Teoria della NP completezza

Luca Tagliavini

March 17-.., 2021

1 Introduzione

I problemi risolvibili tramite algoritmi hanno un costo associato come sappaimo. I problemi NP-completi cercano di capire se un qualcosa e' computabile, entro un dato limite. I risultati di tali ragionamenti saranno dunque valori booleani.

Ecco alcuni esempi di problemi NP-completi:

1. **Commesso viaggiatore**: dato un grafo completo ed un valore k, decidere se esiste un cammino semplice che copre tutti i vertici di peso inferiore a k.

2. **Bin packing problem**: dati n oggetti ognungo con un peso $P[n]$ e k contenitori di capacita' c, decidere se e' possibile distribuire tutti gli n oggetti nei k contenitori senza eccedere la capacita' c.

3. **Sudoku**: data una matrice di dimensione $n^2 \times n^2$(sudoku classico 9×9 organizzata in blocchi di dimensioni $n \times n$ (sudoku classico 3×3) con alcune celle gia' riempite, sia possibile riempire le restanti in modo che ogni riga, ogni colonna e ogni blocco $n \times n$ contenga tutti i numrei compresi tra $1en^2$.

1.1 Formalizzazione

Consideriamo un problema K come una relazione tra $Q \subseteq I \times S$ dove

- I e' l'insieme delle *istanze d'ingresso.*
- S e' l'insieme delle *soluzioni.*

Immaginiamo Q come un predicato che preso in input il dato $x \in I$ e la soluzione $s \in S$ restituisce:

1. $1 \iff (x, s) \in Q$ (s soluzione del problema Q sull'istanza x)

2. 0 altrimenti (s **non** e' soluzione del problema Q sull'istanza x)

1.1.1 Tipologie di problemi

Esistono problemi di vario tipo, come ricerca (trovare un elemento in un array) o di ottimizzazione (trovare il cammino minimo) ma sappiamo da prima che ogni problema di un qualunque tipo ha un suo problema corrispondente del tipo **decisionale** che restituisce un booleano in base alla *trattabilita'* del problema.

1.2 Classi di complessita'

Data una funzione $f(n)$, chiamiamo $TIME(f(n))$ o $SPACE(f(n))$ l'insieme di tutti i problemi che hanno una soluzione algoritmica di costo computazionale $O(f(n))$.

1.2.1 Classe polinomiale

La classe P dei problemi risolvibili in **tempo** *polinomiale* nella dimensione n dell'istanza d'ingresso:

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

La classe $PSPACE$ dei problemi risolvibili in **spazio** *polinomiale* nella dimensione n dell'istanza d'ingresso:

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

1.2.2 Classe esponenziale

La classe $EXPTIME$ dei problemi risolvibini il **tempo** *esponenziale* nella dimensione n dell'istanza d'ingresso:

$$EXPTIME = \cup_{c=0}^{\infty} TIME(2^{n^c})$$

1.2.3 Ordinamento delle classi

E' abbastanza facile inutire come $P \subset EXPTIME$, in quanto:

1. Entambe misurano il tempo.
2. Un problema con tempo polinomiale sara' sempre di complessita' inferiore rispetto a uno di tempo esponenziale.

Capiamo oltretutto che $P \subseteq PSPACE$ in quanto un algoritmo che richiede un tempo polinomiale al piu' ad accedere ad un numero polinomiale di locazioni di verse.
Si puo' oltretutto capire che $PSPACE \subseteq EXPTIME$, in quanto una memoria di n^c locazioni puo' essere al piu' essere in 2^{n^c} stati, che sono sempre inferiori ad un numero esponenziale.

1.3 Esempio: problema SAT

Il problema SAT, ovvero della soddisfacibilita' di una espressione booleana, che contiene solo operazioni di $\wedge, \vee, \bar{\cdot}$. Anche espressioni booleane piu' complesse possono sempre essere ridotte a *forme normali congiuntive*, ovvero forme che contegono piu' clausole ($\vee$ tra letterali, possibilmente negati) a loro volta unite da una serie di $\wedge$.

Una possibile soluzione al problema e' provare ogni coppia di soluzioni, dando ogni possibile valore a ogni n variabile. Prenderemo dunque coppie $\overline{c} \in \mathbb{B}^n$ che ha dimensione 2^n, il che rende il problema appartenente alla famiglia $PSPACE$.

1.4 Certificare vs Verificare

Quando ci viene chiesto di *verificare* se una determinata formula booleana e' soddisfacibile per un numero n di variabili, ci basta capire se esiste almeno una coppia di n variabili tale che l'espressione vale.
Tuttavia, e' desiderabile non solo capire se un problema e' risolvibile, ma bensi' *certificare* restituendo un valore che accerta che tale problema ammette la soluzione data.

1.5 La cateogirna NP

La cateogira NP e' quella categoria di problemi che ammettono *certificati verificabili* in tempo polinomiale.

1.6 Non determinismo

Gli algoritmi visti in precedenza si comportano in maniera *deterministica*, le loro mosse sono prevedibili e ripercorribili in modo sicuro e oggettivo. Immaginiamo ora di ricevere un aiuto esterno, che ci consente di trovare una soluzione conveniente per un determinato problema, e fa proseguire il nostro algoritmo nella direzione giusta. Questa tecnica di soluzione e' detta **non deterministica**, poiche' non ha uno svolgimento prevedibile.

L'albero di tute le scelte *non deterministiche* fatte viene rappresentato come un albero che ha nelle foglie valori compresi in $\mathbb{B}$, dove 1 indica il successo dell'algoritmo, mentre lo 0 indica una strada seguita ma non valida. Se esiste almeno una foglia con valore 1 l'intero calcolo e' fallito e' corretto, e si restituisce dunque 1, altrimenti 0.

1.7 Classe polinomiale non deterministica

Data una funzione $f(n)$, chiamiamo $NTIME(f(n))$ l'insieme dei problemi decisionali che possono essere risolti da un algoritmo non deterministico in tempo $O(f(n))$. La classe NP e' quella dei problemi risolvibili in tempo polinomiale non deterministico nella dimensione n dell'istanza d'ingresso:

$$NP = \cup_{c=0}^{\infty} NTIME(n^c)$$