Automi e Linguaggi Formali

Problemi intrattabili, classi P e NP

Università degli Studi di Padova

A.A. 2014-2015
Alessandro Sperduti
sperduti@math.unipd.it

Problemi intrattabili

- Ci occuperemo solo di problemi decidibili, cioè ricorsivi.
- Tra loro, alcuni sono detti trattabili, se si può provare che sono risolvibili in tempo polinomiale in modo deterministico.
- Gli altri sono detti intrattabili.

Classe P

- Una Tm M ha **complessità in tempo T(n)** se, dato un input w di lunghezza n, M si ferma dopo al massimo $T(n)$ passi.
- Esempi: $T(n) = 5n^2 + 3n$, o $T(n) = 4^n + 3n^2$
- Un linguaggio L è nella classe $\mathcal{P}$ se esiste un polinomio $T(n)$ tale che $L = L(M)$ per una TM deterministica M con complessità in tempo $T(n)$.
- Se la complessità non è polinomiale, si dice che è esponenziale, anche se $T(n)$ non è un esponenziale.
- Esempio: $T(n) = n^{\log_2 n}$. Cresce più velocemente di qualunque polinomio, ma più lentamente di qualunque esponenziale 2^{cn}

Esempio di problema in $\mathcal{P}$

- Problema: trovare un albero di copertura di peso minimo in un grafo.
- Grafo: nodi, archi con peso (intero).
- Albero di copertura: sottoinsieme degli archi che connetta tutti i nodi senza cicli.
- Peso minimo: se il peso totale dei suoi archi è minimo fra tutti gli alberi di copertura.

Algoritmo di Kruskal

- Componente connessa di un nodo: nodi raggiungibili da lui con gli archi selezionati finora.
- All'inizio: nessun arco, quindi ogni nodo è una componente connessa da solo.
- Ad ogni passo, si considera un altro arco, di peso minimo: se unisce due nodi in componenti separate, lo scelgo e unisco le componenti, altrimenti non lo scelgo (creerebbe un ciclo).
- Finché tutti gli archi sono stati esaminati, o il numero degli archi scelti è uguale al numero dei nodi meno 1 (tutti connessi).

Complessità in tempo

- Se m nodi e e archi, tempo $O(e(e+m))$:
- Ad ogni passo, in tempo $O(e)$ scelgo un arco, in tempo $O(m)$ riunisco due componenti. Ci sono al massino e passi.
- Quindi ha una complessità in tempo polinomiale.

Classe NP

- Un linguaggio L è nella classe NP (**non deterministica polinomiale**) se esiste una TM non deterministica M tale che $L = L(M)$ e, quando M ha un input lungo n, al massimo fa $T(n)$ mosse con $T(n)$ polinomio.
- Dato che ogni TM deterministica è una TM non-deterministica, allora $\mathcal{P} \subseteq \mathcal{NP}$.
- Una TM non deterministica polinomiale può esaminare un numero esponenziale di strade "in parallelo". Quindi sembra che sia impossibile che $\mathcal{P} = \mathcal{NP}$. Ma **nessuno lo ha mai provato**.

Algoritmo non-deterministico con tempo polinomiale

- prima fase (non deterministica): genero una stringa
- seconda fase: controllo se la stringa appartiene al linguaggio (algoritmo deterministico con tempo polinomiale)

Problema del commesso viaggiatore (TSP)

- Input: grafo con pesi interi sugli archi + limite di peso W.
- Problema: se il grafo ha un circuito Hamiltoniano di peso totale al massimo W.
- Circuito Hamiltoniano: insieme di archi che connettono i nodi in un unico ciclo in cui ogni nodo appare una sola volta.
- In un grafo di m nodi, il numero di cicli distinti cresce come il fattoriale di m, che cresce più di 2^{cm}.
- Qualunque algoritmo di soluzione sembra dover esaminare tutti (o un numero esponenziale) di cicli e calcolare il loro peso totale.
- Con un computer non-deterministico, potremmo scegliere in ogni ramo una permutazione dei nodi e calcolare il suo peso in tempo polinomiale. Quindi il problema è in NP.

Riduzioni polinomiali

- Stesso concetto delle riduzioni già viste, solo con l'aggiunta del vincolo che deve essere fatta in tempo polinomiale
- Riduzione: da una istanza (stringa accettata) di P_1 ad una di P_2
- Se P_1 è riducibile a P_2: se P_2 è in P, allora lo è anche P_1. Se P_1 non è in P, neanche P_2 lo è.
- Intuizione: P_2 è difficile almeno tanto quanto P_1.

Problemi NP-completi

- Un linguaggio L è NP-completo se
  - è in NP
  - per ogni altro linguaggio L' in NP, esiste una riduzione polinomiale di L' a L.
- Esempio: problema del commesso viaggiatore.
- Sono i problemi più difficili tra quelli in NP.
- **Teorema 10.4**: Se P_1 è NP-completo e P_2 è in NP, e posso ridurre polinomialmente P_1 a P_2, allora P_2 è NP-completo.
  Prova: la riduzione polinomiale è transitiva.
- **Teorema 10.5**: Se un problema NP-completo è in P, allora P=NP.

Problemi NP-ardui

- Un problema è NP-arduo se ogni problema in NP è riducibile polinomialmente a lui.
- Non è detto che sia in NP.
- Quindi potrebbe anche non avere un algoritmo non-deterministico polinomiale (se non è in NP).

Il problema della soddisfacibilità

Espressioni Booleane:

- variabili a due valori: 1 (vero) e 0 (falso)
- operatori binari $\wedge$ (and logico) e $\vee$ (or logico)
- operatore unario $\neg$ (negazione logica)
- parentesi per raggruppare

Esempio: $x \wedge \neg(y \vee z)$

SAT

- Un assegnamento T di valori alle variabili soddisfa l'espressione Booleana E se il valore di E (cioè $E(T)$) è 1 (vero).
- E è soddisfacibile se esiste almeno un T tale che $E(T) = 1$.
- Esempio: $x \wedge \neg(y \vee z)$ è soddisfacibile (x=1,y=0,z=0)
- Esempio: $x \wedge (\neg x \vee y) \wedge \neg y$ non è soddisfacibile
- Problema (SAT): data E, è soddisfacibile?
- Linguaggio: insieme delle espressioni Booleane soddisfacibili.

Codifica di SAT

- Da un problema SAT ad una stringa su un alfabeto.
- Alfabeto: $\{\wedge, \vee, \neg, (,), 0, 1, x\}$
- 0 e 1 servono per rappresentare l'indice di una variabile.
- Esempio: $x_1 \wedge \neg(x_2 \vee x_3)$ diventa $x1 \wedge \neg(x10 \vee x11)$.
- Se E ha m variabili, la codifica ha $O(m \log m)$ simboli.

Teorema di Cook (1971): SAT è NP-completo

- Prima parte: SAT è in NP.
- Basta prendere una TM non-deterministica che genera tutti gli assegnamenti in parallelo (2^n se n variabili).
- Per ogni assegnamento, controlla se è soddisfacibile (in tempo polinomiale).
- Quindi esiste una TM non-deterministica polinomiale che risolve SAT.
- Seconda parte: Dato un qualunque L in NP, esiste una riduzione polinomiale da L a SAT.

Forme normali

- Letterale: variabile o variabile negata. Es.: x, $\neg y$.
- Clausola: OR di più letterali. Es.: x, $x \vee \neg y$.
- Forma normale congiuntiva (CNF): AND di una o più clausole. Es.: $(x \vee y) \wedge (\neg x \vee z)$.
- k-CNF: se le clausole hanno k letterali.
- CSAT: una data espressione Booleana in CNF è soddisfacibile?
- kSAT: una data espressione Booleana in k-CNF è soddisfacibile?
- CSAT, 3SAT e kSAT per $k \geq 3$ sono NP-completi. Algoritmi lineari per 1SAT e 2SAT.

Riduzioni

- Da espressione Booleana a CNF in tempo polinomiale $\Rightarrow$ CSAT è NP-completo.
- Da CSAT a 3SAT in tempo lineare $\Rightarrow$ 3SAT è NP-completo.

Da CSAT a 3SAT

- CNF: $e_1 \wedge e_2 \wedge \ldots \wedge e_k$
- Se $e_i = x$, due nuove variabili u e v, e sostituiamo x con $(x \vee u \vee v) \wedge (x \vee u \vee \neg v) \wedge (x \vee \neg u \vee v) \wedge (x \vee \neg u \vee \neg v)$
- Se $e_i = (x \vee y)$, nuova variabile z e sostituiamo e_i con $(x \vee y \vee z) \wedge (x \vee y \vee \neg z)$.
- Se $e_i = (x_1 \vee \ldots \vee x_m)$ con $m \geq 4$, nuove variabili $y_1, y_2, \ldots, y_{m-3}$, e sostituiamo e_i con $(x_1 \vee x_2 \vee y_1) \wedge (x_3 \vee \neg y_1 \vee y_2) \wedge \ldots (x_{m-2} \vee \neg y_{m-4} \vee y_{m-3}) \wedge (x_{m-1} \vee x_m \vee \neg y_{m-3})$.
  - Se x_j vera, settiamo $y_1, \cdots, y_{j-2}$ vere e $y_{j-1}, y_{j+1}, \cdots, y_{m-3}$ false
  - Se tutte le x sono false, non è possibile rendere vere $m-2$ clausole con $m-3$ variabili (le y)

2SAT è in P

- Prendiamo una variabile x e diamo valore 1 (vero)
- In ogni clausola con $\neg x$, l'altro letterale deve essere vero
  - Esempio: in $(\neg x \vee \neg y)$, y deve essere falso (0)
- Continuiamo settando le variabili il cui valore è "forzato"

Finché ...

- Può succedere una di tre cose:
  1. una contraddizione: una variabile forzata ad essere sia vera che falsa
  2. tutte le variabili con valore forzato sono state settate, ma ancora ci sono clausole non soddisfatte
  3. si ottiene un assegnamento che dà valore vero all'espressione

Caso 1 e 3

- Caso 1: ci può essere un assegnamento che dà valore vero solo con l'altro valore per la variabile x. Ricominciamo con l'altro valore per x.
- Caso 2: ci sono ancora variabili e clausole che non sono settate. Eliminiamo le clausole soddisfatte, e ripartiamo.
- Caso 3: la risposta è "si" (trovato un assegnamento che soddisfa l'espressione).

Problema degli insiemi indipendenti (IS)

- Insiemi indipendenti (IS): dato un grafo, esiste un insieme indipendente di dimensione almeno k?
- Insieme indipendente = sottoinsieme I di nodi di G dove nessuna coppia di nodi di I è collegata da un arco di G.
- Riduzione polinomiale da 3SAT a IS: da un'espressione E in 3-CNF con m clausole ad un grafo G tale che E soddisfacibile se e solo se G ha un insieme indipendente di dimensione m.

Da 3SAT a IS

- Se $E = e_1 \wedge e_2 \wedge \ldots \wedge e_m$, construisco un grafo con $3m$ nodi.
- Ogni nodo $[i, j]$ rappresenta il j-esimo letterale di e_i.
- Arco tra $[i, j]$ e $[i, k]$ (diversi letterali della stessa clausola): un solo nodo per clausola.
- Arco tra $[i_1, j_1]$ e $[i_2, j_2]$ se uno rappresenta x e l'altro $\neg x$.
- Limite $= m$.
- Da insieme indipendente con m nodi ad assegnamento che soddisfa E.
- Da assegnamento che soddisfa E ad insieme indipendente di dimensione m.

Problema della copertura per nodi (NC)

- Copertura per archi di un grafo: insieme di archi tale che ogni nodo è toccato da almeno un arco. Copertura minimale se ogni altra copertura ha più archi.
- Se un grafo ha una copertura per archi di k elementi: polinomiale.
- Copertura per nodi (NC): insieme di nodi che contiene almeno un estremo di ogni arco.
- Se un grafo ha una copertura per nodi con non più di k nodi: NP-completo.
- Riduzione da IS a NC: il complemento di un insieme indipendente è una copertura per nodi, e viceversa.

Problema del circuito Hamiltoniano (HC)

- Problema: se un grafo ha un circuito Hamiltoniano.
- HC caso speciale di TSP, con pesi tutti a 1 $\Rightarrow$ facile ridurre HC a TSP in tempo polinomiale.
- DHC: HC con archi orientati.
- DHC è NP-completo: riduzione 3SAT a DHC.
- HC è NP-completo: riduzione da DHC a HC.
- TSP è NP-completo: riduzione da HC a TSP.

Riduzione da DHC a HC

- Da G_d (grafo orientato) a G_u (grafo non orientato)
- Nodo v di $G_d \Rightarrow$ 3 nodi v^0, v^1, v^2
- Archi:
  - $\forall v$ di G_d, archi (v^0, v^1) e (v^1, v^2)
  - arco $v \rightarrow w \Rightarrow$ arco (v^2, w^0)
- G_u ha un circuito Hamiltoniano sse G_d ha un circuito Hamiltoniano orientato
- Prova:
  - Se $v_1, \ldots, v_n, v_1$ circuito Hamiltoniano orientato di G_d, allora $v_1^0, v_1^1, v_1^2, v_2^0, v_2^1, v_2^2, \ldots, v_1^0$ circuito Hamiltoniano di G_u
  - v_i^1 ha solo 2 nodi adiacenti (v_i^2 e v_i^0), quindi uno lo deve precedere e uno lo deve seguire in un circuito Hamiltoniano $\Rightarrow$ schema 012012...0 oppure 210210...2 $\Rightarrow$ circuito Hamiltoniano orientato di G_d fatto con gli archi corrispondenti a $v \rightarrow w$ o $w \rightarrow v$

Riduzione da HC a TSP

- Da grafo G a grafo pesato G'
- Stessi archi e nodi di G
- Peso 1 su tutti gli archi
- Limite k = numero n di nodi di G
- G' ha un circuito Hamiltoniano di peso n sse G ha un circuito Hamiltoniano

Complemento

- Se un problema è in P, anche il suo complemento è in P
  - Prendiamo l'algoritmo deterministico polinomiale e scambiamo il "si" con il "no" (stati finali con stati non finali, e viceversa)
- Se un problema è in NP, non sappiamo se il suo complemento è in NP in generale
- Co-NP: linguaggi i cui complementi sono in NP
- Si pensa che nessun problema NP-completo abbia il complemento in NP, quindi che nessun problema NP-completo sia in co-NP
- Se P = NP, allora Co-NP = NP

Esempio: SAT

- Il complemento di SAT (tutte le espressioni Booleane che non sono soddisfacibili, più le stringhe che non sono espressioni Booleane) sembra non essere in NP
- Possiamo considerare (non-deterministicamente) un assegnamento alle variabili e controllare in tempo polinomiale se soddisfa l'espressione
- Se devo dimostrare che non c'è nessun assegnamento che soddisfa l'espressione, devo guardare tutti gli assegnamenti