Insegnamento di algoritmi avanzati

Algoritmi di approssimazione

Roberto Posenato

ver. 0.5, 01/10/2009

Sommario

Introduzione

In questa lezione si richiamano i concetti fondamentali circa gli algoritmi di approssimazione.

Come affrontare problemi difficili

Quando si deve risolvere un problema NP-completo, essenzialmente ci sono due alternative:

- Verificare se si deve risolvere un particolare sottoinsieme di istanze e, in caso positivo, sviluppare un'euristica per lo scopo (ricerca soluzione esatta per un sottoinsieme delle istanze).
- Cercare di determinare soluzioni algoritmiche in grado di calcolare soluzioni quasi ottimali ma calcolabili in tempi ragionevoli (algoritmi di approssimazione).

Come affrontare problemi difficili

Trovare soluzioni approssimate non è una sconfitta...

Dall'introduzione al corso...

Quali sono le alternative?

- si semplifica il modello in modo da poter usare algoritmi di ottimizzazione noti in grado di fornire soluzioni di qualità.
  Problema reale $\stackrel{1}{\Rightarrow}$ Modello$_a$ $\stackrel{2}{\Rightarrow}$ Soluzione$_p$(Modello$_a$)
- si mantiene il modello e si provano metodi non tradizionali per determinare soluzioni quasi ottimali.
  Problema reale $\stackrel{1}{\Rightarrow}$ Modello$_p$ $\stackrel{2}{\Rightarrow}$ Soluzione$_a$(Modello$_p$)

dove $_a$ indica "approssimato" e $_p$ indica "preciso". In entrambi i casi si trovano soluzioni approssimate!
Alcuni autori riferiscono che spesso il secondo approccio porta a migliori risultati.

Concetti preliminari

Classi di problemi

Gli algoritmi di approssimazione sono interessanti soprattutto per i problemi che hanno la versione di decisione nella classe NP.

Definizione 1 (Classe Nondeterministic Polynomial Optimization)

Un problema di ottimizzazione $\Pi = (I, \mathrm{SOL}(), m(), \mathrm{goal})$ appartiene alla classe NPO se gode delle seguenti proprietà:

- l'insieme I è decidibile in tempo polinomiale;
- esiste un polinomio q tale che, data un'istanza $x \in I$, per qualsiasi $y \in \mathrm{SOL}(x)$ vale che $|y| \leq q(|x|)$; inoltre, per ogni y tale che $|y| \leq q(|x|)$, l'appartenenza a $\mathrm{SOL}(x)$ è decidibile in tempo polinomiale;
- la funzione misura $m()$ è computabile in tempo polinomiale.

Concetti preliminari

Classi di problemi

Anche per la classe NPO è possibile raggruppare i problemi che possono essere risolti in modo efficiente nella sottoclasse PO:

Definizione 2 (Classe Polynomial Optimization)

Un problema di ottimizzazione $\Pi \in$ NPO appartiene alla classe PO se ammette un algoritmo con complessità in tempo polinomiale che, per ogni istanza $x \in I$, determina una soluzione ottimale $y \in \text{SOL}^*(x)$ insieme al suo valore $m^*(x)$.

Esempio 1 (Esempio di problema in PO)

RAGGIUNGIBILITÀ $\in$ PO.

Nota!

È un fatto che praticamente tutti i problemi di ottimizzazione di interesse teorico-applicativo sono nella classe NPO.

Concetti preliminari

Valutazione errore

- Per determinare la qualità di un algoritmo di approssimazione è necessario valutare l'entità dell'errore delle soluzioni calcolate.
- Esistono due misure comuni per valutare la qualità di una soluzione non esatta: l'errore relativo e l'indice di prestazione.

Concetti preliminari

Errore relativo

- Dato un problema di ottimizzazione Π,
- per ogni istanza $x \in I_\Pi$ e per ogni soluzione ammissibile $y \in \mathrm{SOL}_\Pi(x)$,
- l'errore relativo della soluzione y rispetto all'istanza x è la quantità

$$\mathrm{Err}_\Pi(x, y) \stackrel{\text{def}}{=} \frac{|m(x, y) - m^*(x)|}{\max(m(x, y), m^*(x))}$$

Nota!

m è definita solo per valori positivi!

Nota!

$\mathrm{Err}_\Pi(x, y) = 0$ quando la soluzione è esatta, ≈ 1 quando è di pessima qualità.

Concetti preliminari

Indice di prestazione

In alternativa all'errore di approssimazione:

- Dato un problema di ottimizzazione Π,
- per ogni istanza $x \in I_\Pi$ e per ogni soluzione ammissibile $y \in \mathrm{SOL}_\Pi(x)$,
- l'indice di prestazione della soluzione y rispetto all'istanza x è la quantità

$$\mathrm{PR}_\Pi(x, y) \stackrel{\text{def}}{=} \max\left(\frac{m(x,y)}{m^*(x)}, \frac{m^*(x)}{m(x,y)}\right)$$

Nota!

$\mathrm{PR}_\Pi(x, y) = 1$ quando la soluzione è esatta, $>> 1$ quando è di pessima qualità.

Nota!

Chiaramente i due indici sono collegati: $\mathrm{Err}_\Pi(x, y) = 1 - \frac{1}{\mathrm{PR}_\Pi(x,y)}$

Algoritmo di approssimazione

- Dato un algoritmo A che risolve un problema di ottimizzazione Π,
- data una costante $\varepsilon \geq 0$,
- si dice che A è un algoritmo ε-approssimante se e solo per qualsiasi input x, l'errore relativo della soluzione determinata è inferiore alla costante ε:

$$\mathrm{Err}_{\Pi}(x, A(x)) \leq \varepsilon$$

Nota!

Un algoritmo esatto è 0-approssimante.

Algoritmo di approssimazione

Note

Dalla definizione segue che:

1. $0 \leq \varepsilon \leq 1$;
2. per i problemi di massimizzazione, la soluzione trovata da un algoritmo ε-approssimante A soddisfa la relazione:

$$(1 - \varepsilon)m^*(x) \leq m(x, A(x)) \leq m^*(x)$$

3. per i problemi di minimizzazione, la soluzione trovata da una algoritmo ε-approssimante A soddisfa la relazione:

$$m^*(x) \leq m(x, A(x)) \leq \frac{1}{(1 - \varepsilon)} m^*(x)$$

Algoritmo di approssimazione

Definizione alternativa

Se si usa l'indice di performance:

- Dato un algoritmo M che risolve un problema di ottimizzazione Π,
- data una costante $r \geq 1$,
- si dice che M è un algoritmo r-approssimante se e solo per qualsiasi input x, l'indice di prestazione della soluzione determinata è inferiore alla costante r:

$$PR_{\Pi}(x, M(x)) \leq r$$

Nota!

Un algoritmo esatto è 1-approssimante.

Algoritmo di approssimazione

Note

Dalla definizione segue che:

1. $1 \leq r$;
2. per i problemi di massimizzazione, la soluzione trovata da un algoritmo r-approssimante A soddisfa la relazione:

$$\frac{m^*(x)}{r} \leq m(x, A(x)) \leq m^*(x)$$

3. per i problemi di minimizzazione, la soluzione trovata da un algoritmo r-approssimante A soddisfa la relazione:

$$m^*(x) \leq m(x, A(x)) \leq rm^*(x)$$

Algoritmo di approssimazione

Note

- Molti autori preferiscono l'indice di prestazione anziché l'errore relativo.
- Dato che i due indici hanno insiemi di definizione disgiunti, il valore stesso chiarisce quale delle due definizioni si sta considerando.

Esempio 2

Un algoritmo $1/2$-approssimante è definito usando l'errore relativo, mentre un algoritmo 2-approssimante è definito usando l'indice di prestazione.
Un algoritmo $1/2$-approssimante fornisce soluzioni di pari qualità di uno 2-approssimante.

Algoritmo di approssimazione

Problema approssimabile

Dato un problema di ottimizzazione Π, è interessante determinare il più piccolo ε (o r) per il quale *esiste* un algoritmo ε-approssimante (o r-approssimante) polinomiale in tempo per Π.

Definizione 3 (Problema ε-approssimabile)

Un problema di ottimizzazione è detto ε-approssimabile (*r-approssimabile*) se esiste un algoritmo ε-approssimante (r-approssimante) polinomiale in tempo che lo risolve.

Algoritmo di approssimazione

Problema approssimabile

- Anche se alcuni problemi di ottimizzazione in NPO hanno la versione di decisione NP-completa, risultano essere sono molto diversi da un punto di vista dell'approssimabilità.
- Accade infatti che:
  - alcuni problemi sono approssimabili per valori di $r \geq k$, dove $k > 1$ è una costante caratteristica del problema e prende il nome di *soglia di approssimazione*;
  - alcuni problemi sono approssimabili per valori di $r > 1$ arbitrariamente piccoli.
  - alcuni problemi *non* sono approssimabili per alcun r costante;

Esempi di problemi approssimabili

Problema MINIMO RICOPRIMENTO DI VERTICI

Versione di decisione di MINIMO RICOPRIMENTO DI VERTICI in NP $\Rightarrow$ problema è NPO.

Un'euristica generale è costruibile considerando il paradigma greedy applicato sul grado dei vertici.

Algoritmo 1: greedy per MINIMO RICOPRIMENTO DI VERTICI

Input: $G = (V, E)$

1: $C = \emptyset$;
2: **while** $(E \neq \emptyset)$ **do**
3: $\quad v =$ vertice con grado maggiore;
4: $\quad C = C \cup \{v\}$;
5: $\quad V = V \setminus \{v\}, E = E \setminus \{\{v, i\} \mid i \in V\}$;
6: **end while**

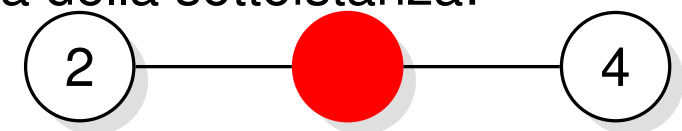
Output: C

Esempi di problemi approssimabili

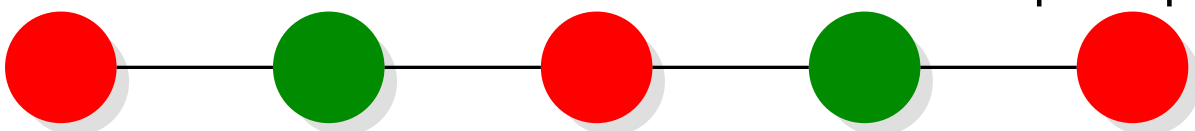
Problema MINIMO RICOPRIMENTO DI VERTICI

L'euristica non determina la soluzione ottima per qualsiasi istanza:

- Per dimostrarlo è sufficiente mostrare che la soluzione ottima può non comprendere le soluzioni ottime dei sottoproblemi.
- La soluzione ottima della sottoistanza:



- non è contenuta nella soluzione ottima dell'istanza principale:



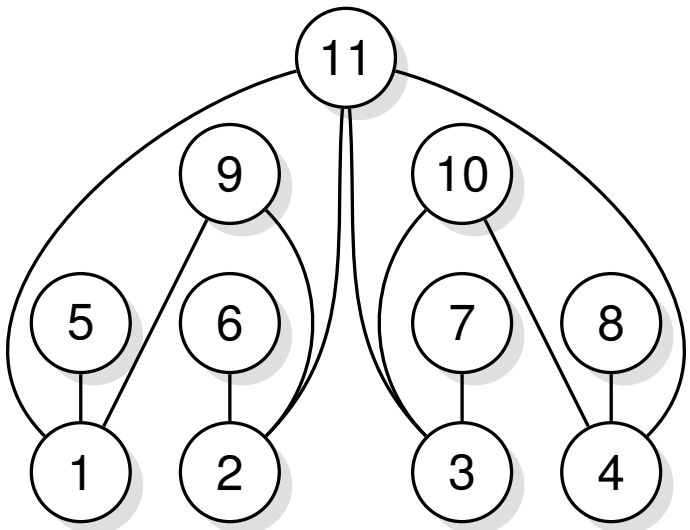
- Ci sono istanze ancora peggiori...

Esempi di problemi approssimabili

Problema MINIMO RICOPRIMENTO DI VERTICI

Infatti l'euristica non è nemmeno un algoritmo r-approssimante perché l'errore della soluzione è dell'ordine di $O(\log n)$.

Si consideri il seguente grafo:



Liv. 3: $m/3$

Liv. 2: $m/2$

Liv. 1: $m/1$

Liv. 0: m nodi

Al livello 0 ci sono m vertici. Al livello $i > 0$ ci sono $\lfloor m/i \rfloor$ vertici. All'ultimo livello, $\lceil (m+1)/2 \rceil$, c'è un solo vertice. Ogni arco collega un vertice del livello 0 a un vertice di un altro livello.

Ogni vertice del livello 0 è collegato con un vertice di ciascun livello. Il vertice al livello massimo ha grado massimo m.

Esempi di problemi approssimabili

Problema MINIMO RICOPRIMENTO DI VERTICI

- L'algoritmo greedy sceglie, in ordine, i vertici a partire dal livello massimo fino al livello 0 escluso.
- La cardinalità dell'insieme determinato è

$$\sum_{i=1}^{\left\lceil \frac{(m+1)}{2} \right\rceil} \left\lfloor \frac{m}{i} \right\rfloor \leq m \int_1^{\left\lceil \frac{(m+1)}{2} \right\rceil} \frac{1}{x} dx = m \ln \frac{(m+1)}{2}$$

- La cardinalità minima è m: è sufficiente prendere i vertici della base per fare un insieme di ricoprimento minimo.
- L'indice di prestazione è quindi

$$r \leq \frac{m \ln \frac{(m+1)}{2}}{m} = \ln \frac{(m+1)}{2} \leq \ln m$$

che non è costante, ma dipendente da m.

Esempi di problemi approssimabili

Problema MINIMO RICOPRIMENTO DI VERTICI

Un algoritmo r-approssimante è dato adottando un approccio ancora più semplice:

Algoritmo 2: r-approssimanteper MINIMO RICOPRIMENTO DI VERTICI

Input: $G = (V, E)$

1: $C = \emptyset$;
2: **while** $(E \neq \emptyset)$ **do**
3: Scegli casualmente un arco $\{v, u\} \in E$;
4: $C = C \cup \{v, u\}$;
5: $V = V \setminus \{v, u\}$;
6: $E = E \setminus (\{\{v, i\} \mid i \in V\} \cup \{\{u, i\} \mid i \in V\})$;
7: **end while**

Output: C

Esempi di problemi approssimabili

Problema MINIMO RICOPRIMENTO DI VERTICI

- C contiene $|C|/2$ archi di G, ciascuno dei quali è disgiunto da tutti gli altri.
- Ogni copertura deve contenere almeno un vertice di questi archi, quindi

$$m^*(G) \geq \frac{|C|}{2} \Rightarrow r = \frac{m(G,C)}{m^*(G)} \leq |C|\frac{2}{|C|} \leq 2$$

- Si verifica facilmente che la complessità dell'algoritmo è polinomiale: ci sono al più $n/2$ cicli while, ciascuno dei quali richiede un tempo $O(n^2)$.
- Il problema quindi è 2-approssimabile.
- È un problema aperto se 2 è il miglior rate possibile per MINIMO RICOPRIMENTO DI VERTICI.

Esempi di problemi approssimabili

Problema Minima Partizione

PROBLEMA MINIMA PARTIZIONE (MIN PARTITION)

DESCRIZIONE: Un insieme finito X di elementi x_i, ciascuno dei quali con peso $a_i \in \mathbf{N}$.

QUESITO: Una partizione degli elementi in due sottoinsiemi Y_1 e Y_2

MISURA: $\max(\sum_{x_i \in Y_1} a_i, \sum_{x_i \in Y_2} a_i)$

Esempi di problemi approssimabili

Problema Minima Partizione

Idea per un algoritmo greedy r-approssimante:

- si ordinano gli elementi in modo non crescente;
- si analizzano gli elementi uno alla volta;
- l'elemento *i* si inserisce nel sottoinsieme che ha peso corrente inferiore.

Esempi di problemi approssimabili

Problema Minima Partizione

La soluzione greedy ha indice di prestazione $7/6$:

- Per assurdo, $x =$ istanza con il minor numero di elementi tale che l'algoritmo greedy è $> 7/6$-approssimante.
- $W(Y_j) \stackrel{\text{def}}{=} \sum_{x_i \in Y_j} a_i \quad j = 1, 2.$
- Due casi possibili: $a_n > m^*(x)/3$ o $a_n \leq m^*(x)/3$.
- Caso 1: $a_n > m^*(x)/3$
  - Si consideri una partizione ottima Y_1^*, Y_2^*.
  - Y_1^*, Y_2^* hanno al più 2 oggetti ciascuno perché tutti gli elementi hanno peso $> m^*(x)/3$ e la somma dell'insieme più pesante deve essere $m^*(x)$.
  - Ci possono essere quindi o 3 o 4 elementi in tutto.
  - In ogni caso, la partizione è la medesima che viene generata dall'algoritmo greedy.
  - Quindi $m(x, y) = m^*(x)$, contraddizione.

Esempi di problemi approssimabili

Problema Minima Partizione

- Caso 2: $a_n \leq m^*(x)/3$
  - $x' \stackrel{\text{def}}{=}$ l'istanza senza l'elemento x_n.
  - Per x' vale $m(x', y') \leq 7/6 m^*(x')$.
  - Dato che, nel caso migliore, $m^*(x') = m^*(x)$, segue che $m(x, y) > m(x', y')$.
  - Quindi l'elemento x_n viene aggiunto all'insieme che avrà peso maggiore alla fine: Y_1
  - $W(Y_2) \geq W(Y_1) - a_n$.
  - Vale $W(x) \geq 2(W(Y_1) - a_n) + a_n$
  - $W(Y_1) \leq W(x)/2 + a_n/2$.
  - Dato che $W(x)/2 \leq m^*(x)$ e $a_n \leq m^*(x)/3$,

$$m(x, y) = W(Y_1) \leq m^*(x) + m^*(x)/6 = 7/6 m^*(x)$$

che contraddice l'assunzione iniziale.

Esempi di problemi approssimabili

Problema Minima Partizione

Altro algoritmo greedy: *si determina una partizione ottima dei primi k elementi più pesanti e, in seguito, si completa tale partizione con lo stesso criterio dell'algoritmo greedy.*

Algoritmo 3: MinimaPartizione(X, r)

Input: Insieme X di elementi x_i con peso a_i; $r > 1$

1: **if** $(r \geq 2)$ **then**
2: Partizione = $X, \emptyset$;
3: **else**
4: Ordina in modo decrescente $(x_1, x_2, \ldots, x_n)$;
5: $k(r) = \lceil (2 - r)/(r - 1) \rceil$;
// Prima fase
6: Trova una partizione ottima Y_1, Y_2 degli elementi $x_1, x_2, \ldots, x_{k(r)}$;

Esempi di problemi approssimabili

Problema Minima Partizione

continua...

7: //continuazione **else**

// Seconda fase

8: **for** $j = k(r) + 1, \ldots, n$ **do**

9: **if** $(\sum_{x_i \in Y_1} a_i \leq \sum_{x_i \in Y_2} a_i)$ **then**

10: $Y_1 = Y_1 \cup \{x_j\}$;

11: **end if**

12: $Y_2 = Y_2 \cup \{x_j\}$;

13: **end for**

14: Partizione = Y_1, Y_2;

15: **end if**

Output: Partizione

Esempi di problemi approssimabili

Problema Minima Partizione

- Data un'istanza X e un $r > 1$, l'algoritmo fornisce una partizione il cui indice di prestazione è al più r.
- Se $r \geq 2$, la soluzione $(X, \emptyset)$ è una soluzione 2-approssimante in quanto qualsiasi soluzione (quindi anche l'ottima) ha valore almeno $W(X)/2$ dove $W(X) = \sum_{x_i \in X} a_i$.
- Si assume quindi che $r < 2$ e si ponga $w(Y_i) = \sum_{x_j \in Y_i} a_j$ per $i = 1, 2$ e $L = w(X)/2$.
- Senza perdere in generalità, si può assumere che $w(Y_1) \geq w(Y_2)$ e che x_h sia l'ultimo elemento inserito in Y_1.

Esempi di problemi approssimabili

Problema Minima Partizione

- Dato il criterio con cui si devono inserire gli elementi, vale che

$$w(Y_1) - a_h \leq w(Y_2)$$

- Sommando $w(Y_1)$ ad entrambi i membri e dividendo per due, si ottiene

$$w(Y_1) - L \leq \frac{a_h}{2}$$

- Se a_h è stato inserito nella prima fase dell'algoritmo, allora la partizione ottenuta è ottima.
- Se, invece, a_h è stato inserito nella seconda fase dell'algoritmo, si deve avere che $a_h \leq a_j$ per qualsiasi $1 \leq j \leq k(r)$ e che $2L \geq a_h(k(r) + 1)$.

Esempi di problemi approssimabili

Problema Minima Partizione

- Dalla relazione $w(Y_1) \geq L \geq w(Y_2)$ e che $m^*(X) \geq L$, si ottiene che

$$\frac{w(Y_1)}{m^*(X)} \leq \frac{w(Y_1)}{L} \leq 1 + \frac{a_h}{L} \leq 1 + \frac{1}{k(r)+1} \leq 1 + \frac{1}{\frac{2-r}{r-1}+1} = r$$

- Si deve ora dimostrare che l'algoritmo opera in tempo polinomiale.
- Occorre tempo $O(n \log n)$ per ordinare gli elementi.
- La prima fase richiede un tempo esponenziale nel numero di elementi $k(r) \Rightarrow O(2^{k(r)})$, quindi sicuramente $O(n^{2/(r-1)})$.
- La fase greedy ha complessità $O(n)$.
- La complessità totale è $O(n \log n + n^{2/(r-1)})$, polinomio in n.

Esempi di problemi approssimabili

Problema Minima Partizione

Nota!

Il tempo di computazione di `MinimaPartizione`(X, r) ha una dipendenza esponenziale rispetto alla costante di approssimazione r: più r si avvicina a 1, più è lunga la sequenza di elementi per i quali si deve cercare la partizione ottima.
Quindi le soluzioni di qualità crescente possono richiedere un tempo che può divenire impraticabile anche a parità di dimensione di istanza.

Nota!

Un algoritmo r-approssimante che determina soluzioni con indice di prestazione passato come parametro si dice essere uno schema di approssimazione polinomiale.

Esempi di problemi approssimabili

Zaino

- ZAINO ammette uno schema di approssimazione polinomiale.
- La dimostrazione dell'esistenza di tale algoritmo è simile alla dimostrazione dell'esistenza di un algoritmo pseudo-polinomiale:
  - Si fissa un'approssimazione nella rappresentazione degli interi nella computazione in modo tale da rendere la computazione polinomiale in tempo.
  - L'errore che si introduce da tale approssimazione può essere "controllata" e resa piccola a piacere.

Esempi di problemi approssimabili

Zaino

Algoritmo 5: ZainoApprossimato(X, W, r)

Input: Insieme X di elementi x_i con peso p_i e valore v_i; W capacità; $r > 1$

1: $v_{\max} = \max v_i\}$;
2: $t = \lfloor \log(\frac{r-1}{r} \frac{v_{\max}}{n}) \rfloor$;
3: $X' =$ istanza con $v'_i = \lfloor \frac{v_i}{2^t} \rfloor$;
4: Elimina da X' elementi con profitto = 0;
5: $Y =$ soluzione programmazione dinamica per X';
6: **return** Y

Esempi di problemi approssimabili

Zaino

Complessità

- La complessità di `ZainoApprossimato`(X, W, r) è $O(n^3 r/(r-1))$.
- È polinomiale in n e proporzionale a $r/(r-1)$.
- Si dice quindi che è polinomiale in n e $r/(r-1)$.

Nota!

Uno schema di approssimazione si dice totale quando è polinomiale anche rispetto alla funzione $1/(r-1)$.