CALCOLABILITÀ IN TEMPO POLINOMIALE

Terminologia.

PROBLEMI DECIDIBILI

Esiste un algoritmo e sono quindi sempre risolubili (ma, in generale, solo in linea di principio).

- Problemi decidibili in linea di principio ma non in pratica: PROBLEMI INTRATTABILI

- Problemi decidibili sia in linea di principio che in pratica: PROBLEMI TRATTABILI

PROPOSTA

"Explicatum" formale della nozione intuitiva di "essere trattabile": algoritmo "polinomiale".

UN PROBLEMA È CONSIDERATO TRATTABILE SE ESISTE UN ALGORITMO CHE LO RISOLVE IL QUALE RICHIEDE UN NUMERO DI PASSI LIMITATO DA QUALCHE POLINOMIO DELLA LUNGHEZZA DELLA VARIABILE DI INGRESSO.

Time complexity function	Size n					
	10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second
n^5	.1 second	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

Figure 1.2 Comparison of several polynomial and exponential time complexity functions.

Time complexity function	Size of Largest Problem Instance Solvable in 1 Hour: With present computer	With computer 100 times faster	With computer 1000 times faster
n	N_1	$100\ N_1$	$1000\ N_1$
n^2	N_2	$10\ N_2$	$31.6\ N_2$
n^3	N_3	$4.64\ N_3$	$10\ N_3$
n^5	N_4	$2.5\ N_4$	$3.98\ N_4$
2^n	N_5	$N_5+6.64$	$N_5+9.97$
3^n	N_6	$N_6+4.19$	$N_6+6.29$

Figure 1.3 Effect of improved technology on several polynomial and exponential time algorithms.

Definizione

Un linguaggio L su un alfabeto A si dice decidibile in tempo polinomiale se esiste una MdT M che accetta L ed un polinomio $p(n)$ tale che il numero di passi di ogni calcolo di M che accetta una stringa x è $\leq p(|x|)$.

Sarà indicata con P la classe dei linguaggi decidibili in tempo polinomiale (con P_A se vogliamo indicare esplicitamente l'alfabeto)

Definizione

Una funzione f su A^* si dice calcolabile in tempo polinomiale se esiste una MdT M che calcola f ed un polinomio $p(n)$ tale che il numero di passi del calcolo che effettua M a partire dalla variabile d'ingresso x è $\leq p(|x|)$.

Osservazione

È sufficiente che la limitazione mediante $p(|x|)$ avvenga per tutte le stringhe tranne un numero finito.
Infatti, sia c il massimo numero di passi usati da M in queste "eccezioni" (in numero finito): basta sostituire il polinomio $p(n)$ col polinomio $p(n)+c$.

Proposizione

Sia $L \in P$ e sia f una funzione calcolabile in tempo polinomiale su A^*. Sia $Q = \{x \in A^* \mid f(x) \in L\}$, allora $Q \in P$.

Dimostrazione

Sia M la MdT che accetta L in tempo polinomiale e sia N la MdT che calcola $f(x)$ in tempo polinomiale.

L'appartenenza di x a Q può quindi essere determinata mediante una MdT R che prima fa girare N su x per calcolare $f(x)$ e poi fa girare M su $f(x)$ per determinare se $f(x) \in L$.

R darà quindi una risposta in tempo polinomiale (anche se il polinomio che limita il calcolo sarà di grado maggiore: il prodotto dei gradi di quelli di M ed N).

Proposizione

Siano f e g due funzioni calcolabili in tempo polinomiale tali che la loro composizione $h(x) = f(g(x))$ sia totale.
Allora h è calcolabile in tempo polinomiale.

Dimostrazione

Anche in questo caso basta dare come variabile di ingresso alla MdT che calcola f il valore calcolato dalla MdT che calcola g.
Poiché ciascuna delle due MdT lavora in tempo polinomiale anche la risposta finale sarà fornita in tempo polinomiale.

TESI DI COOK-KARP

IL CALCOLO DI UNA FUNZIONE f È UN PROBLEMA TRATTABILE SE E SOLO SE f È CALCOLABILE IN TEMPO POLINOMIALE

o, in modo equivalente,

IL PROBLEMA DI DETERMINARE L'APPARTENENZA DI UNA STRINGA AD UN LINGUAGGIO L È TRATTABILE SE E SOLO SE $L \in P$

(viene enunciata in analogia alla tesi di Church-Turing)

Definizione

Si dice che un linguaggio L appartiene alla classe NP se esiste una MdT NON-DETERMINISTICA M che accetta L ed un polinomio $p(n)$ tale che per ciascun $x \in L$ vi è un calcolo che accetta x in un numero di passi inferiore o eguale a $p(|x|)$.

TEOREMA

Se $L \in NP$ allora L è ricorsivo.

Dimostrazione

Se $L \in NP$ allora esiste una MdT non deterministica che accetta L e sia $p(n)$ il polinomio che limita il tempo di calcolo.

Sia γ_1 la configurazione iniziale della MdT: $\overset{s_0\, x}{\uparrow}\ q_1$

Esaminando le quadruple della MdT (NON-DET.) possiamo scrivere tutte le varie configurazioni γ_2^i alle quali possiamo passare a partire da γ_1.

Possiamo quindi determinare tutte le varie configurazioni $\gamma_1, \dots, \gamma_m$ con $m \leq p(|x|)$ e quindi tutte le possibili sequenze

$$\gamma_1 \vdash \gamma_2 \vdash \dots \vdash \gamma_m$$

che portano dalla iniziale ad una raggiunta dopo un numero di passi minore o eguale a $p(|x|)$.

Abbiamo quindi un modo di controllare se $x \in L$, vedendo se almeno una di queste sequenze conduce dalla configurazione iniziale ad una configurazione terminale.

Abbiamo quindi un algoritmo che risponde SI o NO alla domanda: $x \in L$?

Quindi L è ricorsivo.

Corollario

Se $L \in P$ allora L è ricorsivo

Dim.

Basta ricordare che $P \subseteq NP$.

Poniamoci adesso la domanda:

L'inclusione $P \subseteq NP$ è propria, esiste un linguaggio L tale che $L \in NP$ ma $L \notin P$?

A tutt'oggi non si è data risposta al problema. Elencheremo di seguito alcuni degli "strumenti concettuali" che sono stati forgiati per affrontare il problema.

Definizione

Siano L e Q due linguaggi, diciamo che Q è riducibile ad L in tempo polinomiale

$$Q \leq_P L$$

se esiste una funzione calcolabile in tempo polinomiale f tale che:

$$x \in Q \iff f(x) \in L.$$

Discende immediatamente dalla definizione precedente e dalla proposizione sulla composizione di funzioni calcolab. in t. polinomiale che:

PROPOSIZIONE:

Se $R \leq_P Q$ e $Q \leq_P L$ allora $R \leq_P L$

Definizione

Un linguaggio L si dice NP-completo se $L \in NP$ e per ogni $Q \in NP$ Q è riducibile in tempo polinomiale ad L, $Q \leq_p L$.

Perché è significativa la definizione precedente? La risposta è data dalla proposizione seguente:

Proposizione

Se esiste un linguaggio L NP-completo tale che $L \in P$ allora $NP = P$

Dimostrazione

Si deve mostrare che qualunque sia $Q \in NP$, $Q \in P$.

Sia $Q \subseteq A^*$ un generico linguaggio NP.

Poiché, per ipotesi, L è NP-completo si ha che $Q \leq_p L$, cioè esiste una f calcolabile in tempo polinomiale tale che

$$x \in Q \iff f(x) \in L ;$$

ma allora si ha che

$$Q = \{x \in A^* \mid f(x) \in L\}$$

Poiché per ipotesi $L \in P$ ed f è calcol. in tempo polinomiale, in base ad una proposizione precedente, si ha che $Q \in P$.

Richiamo

Il problema della soddisfacibilità.

(è un problema chiave del calcolo proposizionale; qui ne diamo una versione direttamente utilizzabile in questo contesto).

Sia $V = \{u_1, \ldots, u_m\}$ un insieme di variabili che assumono solo il valore 0 o il valore 1.

Un'*assegnazione di verità* per V è una funzione $t : V \to \{0,1\}$ dove $t(u) = 1$ lo interpretiamo come u è vera sotto t e $t(u) = 0$ lo interpretiamo come u è falsa sotto t.

Chiamiamo *letterali* su V sia ogni variabile u di V che ogni sua negazione $\bar{u}$.

(Si ha ovviamente che un letterale u è vero sotto t se e solo se la variabile u è vera sotto t e un letterale $\bar{u}$ è vero sotto t se e solo se la variabile u è falsa).

Si dice *clausola* su V la *disgiunzione* di un insieme di letterali su V.

Una clausola è vera sotto un'assegnazione t se e solo se almeno uno dei suoi disgiunti è vero sotto tale assegnazione.

Un insieme $\mathcal{C}$ di clausole su V si dice *soddisfacibile* se e solo se esiste un'assegnazione di verità per V tale che soddisfa *simultaneamente* tutte le clausole di $\mathcal{C}$.

PROBLEMA DELLA SODDISFACIBILITÀ (SAT)

Sia dato un insieme V di variabili ed un insieme $\mathcal{C}$ di clausole su V.

Problema:
Esiste un'assegnazione di verità per V che soddisfa $\mathcal{C}$?

Esempi

Sia $V = \{u_1, u_2\}$

e $\mathcal{C}_1 = \{u_1 \vee \bar{u}_2,\ \bar{u}_1 \vee u_2\}$

Ponendo $t(u_1) = t(u_2) = 1$ si vede che $\mathcal{C}_1$ è soddisfatto.

Consideriamo adesso

$$\mathcal{C}_2 = \{u_1 \vee u_2,\ u_1 \vee \bar{u}_2,\ \bar{u}_1\}$$

Si vede facilmente che non esiste alcuna assegnazione che soddisfa $\mathcal{C}_2$.

~~$t(u_1) = 1$~~ ~~$t(u_1) = 1$~~ $t(u_1) = 0$ ~~$t(u_1) = 0$~~
~~$t(u_2) = 0$~~ ~~$t(u_2) = 1$~~ $t(u_2) = 1$ ~~$t(u_2) = 0$~~

Il problema SAT è decidibile: esiste un algoritmo "ovvio", quello delle tavole di verità.

Esistono anche algoritmi efficienti?

TEOREMA

SAT $\in$ NP

TEOREMA (Cook, 1971)

SAT è NP-completo.

(Questo che precede è uno dei teoremi centrali della teoria della calcolabilità in tempo polinomiale).

Commenti:

1. In base ad una delle proposizioni precedenti si ha che

   Se SAT $\in$ P allora NP = P

2. Sarebbe interessante avere a disposizione quanti più problemi NP-completi si può. Come si studia la NP-completezza di un problema Π?
   - si mostra che Π è NP e, successivamente,
   - si sceglie qualche problema NP-completo Π' e si costruisce una trasformazione f di Π' in Π.
   - si dimostra, infine, che f è polinomiale.

Per potere usare la procedura precedente è però importante potere avere a disposizione già un problema che è stato dimostrato essere NP-completo.

Da qui si vede l'importanza del teorema di Cook che è stato il primo di questo genere. Vale quindi la pena di guardare un minimo dentro la struttura della dimostrazione del teorema.

I due punti base sono:

1. Mostrare che SAT è NP
2. Mostrare che qualsiasi altro problema NP è riconducibile a SAT

Come si fa a studiare il punto 2?
È ovviamente impossibile prendere in esame tutti i problemi NP.
L'osservazione cruciale è che ciascun linguaggio NP può essere descritto mediante una MdT non deterministica che lavora in tempo polinomiale.
Ciò che viene fatto nella dimostrazione del teorema di Cook è quindi mostrare che il linguaggio riconosciuto da una generica MdT di questo tipo è riducibile a L_{SAT}.

- La ricerca di algoritmi polinomiali ha messo, infine, in evidenza che problemi apparentemente molto lontani tra loro sono collegati da trasformazioni in tempo polinomiale.

ESEMPI

PROBLEMA DEL SOTTOGRAFO COMPLETO

Richiami: Un grafo G consiste di un insieme finito non vuoto di vertici $V = \{v_1, \ldots, v_n\}$ ed un insieme finito E di archi.
Ciascun arco è una coppia di vertici.

Dimensione del grafo: numero di vertici che contiene.

Sottografo di $G = (V, E)$
è un grafo $G' = (V', E')$ con $V' \subseteq V$ e $E' \subseteq E$.

Un grafo è **completo** se esiste un arco in E che collega ogni coppia di vertici distinti di V.

PROBLEMA:

DATO UN GRAFO ED UN NUMERO k,
ESISTE UN SOTTOGRAFO COMPLETO
DI DIMENSIONE k ?

TEOREMA:

IL PROBLEMA DEL SOTTOGRAFO COMPLETO
È NP-COMPLETO

PROBLEMA DEL RICOPRIMENTO (SET-COVER)

Determinare per una famiglia di insiemi $\Delta = \{S_1, S_2, \ldots, S_n\}$ ed un numero k, se esista o meno una sottofamiglia Γ di Δ di dimensione k

$$\Gamma = \{S_{m_1}, \ldots, S_{m_k}\}$$

tale che $\bigcup_{i \le n} S_i = \bigcup_{j \le k} S_{m_j}$

TEOREMA:
SET-COVER è NP-completo.

RICOPRIMENO ESATTO (EXACT COVER)

Come sopra, si cerca un Γ tale che gli elementi di Γ siano a due a due disgiunti.

TEOREMA
EXACT COVER è NP-completo

Conclusioni provvisorie

1. Se $P = NP$
   allora, fatta salva l'esistenza di problemi intrattabili, la situazione dei problemi risolubili in tempo polinomiale è risolta nel modo più semplice.

2. Se $P \neq NP$ allora può dimostrarsi che esistono problemi in NP che non sono né risolubili in tempo polinomiale né NP-completi, per cui la struttura dei problemi NP è un po' complessa.
   Si ha infatti che