PARTE II

ALGORITMI APPROSSIMATI PER PROBLEMI DI OTTIMIZZAZIONE NP-HARD

Complessità di problemi di ottimizzazione

Algoritmi approssimati e schemi di approssimazione

Tecniche algoritmiche

Classi di approssimabilità

Limiti alla approssimabilità

COMPLESSITA' DI PROBLEMI DI OTTIMIZZAZIONE

Problema di ottimizzazione
P=<I, S, m, opt>:

I: istanze del problema
S: spazio di ricerca; data $x \in I$, S(x) indica le soluzioni ammissibili di x
m: misura definita per coppie (x,y) tali che $x \in I$, $y \in S(x)$
opt: tipo di problema (max, min)

Tre problemi:

Valore della soluzione ottima: data $x \in I$, il valore della soluzione ottima è
$m^*(x) = opt \{m(x,s) \mid s \in S(x)\}$.

Soluzione ottima: data $x \in I$ una soluzione ottima è una qualunque soluzione $s \in S^*(x) = \{s \in S(x) \mid m(x,s) = m^*(x)\}$.

Problema di decisione: data $x \in I$ e $k \in N$ decidere se esiste s tale che $m(x,s) \leq k$ (se opt=min) o $m(x,s) \geq k$ (se opt=max).

Esempio: il problema del commesso viaggiatore
(Traveling Salesman Problem: MIN-TSP).

I: grafi completi con pesi sugli archi
S: dato $G \in I$, S(G) = insieme dei possibili cicli Hamiltoniani su G
(permutazioni dei nodi di G che individuano un percorso che visita esattamente una volta ogni vertice e torna al vertice di partenza)
m: somma dei pesi degli archi appartenenti al ciclo $<p_0, p_1, \ldots, p_{n-1}, p_0>$
opt: min

Tre problemi:
- dato G trovare il valore del ciclo di costo minimo
- dato G trovare un ciclo di costo minimo
- dato G e k decidere se esiste un ciclo di costo $\leq k$.

Classe di problemi di ottimizzazione NPO

Un problema di ottimizzazione P=<I, S, m, opt> appartiene alla classe NPO se:

- l'insieme delle istanze I è decidibile in tempo polinomiale,
- esiste un polinomio q tale che data un'istanza $x \in I$, per ogni $s \in S(x)$, $|s| \leq q(|x|)$ e inoltre, per ogni stringa $y \leq q(|x|)$, è decidibile in tempo polinomiale se $y \in S(x)$
- la misura m è calcolabile in tempo polinomiale.

La classe NPO è una classe particolarmente importante di problemi di ottimizzazione. Essa costituisce l'analogo della classe NP per i problemi di decisione. La quasi totalità dei problemi di ottimizzazione che interessano in pratica appartengono a NPO.

Ogni problema di decisione associato a un problema in NPO è in NP.

Esempi.

MIN-PATH: cammino minimo tra due nodi
MIN-SPANNING –TREE: minimo albero ricoprente in un grafo
MAX-MATCHING: accoppiamento di nodi di valore massimo
MAX-FLOW: flusso massimo in un grafo orientato
MAX-PL: programmazione lineare sui reali
MAX-PL(0,1): programmazione lineare con variabili 0-1
MAX-PLI: programmazione lineare con variabili intere
MIN-TSP: problema del commesso viaggiatore
MIN-VERTEX-COVER: minimo ricoprimento dei nodi di un grafo
MAX-SAT: massimo sottoinsieme di clausole soddisfacibili

Per analizzare la complessità dei problemi di ottimizzazione dobbiamo utilizzare una riducibilità più potente della Karp-riducibilità.

Sia dato il problema di calcolare il valore di una funzione (eventualmente a più valori) $f: I \rightarrow S$.

Un oracolo è un dispositivo astratto che data $x \in I$ restituisce in un unico passo di calcolo il valore $y=f(x)$ (o, nel caso di funzione a più valori, un valore $y \in f(x)$).

Sia dato il problema P1 di calcolare una funzione (eventualmente a più valori) $f: I \rightarrow S$ il problema si dice Turing-riducibile ad un problema P2 se esiste un algoritmo R che risolve P1 interrogando un oracolo per P2. In tal caso R viene chiamata Turing-riduzione da P1 a P2 e scriviamo

$P1 \leq_T P2$.

Una Turing-riduzione è polinomiale se R opera in tempo polinomiale (in tal caso scriviamo $P1 \leq_{T\ p} P2$).

Una Turing-riduzione polinomiale da P1 a P2 può interrogare un oracolo per P2 un numero polinomiale di volte.

Si noti che mediante Turing riduzioni la differenza tra NP e co-NP scompare, infatti ogni linguaggio in NP può essere ridotto al proprio complemento in co-NP e viceversa. In altre parole per decidere se $x \in S$ possiamo utilizzare il seguente algoritmo R:

- l'algoritmo chiede ad un oracolo se $x \in S'$ (dove S' è il complemento di S)
- se l'oracolo risponde SI l'algoritmo R restituisce NO e viceversa.

Una Karp-riduzione è un caso particolare di Turing-riduzione in cui
- i due problemi sono problemi di decisione
- l'oracolo per P2 è interrogato una sola volta
- R restituisce lo stesso valore fornito dall'oracolo.

Un problema di ottimizzazione P1 è NP-hard se esiste un problema NP-completo Turing-riducibile in tempo polinomiale a P1.

Esempio.

Il problema MIN-TSP è NP-hard. Sappiamo che il problema di decisione TSP è NP-completo. Possiamo dimostrare che TSP $\leq_{T\,p}$ MIN-TSP. Infatti se avessimo a disposizione un oracolo per il problema di

ottimizzazione ne potremmo derivare una soluzione per il problema di decisione.

Analizziamo la relazione di complessità che sussiste tra i tre problemi associati a un problema di ottimizzazione in NPO.

Dato un problema di ottimizzazione P, chiamiamo P_E il problema di valutazione dell'ottimo, P_O il problema del calcolo di una soluzione ottima, P_D il problema di decisione.

Chiaramente abbiamo:

$P_D \leq_{T\,p} P_E$ e $P_E \leq_{T\,p} P_O$

e quindi se risolviamo in tempo polinomiale il problema P_O possiamo risolvere in tempo polinomiale i problemi P_E e P_D .

Nella direzione opposta abbiamo:

Teorema. $P_E \leq_{T\,p} P_D$

Dimostrazione. La dimostrazione è basata sull'uso della ricerca logaritmica.
Data un'istanza x ed una qualunque soluzione ammissibile s, se M è l'ambito dei valori possibili della funzione m(x,s), poiché tale funzione è calcolabile in tempo polinomiale abbiamo che $M \leq 2^{p(|x|,|s|)}$
Poiché abbiamo anche che esiste un polinomio q tale che $|s| \leq q(|x|)$, ne consegue che $m^*(x) \leq M \leq 2^{p'(|x|)}$ per un opportuno polinomio p'.
Possiamo quindi risolvere il problema di valutazione interrogando

log M = p’(|x|) volte un oracolo per il problema di decisione. Il costo di risoluzione del problema P_E è O(p’(|x|) · costo di risoluzione di P_D).
QED

Per quanto riguarda la relazione tra P_O e P_E un risultato generale del tipo:

$P_O \leq_{T\,p} P_E$

per ogni problema in NPO non è noto ma in molti casi particolari la conoscenza del valore delle soluzioni ottime consente di trovarne una e quindi di risolvere il problema costruttivo.

Si noti che, in generale, non sappiamo se esistono problemi di ottimizzazione in NPO per cui non è vero che $P_O \leq_{T\,p} P_E$.

MAX-CLIQUE-SIZE sia il problema di valutazione associato al problema di ottimizzazione MAX-CLIQUE.

MAX-CLIQUE $\leq_{T\,p}$ MAX-CLIQUE-SIZE

Dimostrazione

Dato un grafo G=<V,E>, sia Gn il sottografo di G indotto da n e da tutti i nodi ad esso adiacenti e sia Gˆn il sottografo indotto solo dai nodi adiacenti ad n.

L'algoritmo Max-clique (G) opera nel seguente modo.

- Interroga l'oracolo per MAX-CLIQUE-SIZE con input G.
- Per ogni v∈V
  - interroga l'oracolo per MAX-CLIQUE-SIZE con input Gv,

- se MAX-CLIQUE-SIZE (Gv) = MAX-CLIQUE-SIZE (G) allora una soluzione ottima è $\{v\} \cup$Max-clique(G^v)

Chiaramente la riduzione di Turing opera in tempo polinomiale. Infatti il numero di chiamate all'oracolo è

$T(n) = T(n-1) + (n+1)$

la cui soluzione è $T(n) = O(n^2)$

Enunciamo senza dimostrazione il seguente risultato:

Dato un problema P in NPO, se P_D è NP-completo, allora $P_O \leq_{T\,p} P_E$.

Molti problemi di ottimizzazione in NPO sono in realtà risolubili in tempo polinomiale.

Classe di problemi di ottimizzazione PO

Un problema di ottimizzazione $P = \langle I, S, m, opt \rangle \in NPO$ appartiene alla classe PO se esiste un algoritmo che data un'istanza $x \in I$ determina una soluzione ottima $s \in S^*(x)$ in tempo polinomiale.

Esempi.

MIN-PATH
MIN-SPANNING TREE
MAX-MATCHING

MAX-FLOW
MAX-PL

In base alle Turing-riduzioni viste precedentemente (ad esempio la riduzione TSP $\leq_{T\,p}$ MIN-TSP) è semplice verificare che, essendo TSP NP-completo:

se PO = NPO allora P = NP.

In conclusione, molti problemi in NPO non ammettono, probabilmente, algoritmi polinomiali. Per tali problemi dobbiamo ricorrere, se possibile, ad algoritmi in grado di calcolare efficientemente soluzioni approssimate.

ALGORITMI DI APPROSSIMAZIONE

Dato un problema di ottimizzazione P=<I, S, m, opt> un algoritmo di approssimazione è un algoritmo A che data un'istanza $x \in I$, fornisce una soluzione $A(x) = y \in S(x)$.

Errore relativo:

$$E(x,y) = \frac{|m^*(x)-m(x,y)|}{\max\{m^*(x), m(x, y)\}}$$

Dato un problema di ottimizzazione P, diciamo che un algoritmo di approssimazione A è ε-approssimato se per ogni istanza $x \in I$, abbiamo

$E(x, A(x)) \leq \varepsilon$, dove $0 \leq \varepsilon \leq 1$.

In alternativa all'errore relativo possiamo considerare il rapporto di approssimazione

$$R(x,y) = \max\left(\frac{m(x,y)}{m^*(x)}, \frac{m^*(x)}{m(x, y)}\right)$$

Dato un problema di ottimizzazione P, diciamo che un algoritmo di approssimazione A è r-approssimato se per ogni istanza $x \in I$, abbiamo $R(x, A(x)) \leq r$ dove $r \geq 1$. Si noti che:

$E(x, y) = 1 - 1/R(x, y)$ e, viceversa, $R(x, y) = 1/(1-E(x, y))$.

Spesso può essere più intuitivo usare sempre il rapporto $m(x, y) / m^*(x)$ che risulta ≤ 1 per problemi di massimizzazione e ≥ 1 per problemi di minimizzazione.

Chiaramente un algoritmo di approssimazione A per un problema di ottimizzazione P NP-hard è interessante se, data un'istanza x di P, esso fornisce efficientemente, cioè in tempo polinomiale in $|x|$, una soluzione approssimata $A(x)$.

In tal caso diciamo che A è un algoritmo di approssimazione polinomiale e che P è un problema polinomialmente approssimabile.

Esempio. Si consideri il seguente algoritmo per MAX-SAT:

ALGORITMO GREEDY SAT
(Tecnica greedy)

Data una formula $w=\{c_1, c_2, ..., c_m\}$ in CNF:
ripeti
1. scegliere un letterale l tra quelli che occorrono il massimo numero di volte (affermato o negato);
2. se l=x porre x=V
3. se l=¬x porre x=F
4. cancellare le clausole in cui occorre l e cancellare ¬l dalle altre

finché w è vuota.

Teorema L'algoritmo Greedy Sat è 2-approssimato.

Dimostrazione. Per induzione sul numero n di variabili. Supponiamo di avere m clausole.

Passo base: se n=1 il risultato è vero.
Ipotesi induttiva: supponiamo che l'enunciato del teorema sia soddisfatto nel caso di n-1 variabili ed m' clausole. L'algoritmo garantisce dunque il soddisfacimento di almeno m'/2 clausole.
Supponiamo ora di avere n variabili.
Sia l_v il letterale che compare più volte e supponiamo $l_v = v$.
Supponiamo che v compaia c_1 volte e $\neg v$ compaia c_2 volte con $c_1 \geq c_2$.
Assegnando a v valore Vero soddisfiamo c_1 clausole. Ora abbiamo una formula con n-1 variabili ed $m' \geq m - c_1 - c_2$ clausole. Per ipotesi induttiva

di queste ne possiamo soddisfare almeno m'/2 e quindi in tutto possiamo soddisfare $c_1+m'/2 \geq c_1+(m-c_1-c_2)/2 \geq m/2$ clausole. QED

Un altro esempio di algoritmoa approssimato molto molto emplice e' il seguente algoritmo per MIN-VERTEX-COVER (algoritmo di Gavril, detto anche algoritmo basato sul matching):

Dato un grafo G = <V, E> procedi come segue fino a che E non e' vuoto:

- scegli un arco (i, j) in E
- inserisci i vertici i e j nella soluzione corrente S
- elimina dal grafo i vertici i e j e tutti gli archi in essi incidenti

Stampa S.

Non è difficile verificare che S ha al più cardinalita doppia rispetto all'ottimo perché ogni soluzione ottima deve contenere o il vertice i o il

vertice j o entrambi. Si osservi anche che l'insieme degli archi via via prescelti costituisce un matching sul grafo dato.

Altri esempi di problemi che ammettono algoritmi approssimati:

- MAX CUT
- MIN SCHEDULING (su macchine uguali)
- MIN BIN PACKING
- MIN TSP METRICO

ecc.

SCHEMI DI APPROSSIMAZIONE

Un algoritmo A(x, ε) si chiama schema di approssimazione polinomiale (PTAS) per un problema di ottimizzazione P se per ogni valore ε esiste un polinomio q tale che, per ogni istanza x di P l'algoritmo restituisce una soluzione approssimata y tale che $E(x, y) \leq \varepsilon$, in tempo limitato da q(|x|). Si noti che in questo caso il grado del polinomio può dipendere da ε.

Un algoritmo A(x, ε) si chiama schema di approssimazione pienamente polinomiale (FPTAS) per un problema di ottimizzazione P se esiste un polinomio q tale che, per ogni istanza x di P e per ogni valore ε l'algoritmo restituisce una soluzione approssimata y tale che $E(x, y) \leq \varepsilon$, in tempo limitato da $q(|x|, 1/\varepsilon)$.

Definizioni analoghe possono essere date con riferimento al rapporto di approssimazione anziché all'errore relativo.
Esempio di schema di approssimazione.

MIN PARTITION: dato un insieme di oggetti con pesi $a_1, \ldots, a_n$, individuare una partizione in due insiemi A1 e A2 tali che sia minimizzato il valore $\max \{\Sigma_{A1} a_i, \Sigma_{A2} a_i\}$.

Il seguente algoritmo è uno schema di approssimazione polinomiale per MIN PARTITION.

ALGORITMO MIN PARTITION
(Tecnica di completamento)
Data un'istanza I del problema costituita da un insieme di oggetti con pesi $a_1, \ldots, a_n$, e un valore $r > 1$:

- ordinare gli oggetti in ordine di peso non crescente;
- definire $k=\lceil (2-r)/(r-1) \rceil$ (ad esempio: se r=11/10, k=9);
- individuare una partizione ottima dei primi k elementi;
- inserire i rimanenti elementi via via nell'insieme di peso minore.

Dimostrazione. Assumiamo che:

- $a_1, \ldots, a_n$ siano i pesi degli oggetti ordinati in modo non crescente,
- A1 ed A2 siano gli insiemi creati dall'algoritmo, w(A1) e w(A2) siano i loro pesi e w(A1)>w(A2),
- m*(I) sia il valore della soluzione ottima,
- sia $L=(w(A1)+w(A2))/2 \leq m^*(I)$
- sia $r < 2$,
- l'ultimo oggetto inserito in A1 sia l'l-esimo con $l > k$ (quindi solo parte degli oggetti sono stati distribuiti in modo ottimo).

Abbiamo dunque:

$w(A1) - a_l \leq w(A2) = 2L - w(A1)$

e quindi $w(A1) - L \leq a_1/2$.
Inoltre $a_1(k+1) \leq 2L$ e quindi $a_1/2L \leq 1/(k+1)$.

$w(A1)/m^*(I) \leq w(A1)/L \leq 1 + a_1/2L \leq$
$\leq 1 + 1/(k+1) \leq 1 + 1/((2-r)/(r-1)+1) \leq r$.

Per quanto riguarda il tempo di esecuzione abbiamo:

- $O(n \log n)$ per ordinare gli oggetti,
- $O(2^k)$ per effettuare il partizionamento ottimo dei primi k oggetti,
- $O(n)$ per inserire gli ultimi $n - k$ oggetti.

In totale $O(n \log n + 2^k)$ QED

Come si vede il costo è polinomiale se si assume k fissato, altrimenti sarebbe esponenziale in k (sostanzialmente, esponenziale in 1/r).

Altri esempi di problemi che ammettono schemi di approssimazione polinomiali sono:

- MAX INDEPENDENT SET SU GRAFI PLANARI
- TSP EUCLIDEO
- MAX INTEGER KNAPSACK
- alcune varianti di problemi di scheduling

ecc.

Esempio di schema di approssimazione pienamente polinomiale.

Per fornire un esempio di problema che ammette uno schema di approssimazione polinomiale pieno, dobbiamo prima introdurre il problema dello 'knapsack' (bisaccia) ed illustrare l'algoritmo per la sua soluzione ottima.

MAX KNAPSACK: dato un insieme I di oggetti dotati di pesi interi a_1, …, a_n e di profitti interi p_1, …, p_n e dato un intero b (capacità), individuare un sottoinsieme S di I tale da massimizzare $\Sigma_S p_i$ con il vincolo $\Sigma_S a_i \leq b$.

Data un'istanza I del problema MAX KNAPSACK il seguente algoritmo, basato sulla tecnica di programmazione dinamica, fornisce la soluzione esatta in tempo $O(n^2 p_{MAX})$.

ALGORITMO BISACCIA
(Tecnica di programmazione dinamica)

1. Sia $M_0 = \{(\varnothing, 0)\}$;
2. for i = 1 to n do
   - $M_i = \varnothing$;
   - per ogni elemento $(S,p) \in M_{i-1}$ inserire in M_i (S,p) e, se $\Sigma_S a_j + a_i \leq b$, anche $(S \cup \{i\}, p+p_i)$
   - (test di dominanza) se (S,p) e $(S',p) \in M_i$ allora, se $\Sigma_S a_j \leq \Sigma_{S'} a_j$ tieni (S,p) ed elimina (S',p), altrimenti tieni (S',p) ed elimina (S,p);
3. stampa $(S,p) \in M_n$ tale che p è massimo.

Poiché per ogni i $|M_i| \leq n\ p_{MAX}$ il tempo di esecuzione è
$O(n\ |M_n|) = O(n^2\ p_{MAX})$. QED

Nota bene. Tempo di esecuzione pseudopolinomiale: polinomiale non nella taglia dell'input ma nei valori che compaiono nell'input!
Il seguente algoritmo è uno schema di approssimazione pienamente polinomiale.

ALGORITMO APPROX-BISACCIA
(Tecnica di scalatura, o partizionamento fisso)
Data un'istanza I del problema MAX KNAPSACK e dato un valore $\varepsilon \leq 1$:

1. sia $k = \varepsilon p_{MAX}/n$;
2. creare una nuova istanza I' con pesi $a_1, \ldots, a_n$, e profitti $p'_1, \ldots, p'_n$ con $p'_i = p_i/k$;
3. risolvere I' in modo esatto con l'algoritmo precedente;
4. determinare $m_A(I) = k\, m(I')$.

Dimostrazione.
Errore compiuto ad ogni passo $\leq k$.
Errore totale $\leq n\,k$.
Errore relativo $\leq n\,k/m^*(I) \leq n\,k/\,p_{MAX} \leq \varepsilon$.
Tempo di esecuzione $O(n^2 p'_{MAX}) = O(n^2 p_{MAX}/k) = O(n^3/\varepsilon)$. QED

Altri esempi di problemi che ammettono schemi di approssimazione pienamente polinomiali sono:

- MIN KNAPSACK,
- PRODUCT KNAPSACK,
- alcune varianti di problemi di scheduling,

ecc.

CLASSI DI APPROSSIMABILITA'

APX: classe di problemi di ottimizzazione in NPO che ammettono un algoritmo polinomiale r-approssimato per un opportuno r.

PTAS: classe di problemi di ottimizzazione in NPO che ammettono uno schema di approssimazione polinomiale.

FPTAS: classe di problemi di ottimizzazione in NPO che ammettono uno schema di approssimazione pienamente polinomiale.

E' possibile dimostrare che, se P≠NP, allora NPO ⊃ APX ⊃ PTAS ⊃ FPTAS ⊃ PO

PRIMI RISULTATI DI NON APPROSSIMABILITA':
LA TECNICA DEL GAP

P: problema di minimizzazione
L: un linguaggio NP-completo

Se esistono due funzioni f e c tali che
f trasforma istanze del problema di decisione associato ad L in istanze di P e, per un opportuno valore 'gap':
- se $x \in L$ allora $m^*(f(x)) \leq c(x)$
- se $x \notin L$ allora $m^*(f(x)) \geq c(x)(1+gap)$.

allora per P non può esistere alcun algoritmo r-approssimato con $r<(1+gap)$ a meno che non sia P=NP.

DIMOSTRAZIONE. Supponiamo per assurdo che A sia un algoritmo r-approssimato per P e che $r<1+gap$. Per decidere in tempo polinomiale se $x \in L$ o meno possiamo proceder nel seguente modo. Calcoliamo l'istanza f(x) del problema P e poi applichiamo l'algoritmo approssimato A ad f(x).

Se abbiamo $m(f(x), A(f(x))) \geq c(x)(1+gap)$ ciò implica $m^*(f(x)) > c(x)$ e quindi $x \notin L$. Infatti abbiamo:
$(1+gap)\, m^*(f(x)) > r\, m^*(x) \geq m(f(x), A(f(x))) \geq c(x)(1+gap)$
Al contrario, se $m(f(x), A(f(x))) < c(x)(1+gap)$ a maggior ragione $m^*(f(x)) < c(x)(1+gap)$ e quindi $x \in L$.

QED

Il problema MIN TSP non è r-approssimabile per alcun r (a meno che non sia P=NP).

DIMOSTRAZIONE. Applichiamo la tecnica del gap. Consideriamo come problema NP-completo il problema GRAFO HAMILTONIANO.
Sia $g>0$ un valore reale arbitrario.
Dato un grafo $G=(V,E)$ costruiamo un grafo $G'=(V, VxV)$ in cui, dati due nodi i e j,
$d(i,j)$ = se $(i,j)\in E$ allora 1 altrimenti $1+ng$
Chiaramente:
- G Hamiltoniano $\rightarrow m^*(G') = n$
- G non Hamiltoniano $\rightarrow m^*(G') \geq n(1+g)$.

Quindi, per quanto grande sia il valore di g, non possiamo avere alcun algoritmo (1+g)-approssimato per il problema MIN TSP.
QED

COROLLARIO. Se P≠NP, APX ⊂ NPO.

MIN BIN PACKING: dato un insieme di n oggetti con pesi interi a_1, …, a_n ed un intero B determinare il minimo numero di contenitori di capacità B che possono contenere gli n oggetti.

Il problema MIN BIN PACKING non ammette alcun algoritmo r-approssimato con r<3/2 (a meno che non sia P=NP) e quindi non ammette un PTAS.

DIMOSTRAZIONE. Applichiamo la tecnica del gap. Assumiamo come problema di decisione NP-completo il problema del partizionamento esatto: dati n oggetti con pesi a_1, …, a_n decidere se essi possono essere

suddivisi in due insiemi A1, A2 tali che w(A1)=w(A2). Data un'istanza I del problema del partizionamento esatto possiamo costruire un'istanza I' di MIN BIN PACKING costituita dagli stessi n oggetti con pesi a_1, …, a_n e dal valore $B=\Sigma a_i/2$.

I istanza positiva di partizionamento esatto
→ $m^*(I') = 2$
I istanza negativa di partizionamento esatto
→ $m^*(I') = 3 = 2(1+g)$ (con $g=1/2$)

Quindi, se P≠NP non può esistere un algoritmo polinomiale r-approssimato per il problema MIN BIN PACKING con $r<3/2$ e pertanto il problema stesso non ammette un PTAS. QED

COROLLARIO. Se P≠NP, PTAS $\subset$ APX.

Successivamente vedremo che MIN BIN PACKING ammette una forma di PTAS più debole, chiamato PTAS asintotico.

PROBLEMI FORTEMENTE NP-HARD E ALGORITMI PSEUDOPOLINOMIALI

Data un'istanza x di un problema P=<I, S, m, opt> definiamo MAX(x) il massimo valore numerico che compare nell'istanza del problema.

Esempio.
Nel caso del problema MAX KNAPSACK, se l'istanza è costituita da un insieme di oggetti dotati di pesi interi $a_1, \ldots, a_n$ e di valori interi $p_1, \ldots, p_n$ e da un intero b, avremo $MAX(x)=\max\{p_1, \ldots, p_n, b\}$.

Un algoritmo A per un problema di ottimizzazione P=<I, S, m, opt> in NPO si definisce pseudopolinomiale se, esiste un polinomio p tale che, data un'istanza x∈I, abbiamo che il tempo di esecuzione di A(x) è ≤p(|x|, MAX(x)).

Esempio.
Il problema MAX KNAPSACK ammette un algoritmo pseudopolinomiale poiché si risolve in tempo $O(n^2 p_{MAX}) \leq O(n^2 MAX(x))$.

Se un problema P=<I, S, m, opt> in NPO ammette una soluzione esatta in tempo pseudopolinomiale p(|x|, MAX(x)), allora dato un qualunque polinomio q, evidentemente, il sottoproblema P'=<I', S, m, opt> in cui

$I' = \{x | x \in I \text{ e } MAX(x) \leq q(|x|)\}$ si risolve in tempo polinomiale $p(|x|, q(|x|))$.

Un problema P=<I, S, m, opt> in NPO si definisce fortemente NP-hard se esiste un polinomio q tale che il problema P'=<I', S, m, opt> in cui I' = {x|x∈I e MAX(x) ≤ q(|x|)} è NP-hard.

Esempio.
Il problema MAX SAT PESATO, in cui ad ogni clausola è associato un peso intero e l'obiettivo è massimizzare la somma dei pesi delle clausole soddisfatte, è fortemente NP-hard poiché la versione non pesata è essa stessa NP-hard.

Se un problema ammette una soluzione pseudopolinomiale esso non è fortemente NP-hard. Come abbiamo visto nel caso del problema MAX KNAPSACK, dato un qualunque polinomio q, da una soluzione

pseudopolinomiale possiamo ottenere una soluzione polinomiale per il sottoproblema in cui $MAX(x) \leq q(|x|)$.

Dato un problema $P=<I, S, m, opt>$ in NPO per cui esiste un polinomio q tale che per ogni x, $m^*(x) \leq q(|x|, MAX(x))$, se esiste un FPTAS per P allora P ammette un algoritmo pseudopolinomiale per la soluzione esatta.

DIMOSTRAZIONE. Supponiamo che $T(x,\varepsilon)$ sia un FPTAS che su un'istanza x di P fornisce una soluzione ε-approssimata in tempo $p(|x|, 1/\varepsilon)$. Possiamo risolvere P scegliendo
$\varepsilon \leq 1/(q(|x|,MAX(x))+1) \leq 1/(m^*(x)+1)$.
Il tempo di esecuzione sarà
$p(|x|, q(|x|,MAX(x)))$ e quindi pseudopolinomiale. QED

COROLLARIO I. Un problema P=<I, S, m, opt> in NPO fortemente NP-hard per il quale $m^*(x) \leq q(|x|, MAX(x))$ non ammette un FPTAS (se $P \neq NP$).

Un problema di ottimizzazione in NPO P=<I, S, m, opt> è polinomialmente limitato se esiste un polinomio q tale che
$\forall x \in I \; \forall y \in S(x)[m(x,y) \leq q(|x|)]$

Esempi.
Il problema MAX SAT è polinomialmente limitato.
Il problema MAX SAT PESATO in cui ad ogni clausola è associato un peso intero ed il problema consiste nel massimizzare la somma dei pesi delle clausole soddisfatte, non è polinomialmente limitato.

COROLLARIO II. Se un problema NP-hard è polinomialmente limitato esso non ammette uno schema di approssimazione pienamente polinomiale (se P≠NP).

Come vedremo in seguito il problema MAX INDEPENDENT SET SU GRAFI PLANARI ammette un PTAS ma, essendo polinomialmente limitato non può ammettere un FPTAS (a meno che non sia P=NP).

COROLLARIO. Se P≠NP, FPTAS ⊂ PTAS.