

## ALGORITMICA 2 LEZIONE DEL 21/12/2010

appunti a cura di Emanuele Dinelli

In questa lezione vedremo un esempio completo di approssimazione di problemi NPC.

Prima però facciamo delle considerazioni iniziali che ci permetteranno di capire meglio l'argomento.

- Il fatto che un problema sia NPC potrebbe comunque farci sperare che le istanze reali siano computazionalmente più abordabili analogamente a quanto succede, per esempio, nella compressione dei dati: qui la maggior parte delle stringhe non sono comprimibili.

Kolmogorov complexity: calcola la lunghezza del più piccolo programma codificato in binario, per generare una stringa e quindi indica una misura della "compressibilità" di una stringa.

$2^n$  stringhe binarie

$(01)^n = 010101\dots01$  con circa  $\Theta(\log n)$  bit posso rappresentare la stringa, in questo caso la possiamo comprimere.

Una stringa  $x$  non è comprimibile se la sua complessità di Kolmogorov è  $|x| - c$  per una costante  $c$  (che dipende solo dal modello computazionale adottato). In generale la maggior parte delle stringhe *non* sono comprimibili.

Tuttavia utilizziamo continuamente winzip, gzip, 7zip, bzip e altri tool di compressione perché la maggior parte delle stringhe che incontriamo nella pratica sono comprimibili.

In modo analogo, uno potrebbe sperare che la maggior parte delle istanze pratiche dei problemi NPC completi siano risolvibili efficientemente. Una raccolta dei problemi NPC si trova in un libro scritto da Garey e Johnson e nel compendio indicato nelle pagine web del corso. Sono problemi che frequentemente si incontrano in casi reali.

Tuttavia non sempre accade che le istanze pratiche siano più abordabili dal punto di vista computazionale. Oppure possiamo fare delle ipotesi che i dati di input in ingresso abbiano una certa distribuzione, ma nella realtà si finisce per assumere spesso che la distribuzione sia uniforme. Abbiamo quindi bisogno di affrontarle con tecniche più sofisticate che forniscono un'approssimazione della soluzione per tutte le istanze ammissibili.

- Una variazione del problema può farlo diventare un membro di P o NPC. Per esempio:

#### Problema del ciclo Hamiltoniano

Dato un grafo  $G = (V, E)$ , vogliamo attraversare tutti i *vertici* una e una sola volta.

Questo problema è NPC.

#### Problema del ciclo Euleriano

Dato un grafo  $G = (V, E)$ , vogliamo attraversare tutti gli *archi* una e una sola volta;

Questo problema è in P.

Come si può dedurre dagli esempi precedenti, in alcuni casi, basta cambiare un parametro per far diventare un problema NPC in P.

I problemi NPC è vero che non sono risolvibili, ma se prendiamo solo le istanze che ci servono lo sono; è anche vero però che in questo modo dovremmo forzare i dati in input.

Concentriamoci adesso su problemi di ottimizzazione ( fino ad ora abbiamo considerato quelli decisionali ).

Dato un problema computazionale  $\Pi \subseteq \{0, 1\}^*$  :

#### 1. Versione Decisionale

Verifica che  $x \in \Pi$

Esempio:

$\Pi$  = codifica binaria dei grafi che contengono i cicli Hamiltoniani

$x \in \Pi \Leftrightarrow x$  è la matrice di adiacenza ( memorizzata per righe ) di un grafo che contiene un ciclo hamiltoniano.

#### 2. Versione di Ricerca

Restituisce anche il certificato polinomiale  $y$  oltre a dire sì o no.

Ricordate la definizione di NP che utilizza  $V(x, y)$  verificatore polinomiale.

#### 3. Versione di Ottimizzazione

Oltre al problema  $\Pi$  abbiamo la funzione costo  $C : \text{certificati}(\Pi) \rightarrow \mathbb{R}^+$ . Dato un problema se ammette soluzione, ne può ammettere più

di una e le nostre soluzioni sono i certificati  $y$ , è per questo che  $C$  va dai certificati ai reali positivi.

Per ogni certificato posso avere un costo, e vogliamo trovare quello con costo minimo. Non solo verifichiamo che  $x \in \Pi$  ma vogliamo trovare l'ottimo  $\bar{y} \in \text{certificati}(\Pi(x))$  (cioè t.c.  $V(x, \bar{y}) = \text{TRUE}$ ) per cui vale  $c(\bar{y}) \leq c(y)$  per ogni  $y \in \text{certificati}(\Pi(x))$ .

Questo vale per un problema di minimo costo. Nel caso che il problema sia di massimizzazione (del guadagno), abbiamo  $c(\bar{y}) \geq c(y)$ . Di solito si massimizza il guadagno e si minimizza il costo.

Se abbiamo una soluzione per il problema di ottimizzazione abbiamo la soluzione anche per il decisionale:  $(3) \Rightarrow (2) \Rightarrow (1)$ .

#### 4. Versione di Conteggio

Dà luogo alla classe #P

Non solo verifichiamo se  $x \in \Pi$  ma vogliamo sapere anche il numero di soluzioni, ossia  $|\{y : V(x, y) = \text{TRUE}\}| = |\text{certificati}(\Pi(x))|$  (contiamo il numero di certificati):

se  $\#P > 0 \Rightarrow$  abbiamo soluzione;

se  $\#P = 0 \Rightarrow$  non abbiamo soluzione;.

N.B.  $(4) \Rightarrow (2) \Rightarrow (1)$

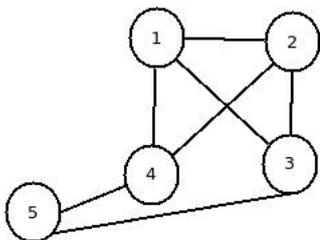
Esercizio. Dimostrare che  $(4) \Rightarrow (2)$ . prendiamo  $\Pi$  e lo trasformiamo in 2 problemi

$\Pi^0 \equiv \Pi$  ma il primo bit è fissato a 0 ( cioè  $y[0] = 0$  )

$\Pi^1 \equiv \Pi$  ma il primo bit è fissato a 1 ( cioè  $y[0] = 1$  )

Esempio:

Ciclo Hamiltoniano:



Prendiamo il vertice (1) e (2) e ci chiediamo quanti cicli hamiltoniani abbiamo da (1) a (2). Se ci risponde sì ok. Se ci risponde no andiamo a vedere quanti ne abbiamo tra (1) e (3) e così via.

Una volta che abbiamo scoperto di avere qualche ciclo, per esempio tra (4) e (3), questi vertici li fissiamo e andiamo avanti chiedendoci tra (1),(3),(5).... ( nel nostro caso fissiamo il primo bit, per togliere ambiguità, in questo esempio logicamente fissiamo i vertici).

Fissiamo (1);

Ci chiediamo se abbiamo un cammino hamiltoniano da (1) a (2) togliendo l'arco che incide sui due vertici; ok  $\exists$  ;

Fissiamo (1),(2)

Ci chiediamo adesso se  $\exists$  da (1)(2)(4); Togliendo tutti gli archi che li collegano. ok  $\exists$  ( da (1) a (4) ci andiamo ) .

Poi fissiamo (1)(2)(4)

Ci chiediamo (1)(2)(4)(3)  $\Rightarrow \nexists$

Ci chiediamo (1)(2)(4)(5) ok  $\exists$

$\Rightarrow$  piano piano troviamo la soluzione,  $\Rightarrow$  (4)  $\Rightarrow$  (2) ok

Altro esempio con SAT:

$$F = (a \vee b \vee c) \wedge (\bar{a} \vee b) \wedge (\bar{b} \vee \bar{c})$$

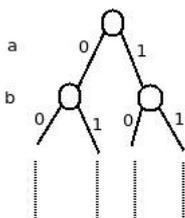
Quanti assegnamenti ci sono?

$a = TRUE$  fornisce la formula ridotta  $b \wedge (\bar{b} \vee \bar{c})$

$a = FALSE$  fornisce la formula ridotta  $(b \vee c) \wedge (\bar{b} \vee \bar{c})$

Ora ci basta sapere quante soluzione ha ciascuna formula ridotta per indovinare il valore di  $a$ . Se sappiamo contare le soluzioni sappiamo anche enumerarle (in realtà basta solo sapere se ce ne è almeno una).

Facciamo così(dove 0=FALSE, 1=TRUE):



Si espande solo dove abbiamo un conteggio  $> 0$  ! Se non abbiamo soluzione ( =0 ) non si visita quella parte dell'albero.

altezza pari al numero  $n$  di variabili.

Grazie alla capacità di contare possiamo togliere i rami che non ci danno soluzione => ha # di foglie e nodi interni proporzionale al # di certificati : **Output Sensitive**

=> se abbiamo poche soluzioni paghiamo poco

=> se abbiamo molte soluzioni paghiamo molto

$O(s * n^c)$  tempo dove  $s$  è il numero di soluzioni e  $n^c$  è un polinomio.

## 5. Versione di Enumerazione

Se sappiamo contare le soluzioni le sappiamo anche enumerare.

La risposta di questa versione può essere naturalmente molto costosa (come le Torri di Hanoi) semplicemente perché di lunghezza esponenziale

La parte seguente della lezione riguarda l'approssimazione del Problema del Commesso Viaggiatore (TSP), la potete trovare sul libro di testo al capitolo 9 spiegata in maniera dettagliata.