# Algoritmi di Approssimazione per Problemi di Ottimizzazione NP-Ardui

*Vincenzo Bonifaci*
Dipartimento di Informatica e Sistemistica
Sapienza Università di Roma, Roma, Italy

**Sommario**

In questa trattazione prenderemo in considerazione problematiche relative al progetto ed analisi di algoritmi efficienti per problemi di ottimizzazione NP-ardui, discutendo stato dell'arte, tendenze di ricerca, e ricadute applicative.

## 1 Introduzione

Procter & Gamble

Nel 1962, la società commerciale Procter & Gamble pubblicizzava un concorso in cui i partecipanti dovevano trovare il più breve cammino chiuso che visitasse tutte e 33 le località mostrate su una mappa degli Stati Uniti. Per la soluzione, la compagnia offriva svariate migliaia di dollari.

NP-completezza

Si trattava di una istanza del problema del commesso viaggiatore (Traveling Salesman Problem, in breve TSP), un problema di ottimizzazione classico che è stato oggetto di innumerevoli studi [Lawler ed altri, 1985]. Nonostante gli sforzi, però, un metodo efficiente di soluzione ha sempre eluso gli studiosi. Le ragioni di queste difficoltà sono divenute più chiare con l'avvento della teoria della NP-completezza, che ha mostrato come il TSP – così come moltissimi altri problemi con importanti applicazioni industriali – sia un problema di ottimizzazione NP-arduo [Karp, 1972]. A meno di risolvere in positivo la questione $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$, quindi, la ricerca di un metodo efficiente di soluzione doveva essere accantonata.

Necessità di approssimazioni

Non per questo, però, è venuta meno la necessità del mondo industriale e scientifico di risolvere problemi di ottimizzazione NP-ardui. In assenza di algoritmi efficienti (tempo polinomiale) che determinino la soluzione ottima di un dato problema, si possono percorrere principalmente due strade: da un lato, si può semplicemente rinunciare a risolvere istanze di grandi dimensioni o comunque ci si può limitare a quelle istanze con una particolare struttura combinatoria che può essere sfruttata per ottenere un algoritmo efficiente; in alternativa, si può fare ricorso ad un algoritmo tempo polinomiale, ottenendo però in genere una soluzione non ottima ma soltanto approssimativamente tale. Infatti, dal punto di vista delle applicazioni può essere perfettamente ragionevole avere

algoritmi efficienti che determinino soluzioni con un costo aggiuntivo non superiore al 2% o al 5% del costo ottimo. Dalle precedenti definizioni scaturisce la seguente definizione: per un dato problema di minimizzazione, il *rapporto di approssimazione* di un algoritmo è il minimo valore $\rho$ per il quale le soluzioni generate dall'algoritmo hanno costo sempre limitato superiormente da $\rho$ volte il costo ottimo. Per un problema di massimizzazione si può fornire una definizione analoga.

Rapporto di approssimazione

Le principali tematiche di ricerca in questo contesto si articolano secondo le seguenti linee fondamentali:

Linee di ricerca

1. Progetto ed analisi di algoritmi polinomiali con rapporto di approssimazione garantito per specifici problemi di ottimizzazione NP-ardui.
2. Sviluppo di tecniche algoritmiche generali applicabili all'approssimazione di problemi NP-ardui.
3. Studio di limiti inferiori all'approssimabilità garantita da algoritmi polinomiali.

Non è fuori luogo una precisazione sull'impatto pratico di questi studi: infatti, quando nella pratica si cercano soluzioni con errore entro il 2% o il 5% dall'ottimo, ci si può chiedere a che fine studiare algoritmi che approssimano l'ottimo entro un fattore 3 o magari $O(\log n)$, dove $n$ è la dimensione dell'input.

L'impatto pratico: 3 vs $O(\log n)$

La risposta è duplice. Innanzitutto, il rapporto di approssimazione descrive la performance di un algoritmo solo sulle istanze più patologiche; molti algoritmi commettono un errore massimo dell'entità desiderata su istanze tipiche, sebbene il limite d'errore garantito nel caso peggiore sia molto più grande. Inoltre, la garanzia sul limite di errore dovrebbe essere vista come una misura obiettiva che costringe ad esplorare maggiormente la struttura combinatoria del problema studiato. Un algoritmo con un limite teorico dimostrato va visto soprattutto come un'idea di base che dovrà essere di volta in volta adattata alle categorie di istanze tipiche della specifica applicazione.

Istanze patologiche, misura obiettiva

# 2 Progetto ed analisi di algoritmi di approssimazione

**Algoritmi combinatori.** I primi lavori nella letteratura degli algoritmi di approssimazione hanno riguardato l'analisi di semplici euristiche di tipo sequenziale o di tipo goloso (*greedy*). Si consideri ad esempio il problema (NP-arduo) di allocare un insieme di lavori su più macchine in modo da minimizzare il tempo di completamento complessivo (*makespan scheduling*). Un algoritmo naturale per questo problema è l'algoritmo sequenziale che considera i lavori uno alla volta e li assegna di volta in volta alla macchina al momento meno carica. Graham [Bell System Tech J 1966] ha provato che questo algoritmo garantisce un rapporto di approssimazione pari a $2 - 1/m$, dove $m$ è il numero di macchine. L'idea consiste nel considerare il carico della macchina che determina il tempo di

Algoritmi sequenziali: scheduling

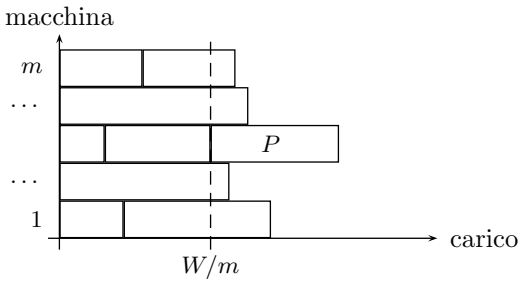


Figura 1: Illustrazione dell'analisi dell'algoritmo di Graham. Il costo incorso dall'algoritmo è ALG $= P + (W/m)$. Ogni soluzione costa almeno $P$ e almeno $(P + W)/m$. Quindi ALG $\leq (2 - 1/m)$ OPT, dove OPT è il costo ottimo.

completamento massimo ed esprimerlo come somma di due termini (Figura 1): $P$ (tempo di processamento dell'ultimo lavoro assegnato alla macchina) e $W/m$ (limite inferiore di carico di tutte le macchine). Il rapporto di approssimazione segue da due semplici osservazioni: (1) ogni soluzione deve assegnare il lavoro di lunghezza $P$ ad una singola macchina; (2) ogni soluzione deve suddividere tra le macchine un carico totale pari ad almeno $P + W$. Confrontando questi limiti inferiori con il costo della soluzione generata dall'algoritmo si ottiene il rapporto di approssimazione sopra citato. Questa interazione tra limiti inferiori (sul costo di soluzioni arbitrarie) e limiti superiori (sul costo della soluzione generata dall'algoritmo) gioca un ruolo cruciale in tutte le analisi di algoritmi di approssimazione.

Algoritmi golosi: set cover

Un altro dei primi esempi di analisi riguarda il problema della copertura di insieme (*set cover*) per il quale Johnson [Johnson, JCSS 1974] e successivamente altri autori hanno dimostrato che il naturale algoritmo goloso, che seleziona l'insieme che copre un maggior numero di elementi non ancora coperti, ha un rapporto di approssimazione pari a $O(\log n)$ dove $n$ è il numero di elementi dell'universo.

Programmazione dinamica: knapsack

In generale, le tecniche già ben note nel campo dei problemi risolubili in tempo polinomiale hanno trovato nuova applicazione nell'ambito degli algoritmi approssimati. La classica tecnica della programmazione dinamica, ad esempio, è stata applicata insieme ad un passo di arrotondamento per ottenere un cosiddetto schema pieno di approssimazione per il problema della bisaccia [Ibarra e Kim, JACM 1975]. È noto infatti che il problema della bisaccia, sebbene NP-arduo, è risolubile in tempo polinomiale quando i pesi degli oggetti hanno un valore limitato superiormente da una funzione polinomiale nel numero degli oggetti – proprio sfruttando la programmazione dinamica. L'idea dell'algoritmo di Ibarra e Kim è di troncare le rappresentazioni binarie dei pesi ad un numero di bit dipendente da un parametro razionale $\epsilon > 0$ di ingresso. Applicando all'istanza risultante l'algoritmo di programmazione dinamica, è possibile dimostrare che la soluzione ottenuta approssima il costo ottimo entro un fattore $1 - \epsilon$ e che il costo computazionale dell'algoritmo è polinomiale nel numero di oggetti e in $1/\epsilon$. Una tale situazione, in cui per ogni $r > 1$ esiste un algoritmo

polinomiale che approssima la soluzione ottima entro un fattore $r$ (schema di approssimazione), costituisce uno dei casi in cui l'intrattabilità del problema si rivela, per molti fini pratici, aggirabile.

**Algoritmi basati sulla programmazione lineare.** Successivamente ai risultati sugli algoritmi combinatori descritti sopra, in letteratura sono stati analizzati algoritmi più sofisticati basati sull'idea di risolvere appropriati programmi lineari (nel senso della ricerca operativa), usando poi le soluzioni ottenute per costruire una soluzione approssimata al problema considerato. Algoritmi di questo tipo hanno il vantaggio di poter sfruttare la potente teoria della programmazione lineare.

LP Rounding

Un tipico esempio di questa classe di algoritmi è l'algoritmo per il problema del *vertex cover* di Hochbaum [Hochbaum, SICOMP 1982]. Nel problema del vertex cover l'input è rappresentato da un grafo non orientato $G$ e l'obiettivo è trovare un sottoinsieme dei nodi a cardinalità minima tale che ogni arco del grafo sia incidente ad un nodo appartenente al sottoinsieme. Il problema può essere formulato come problema di programmazione lineare intera:

Esempio: Vertex Cover

$$\begin{aligned} \min \sum_{i \in V(G)} & x_i && \text{(PLI)} \\ x_i + x_j & \geq 1 && \text{per ogni } \{i,j\} \in E(G) \\ x & \in \{0,1\}^{E(G)}. \end{aligned}$$

Rilassando il vincolo di interezza delle variabili si ottiene il cosiddetto rilassamento lineare del problema. Una soluzione ottima per il rilassamento può essere trovata in tempo polinomiale utilizzando noti algoritmi di programmazione matematica. Si ottiene così una soluzione frazionaria $x^*$ ottima per il rilassamento, con la proprietà che per ogni soluzione $x$ del problema originario, $\sum_i x_i^* \geq \sum_i x_i$. A questo punto la soluzione $x^*$ viene trasformata in una ammissibile per il problema originario arrotondando i valori frazionari in maniera opportuna. Nel caso del vertex cover, si pone $y_i := 1$ quando $x_i^* \geq 1/2$, e $y_i := 0$ altrimenti. La soluzione $y$ così ottenuta verifica i vincoli della formulazione PLI poiché $x^*$ verifica i vincoli del rilassamento. Inoltre si verifica facilmente che $\sum_i y_i \leq \sum_i 2x_i^* \leq 2\sum_i x_i$ per ogni soluzione $x$ ammissibile per PLI. L'algoritmo restituisce quindi una soluzione che ha un fattore di approssimazione pari al più a 2.

Un secondo gruppo di algoritmi basati sulla programmazione lineare fa leva, anziché sull'arrotondamento delle soluzioni frazionarie, sulla teoria della dualità (algoritmi primale-duale). Spesso infatti il duale di un rilassamento lineare di un problema ha anche una interpretazione intuitiva che può essere sfruttata per il progetto di un algoritmo. Algoritmi di questo tipo operano in genere mantenendo una soluzione valida per il duale, che viene di passo in passo migliorata mentre al tempo stesso, sulla base dei passi intrapresi, si costruisce una soluzione valida per il problema originale. La relazione di dualità tra le due soluzioni permette di stabilire un vincolo superiore al rapporto di approssimazione dell'algoritmo. Un vantaggio di questo approccio è che sebbene l'analisi si basi

Primale-duale

su variabili di tipo continuo, l'algoritmo può spesso essere riformulato in maniera puramente combinatoria. Si evita così di dover invocare un risolutore di programmi lineari, che talvolta rappresenta un collo di bottiglia degli algoritmi basati sull'arrotondamento. Lo schema primale-duale ha trovato applicazione nell'analisi di algoritmi per problemi di progetto di reti, quali l'albero di Steiner [Goemans e Williamson, SICOMP 1995].

Lower bound e PCP

**Limiti inferiori.** Parallelamente al progetto e all'analisi di algoritmi di approssimazione, un'altra linea di ricerca ha investigato i limiti di approssimabilità di problemi fondamentali, sotto l'ipotesi P $\neq$ NP. Inizialmente i risultati in questo campo sono stati pochi ed isolati [Sahni e Gonzalez JACM 1976], ma lo scenario è radicalmente cambiato con l'introduzione di una nuova caratterizzazione della classe NP dovuta a vari autori [Arora e Safra, FOCS 1992; Arora ed altri, FOCS 1992]. Questa importante caratterizzazione ha permesso di provare limiti inferiori per un gran numero di problemi fondamentali. Ad esempio, è stato mostrato che nessun algoritmo polinomiale può approssimare il problema della copertura di insieme con un rapporto inferiore a $\ln n$ e che quindi l'algoritmo goloso (analizzato da Johnson) è ottimale a meno di fattori costanti [Feige, JACM 1998]. Infine, negli ultimi anni sono state sviluppate tecniche che hanno permesso di dimostrare risultati di inapprossimabilità ancora più stringenti per alcuni problemi, sebbene sulla base di ipotesi più forti di P $\neq$ NP [Khot ed altri, FOCS 2004].

# 3 Tendenze di ricerca

Tendenze: programmazione matematica

Una tendenza recente della ricerca nel campo degli algoritmi approssimati è stata quella di passare a poco a poco dalle analisi basate sulla programmazione lineare ad analisi basate su tecniche più sofisticate di programmazione matematica. Ci limitiamo a due esempi: la programmazione semidefinita e i rilassamenti Lagrangiani.

Max Cut

**Programmazione semidefinita.** La programmazione semidefinita è una generalizzazione della programmazione lineare che talvolta permette di fornire rilassamenti più stretti di problemi di ottimizzazione combinatoria. Un importante risultato di Goemans e Williamson [JACM 1995] ha mostrato come essa possa essere sfruttata per analizzare problemi in apparenza puramente combinatori quali la determinazione del massimo taglio in un grafo. L'elegante algoritmo di Goemans e Williamson ha migliorato il limite di errore (in difetto) dal 50% al 13% ed ha rappresentato un nuovo punto di partenza per il progetto di algoritmi approssimati per problemi di partizionamento di grafi e di soddisfacibilità proposizionale.

Un ulteriore esempio in proposito è rappresentato dal lavoro di Skutella [JACM 2001] che ha studiato vari problemi di *scheduling* usando formulazioni basate sulla programmazione quadratica.

**Rilassamento Lagrangiano.** Il rilassamento Lagrangiano è una tecnica classica dell'ottimizzazione con la quale si cerca di semplificare un problema di programmazione matematica trasformandone uno dei vincoli in un termine che confluisce nella funzione obiettivo. Più precisamente, si cerca di approssimare il seguente problema (P1) risolvendo (o approssimando) il problema (P2). Qui $\mathcal{F}$ è l'insieme delle soluzioni ammissibili, $w$ e $c$ sono due funzioni di costo, e $B$ un numero razionale.

Lagrangiano in generale

$$\max_S \ w(S) \ \text{ tale che } S \in \mathcal{F} \text{ e } c(S) \leq B. \tag{P1}$$

$$\min_{\lambda \geq 0} \max_S \ w(S) + \lambda(B - c(S)) \ \text{ tale che } S \in \mathcal{F}. \tag{P2}$$

Poiché il valore ottimo di (P1) è sempre limitato superiormente dal valore ottimo di (P2), per approssimare il problema (P1) è "sufficiente" trovare una soluzione ottima di (P2) e convertirla in una soluzione di qualità approssimativamente simile che sia anche valida per (P1). Questo schema è stato usato per problemi di ottimizzazione NP-ardui con vincoli globali, quali il minimo albero di copertura su $k$ nodi [Garg, FOCS 1996], il minimo albero di copertura con vincolo di spesa [Ravi e Goemans, SWAT 1996] e la localizzazione a minimo costo di $k$ impianti ($k$-*median*) [Jain e Vazirani, JACM 2001]. Una interessante possibile direzione di ricerca consiste nel determinare se questo approccio possa essere esteso ad altri problemi in cui il rilassamento (P2) ha una struttura combinatoria più complessa; si veda ad esempio [Berger ed altri, MP 2010].

$k$-MST, $k$-median

# 4 Ricadute applicative

Come sottolineato nell'introduzione, il ruolo della ricerca nel campo degli algoritmi di approssimazione non è tanto quello di progettare direttamente algoritmi rivolti a particolari applicazioni pratiche, quanto quello di spiegare rigorosamente perché alcuni problemi di ottimizzazione sono più facilmente attaccabili di altri e di fornire tecniche algoritmiche che, opportunamente ingegnerizzate, trovino poi utilizzo nelle applicazioni.

Si consideri ad esempio il problema del TSP Euclideo, ovvero il problema del commesso viaggiatore nel caso in cui le città da visitare corrispondano a punti in un piano Euclideo. Si tratta di un problema con le più svariate applicazioni, dal controllo di processi di fabbricazione di circuiti integrati al posizionamento di sensori per la cristallografia a raggi X. Da un lato, è noto da tempo che anche questo caso speciale del TSP resta NP-arduo [Papadimitriou, TCS 1977]. D'altra parte, le costruzioni che mostrano l'inapprossimabilità del TSP con funzioni distanza arbitrarie non si applicano al caso Euclideo. Un importante risultato è stato quello di Arora [JACM 1998], che ha dimostrato come nonostante i risultati di hardness, il problema ammette uno schema di approssimazione polinomiale; l'intrattabilità del problema si è quindi rivelata "fragile" dal punto di vista delle soluzioni approssimate. Questo risultato ha fornito una spiegazione rigorosa del perché in pratica si riescano a risolvere con ottima approssimazione

Il TSP Euclideo

istanze del TSP Euclideo di grande dimensione. Uno dei solutori del TSP più noti, ad esempio, dal nome Concorde [Applegate ed altri, 2006], è stato usato per risolvere istanze reali da decine di migliaia di nodi con errore minore dello 0.1%. Sebbene le tecniche algoritmiche usate da Concorde siano abbastanza diverse da quelle dell'algoritmo di Arora, non c'è dubbio che esse facciano leva su una struttura combinatoria che l'analisi di Arora ha contribuito a chiarire. L'analisi rigorosa dell'efficacia di solutori quali Concorde rimane un importante problema aperto.

**Monografie citate.**


E. Lawler e altri. *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization.* Wiley, 1985.

R.M. Karp. Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (eds.), *Complexity of Computer Computations.* Plenum, 1972.

R. Applegate e altri. *The Traveling Salesman Problem: a Computational Study.* Princeton University Press, 2006.


**Acronimi delle riviste e conferenze citate.**

JACM: Journal of the Association for Computing Machinery

JCSS: Journal of Computer and System Sciences

MP: Mathematical Programming

SICOMP: SIAM Journal on Computing

TCS: Theoretical Computer Science

FOCS: IEEE Conference on Foundations of Computer Science

STOC: ACM Symposium on the Theory of Computation

SWAT: Scandinavian Workshop on Algorithm Theory