione, è un'associazione nodo-colore
- nel caso del problema Insiemi indipendenti, è un sottoinsieme di V

Tutti questi "certificati" hanno dimensione polinomiale nella dimensione dell'input.
Per convincervi che il concetto di certificato non è banale, provate a cercare di esprimere un certificato che dimostra che una certa formula 3-SAT non è soddisfacibile, ovvero non esiste nessun assegnamento delle variabili che dà valore vero. (Questo è un istanza delle più generale problema Quantified Boolean Formula o QBF, dove la formula è preceduta da un elenco di quantificatori universali o esistenziali).
Questo ci suggerisce un ulteriore modo di discriminare i problemi: in base al tempo di verifica. Informalmente, verificare significa analizzare la soluzione per confermare che questa è effettivamente una soluzione del problema. Negli esempi mostrati in precedenza:

- è sufficiente utilizzare gli assegnamenti di verità e calcolare il risultato della formula;
- è sufficiente visitare l'albero e verificare che l'insieme di archi appartengano effettivamente al grafo e che tutti i nodi siano visitati;
- è sufficiente fare un passo di Dijkstra (per esempio) per vedere se i cammini possono essere migliorati ancora.

E' interessante notare che tutti questi certificati possono essere verificati in tempo polinomiale. Mentre non è noto nessun algoritmo polinomiale per verificare/esprimere un certificato di QBF, di dimensione potenzialmente esponenziale.
In base a questa discriminazione, possiamo definire una nuova classe, detta $\mathbb{NP}$: l'insieme di tutti i problemi che ammettono un certificato verificabile in tempo polinomiale.

8 Non determinismo

Fino ad ora, abbiamo visto solo algoritmi *deterministici*, ovvero algoritmi in cui ogni passo è determinato dallo stato attuale del sistema. Un modo alternativo per vedere la classe $\mathbb{NP}$ (da cui la N) è considerare algoritmi *non deterministici*. [1]
In un algoritmo non deterministico, tutte le volte che ci si pone di fronte ad una scelta, c'è la possibilità di "indovinare" la strada giusta. Ad esempio, nel problema SAT, l'algoritmo potrebbe indovinare ad ogni passo il valore di una variabile, e risolvere quindi il problema in tempo lineare.
La computazione non è più quindi espressa come una sequenza di passi, ma con un albero di possibili scelte. Nel caso di SAT, il problema corrisponde nel cercare un cammino nell'albero che porti ad una foglia, tale per cui gli assegnamenti portano al successo.
La classe $\mathbb{NP}$ può quindi essere vista come l'insieme dei problemi che possono essere risolti in tempo polinomiale da un algoritmo non deterministico.
Notate che QBF non appartiene a $\mathbb{NP}$; infatti, verificare QBF significa verificare un intero albero di scelte.

9 Gerarchia

- $\mathbb{P} \subseteq \mathbb{NP}$: infatti, tutti i problemi decidibili in tempo polinomiale possono semplicemente ignorare il certificato e prendere la loro decisione;
- $\mathbb{NP} \subseteq \mathbb{PSPACE}$: la fase deterministica di verifica può essere condotta solo se il certificato ha dimensioni polinomiali;
- $\mathbb{PSPACE} \subseteq \mathbb{EXPTIME}$: intuitivamente, n^c caselle di memoria possono trovarsi al più in 2^{n^c} stati diversi, che può essere necessario esplorare tutti;

Si ritiene che tutte queste inclusioni siano in realtà strette, ma sono tutti problemi aperti.
Perché studiamo queste classi? Perché $\mathbb{P}$ è l'insieme dei problemi "gestibili"; si noti che $\mathbb{P}$ include anche complessità come n^{100}, ma queste raramente si presentano in pratica, visto che la maggior parte dei problemi si trova entro la complessità n^3.

10 NP-Completezza

In questa classe introdurremo la nozione di $\mathbb{NP}$-completezza, che ci permetterà di caratterizzare i problemi più difficili all'interno della classe $\mathbb{NP}$.

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

[1] Qui facciamo un discorso veramente informale e di basso livello; andrebbe approfondito

Dimostrazione. Sia $t_1(n) = O(n^{k_1})$ il tempo per trasformare un input di R_1 in input di R_2, e $t_2(n) = O(n^{k_2})$ il tempo per risolvere R_2; banalmente sembrerebbe che dovremmo scegliere il valore più alto fra i due; ma la funzione di trasformazione può prendere un input di dimensione n, e trasformarlo in input di dimensione $O(n^{k_1})$; questo viene dato in pasto all'algoritmo per R_2, e otteniamo quindi che il costo totale è $O(n^{k_1 k_2})$. In ogni caso, questo è un valore polinomiale. □

Possiamo a questo punto (finalmente!) definire i problemi $\mathbb{NP}$-ardui ($\mathbb{NP}$-hard) e $\mathbb{NP}$-completi.

Definizione 7 (Problemi $\mathbb{NP}$-ardui, $\mathbb{NP}$-completi)**.** Un problema decisionale R si dice $\mathbb{NP}$-hard se ogni problema $Q \in \mathbb{NP}$ è riducibile polinomialmente a R ($Q \leq_p R$). Un problema decisionale R si dice $\mathbb{NP}$-completo se appartiene alla classe $\mathbb{NP}$ ed è $\mathbb{NP}$-hard.

Sottolineamo che possono esistere problemi $\mathbb{NP}$-hard, ma non appartenenti a $\mathbb{NP}$.

Teorema 1 ($\mathbb{P}$=$\mathbb{NP}$?)**.** Se un qualunque problema decisionale $\mathbb{NP}$-completo appartenesse a $\mathbb{P}$, allora risulterebbe $\mathbb{P} = \mathbb{NP}$.

10.1 Come partire?

Dimostrare che un problema è in $\mathbb{NP}$ è semplice, dimostrare che è $\mathbb{NP}$-completo richiede una dimostrazione difficile, che sembrerebbe impossibile: ovvero che tutti i problemi in $\mathbb{NP}$ sono riducibili polinomialmente a tale problema - anche quelli che non conosciamo!
Fortunatamente, ci viene in aiuto il teorema di Cook:

Teorema 2 (Teorema di Cook)**.** SAT è $\mathbb{NP}$-completo

Dimostrazione. La dimostrazione è enormemente complessa, e quindi non presentabile qui; sostanzialmente, Cook è in grado di esprimere, in modo algoritmico, una formula booleana equivalente a qualunque problema in $\mathbb{NP}$. □

10.2 Come andare avanti

Una volta "partiti" con Cook, è stato Karp ad "andare avanti", dimostrando che SAT può essere risolto tramite altri problemi in $\mathbb{NP}$. Karp è partito da una semplice osservazione sulla transitività della relazione di riduzione:

$$R_1 \prec R_2 \wedge R_2 \prec R_3 \Rightarrow R_1 \prec R_3$$

Ovvero, che è possibile costruire un intera gerarchia di riduzioni. E' possibile dimostrare che SAT $\leq_p$ SAT-3; ovvero, è possibile esprimere una qualunque formula booleana congiuntiva con SAT-3. Sappiamo quindi che:

$$\text{SAT} \leq_p \text{3-SAT} \leq_p \text{Independent Set} \leq_p \text{Vertex Cover}$$

11 Altri esempi

E' anche interessante notare che i problemi $\mathbb{NP}$-completi si nascondono nei problemi più banali, spesso in problemi simili ai problemi per cui esistono soluzioni polinomiali; eppure, sono fondamentalmente diversi:

Esempio 0.8 (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.

Esempio 0.9 (Cammini massimi)**.** Dato un grafo $G = (V, E)$ e una funzione di peso w sugli archi, trovare il cammino con peso massimo. $\mathbb{NP}$-completo

Esempio 0.10 (Ciclo euleriano)**.** Un ciclo euleriano su un grafo orientato connesso è un ciclo che attraversa ciascun *arco* di G una volta sola. Risolvibile in tempo polinomiale.

Esempio 0.11 (Ciclo hamiltoniano)**.** Un ciclo hamiltoniano su un grafo orientato connesso è un ciclo che attraversa ciascun *vertice* di G una volta sola. $\mathbb{NP}$-completo.

Riferimenti bibliografici

[1] Wikipedia. Clay institute, 2006. http://en.wikipedia.org/w/index.php?title=Clay_Mathematics_Institute&oldid=92871507. [Online; versione 10/12/2006].

[2] Wikipedia. Fields medal, 2006. http://en.wikipedia.org/w/index.php?title=Fields_Medal&oldid=92924490. [Online; versione 10/12/2006].

[3] Wikipedia. Grigori perelman, 2006. http://en.wikipedia.org/w/index.php?title=Grigori_Perelman&oldid=93259342. [Online; versione 10/12/2006].

[4] Wikipedia. Poincaré conjecture, 2006. http://en.wikipedia.org/w/index.php?title=Poincar%C3%A9_conjecture&oldid=90286133. [Online; versione 10/12/2006].