Algoritmi e complessità

Francesco Tomaselli

16 febbraio 2021

Indice

1 Algoritmi di approssimazione

In questa parte si introdurranno gli algoritmi di approssimazione, dopo aver accennato ad alcuni concetti preliminari legati alla complessità. Si defineranno in particolari classi di problemi, e si definiranno alcune tecniche note nella risoluzione di problemi di ottimizzazione.

1.1 Classi di complessità

Partiamo dalla definizione di algoritmo, per poi arrivare a definire le classi di complessità note.

Algoritmo Un algoritmo per un problema Π può essere visto come una *balck-box* che verrà indicata con A, che opera come segue:
dato un input $x \in I_\Pi$, l'algoritmo A produrrà un output $y \in O_\Pi$, tale che $y \in Sol_\Pi(x)$.

1.1.1 Complessità algoritmica

La complessità algoritmica è lo studio del dispendio di risorse di un algoritmo. Un esempio è il tempo: $T_a : I_\Pi \to \mathbb{N}$, possiamo passare in una notazione $t_a : \mathbb{N} \to \mathbb{N}$ dove il dominio è la lunghezza di input, in quel caso si avrà che $t_a(n) : \max\{T_a(n),\ x \in I_\Pi,\ |x| = n\}$

Date due soluzioni, è preferibile quella asintoticamente minima, più formalmente, quella a numeratore tale che $\lim_{x\to\infty} \frac{t_1}{t_2} = \infty$.

1.1.2 Complessità strutturale

Si definiscono ora due classi di problemi, *P* e *NP*, per poi introdurre i concetti di *riducibilità polinomiale* e di *NP-completezza*.

Classe P La classe *P* corrisponde ai problemi decisionali per cui esiste un algoritmo che opera in tempo polinomiale, ovvero:

$$P = \{\Pi \mid \Pi\ \textit{decisionale},\ \exists\ A\ \textit{per}\ \Pi,\ \textit{t.c.}\ t_a(n) \ = \ O(\textit{Polinomio})\}$$

Classe NP La classe *NP* corrisponde ai problemi decisionali per cui esiste un algoritmo che opera in tempo polinomiale su una macchina non deterministica, ovvero:

$$\begin{aligned} NP = \{\Pi \mid \Pi\ \textit{decisionale},\ \exists\ A\ \textit{per}\ \Pi,\ \textit{t.c.}\ t_a(n) \ = \ O(\textit{Polinomio}) \\ \textit{su una macchina non deterministica}\} \end{aligned}$$

Riducibilità polinomiale Un problema si dice riducibile polinomialmente se esiste un mapping del suo input in un input per un algoritmo polinomiale, ovvero:

$$\Pi_1 \leqslant_p \Pi_2 \; sse \; \exists f : 2^* \to 2^* \; t.c$$

1. f è calcolabile in tempo polinomiale
2. $\forall x \in I_{\Pi 1}, \; f(x) \in I_{\Pi 2}, \; Sol_{\Pi 1}(x) = Sol_{\Pi 2}(f(x))$

NP completezza Un problema Π è NP completo sse $\forall \; \Pi' \in NP, \; \Pi' \leqslant_p \Pi, \; \Pi \in NP$. Ovvero se ogni problema in NP è riducibile polinomialmente al problema che si sta considerando.

Teorema 1.1. *SAT è NP completo.*

Corollario 1.1.1. *Se $\Pi_1 \leqslant_p \Pi_2$ e Π_1 è NP completo, allora Π_2 è NP completo.*

Dimostrazione. $\Pi' \in NP, \Pi' \leqslant_p \Pi_1 \leqslant_p \Pi_2$, quindi Π_2 è NP completo. □

Osservazione 1. *Se trovassi un problema Π NP completo t.c, $\Pi' \leqslant_p \Pi$, allora* P=NP.

1.2 Problemi di ottimizzazione

Nella definizione di un problema di ottimizzazione bisogna tenere conto dei seguenti parametri:

1. Insieme di input I_Π
2. Insieme di output O_Π
3. $F_\Pi : I_\Pi \to 2^{O_\Pi} \setminus \{\emptyset\}$, $F_\Pi(x)$ indica le soluzioni accettabili per l'input x
4. $C_\Pi : I_\Pi \times O_\Pi \to \mathbb{Q}^{>0}$, funzione obiettivo, con $C_\Pi(x, y)$ si indica il valore della funzione obiettivo per l'input x, con soluzione $y \in O_\Pi$
5. $t_\Pi \in \{min, \; max\}$, ovvero il criterio del problema.

Osservazione 2. $Sol(x) = \{y^* \in O_\Pi | y^* \in F_\Pi, \forall y' \in F_\Pi, c(x, y^*) \leqslant (\geq) c(x, y')\}$

Problema di decisione associato Dato un problema di ottimizzazione Π esiste un problema di decisione associato $\hat{\Pi}$.
L'idea è quella di considerare un input del problema originale e un costo alla soluzione, e rispondere in base all'esistenza di una soluzione con quel costo.
In particolare:

$$I_{\hat{\Pi}} = I_\Pi \times \mathbb{Q}^{>0}$$

$$(x, \theta) = C_\Pi(x, y^*(x)) \leqslant (\geq) \theta$$

1.2.1 Classi di ottimizzazione

PO

NPO