Appunti di Teoria della Calcolabilità e Coplessità

Anonimo

5 ottobre 2015

Capitolo 10

In questo capitolo si parla solo di problemi ricorsivi, chiedendosi non più se siano o meno calcolabili, ma esaminandone la complessità. In particolare si dimostra l'*NP-completezza* di alcuni problemi.

Per questi problemi si ritiene che una macchina di Turing polinomiale deterministica non esista, mentre ne esiste una polinomiale non deterministica che ne accetta il linguaggio.

La procedura normalmente utilizzata per dimostrare l'NP-completezza di un problema consiste nel ridurre polinomialmente un problema noto, che si sa già essere NP-completo, al problema nuovo. Questo metodo presuppone l'esistenza di problemi NP-completi noti.

Come prima cosa questo capitolo dimostrare l'NP-completezza di alcuni problemi di partenza, senza appoggiarci nessun problema noto.

L'NP-completezza viene poi estesa ad altri problemi attraverso il metodo della riduzione polinomiale, fino ad avere un insieme di problemi NP-completi molto famosi, appartenenti a diverse discipline.

Definizione (Riduzione polinomiale)**.** Una riduzione si dice *polinomiale* se, data una istanza in input di lunghezza n, il tempo impiegato dall'algoritmo per produrre l'istanza in output è poliniomiale rispetto a n.

Nota. Chiaramente se la riduzione impiega *tempo* polinomiale per produrre il suo output, anche la *lunghezza* del suo output sarà polinomiale rispetto a l'intput. Questo perchè una TM non può accedere a più di una nuova cella di nastro per mossa.

Definizione (Problema NP-completo)**.** Sia L un linguaggio. Diciamo che L è *NP-completo* se:

1. $L \in \mathbf{NP}$.
2. Per ogni L' in **NP** esiste una riduzione polinomiale di L' a L.

Teorema. Sia P_1 un problema NP-completo e sia $P_2 \in \mathbf{NP}$. Se esiste una riduzione polinomiale di P_1 a P_2 allora P_2 è NP-completo.

Dimostrazione. Per dimostrare che P_2 è NP-completo dobbiamo dimostrare i due punti della definizione.

Dato che per ipotesi $P_2 \in \mathbf{NP}$ (punto 1), dobbiamo solo dimostrare che per ogni $P \in \mathbf{NP}$ esiste una riduzione polinomiale di P a P_2 (punto 2).

Sappiamo che per un generico $P \in \mathbf{NP}$ esiste una riduzione polinomiale da P a P_1, perchè P_1 è NP-completo. Assumiamo la riduzione impieghi tempo $p(n)$.

Sappiamo inoltre che esiste una riduzione polinomiale da P_1 a P_2. Assumiamo impieghi tempo $q(m)$.

Costruiamo la nostra riduzione attraverso la composizione delle due precedenti. La riduzione ottenuta è in grado di trasformare istanze di P in istanze di P_2 in tempo $p(n) + q(p(n))$, pertanto è polinomiale.

□

Teorema. Se un problema NP-completo è in **P**, allora $\mathbf{P} = \mathbf{NP}$.

Dimostrazione. Sia $P \in \mathbf{P}$ NP-completo.

Allora per ogni $P' \in \mathbf{NP}$ esiste una riduzione polinomiale da P' a P.
Allora per ogni $P' \in \mathbf{NP}$ vale anche $P' \in \mathbf{P}$.

□

I teoremi appena dimostrati si appoggiano sempre a un problema NP-completo già noto. Abbiamo quindi bisogno di almeno un problema NP-completo la cui dimostrazione non si basi su una riduzione. Questo ruolo sará ricoperto da SAT, un problema che andremo a definire qui di seguito

Il problema SAT consiste nel dire se una data espressione booleana è *soddisfacibile*, ossia se esiste una particolare configurazione di valori di verità per le variabili dell'espressione per le quali l'espressione risulta vera.

Per capire come rappresentare istanze di SAT dobbiamo chiarire come rappresentare espressioni booleane, per essere elaborate da una macchina di Turing.

Nota (Struttura delle espressioni booleane)**.** Le espressioni booleane sono costituite a partire da:

1. Variabili a valori booleani, ossia 1 (vero) e 0 (falso).
2. Gli operatori binari $\wedge$ (and) e $\vee$ (or).
3. L'operatore unario $\neg$ (not).
4. Le parentesi aperta e chiusa, per raggruppare sotto-espressioni.

Nota (Alfabeto del linguaggio SAT)**.** L'alfabeto del linguaggio SAT è costituito da 8 simboli:

1. I simboli $\wedge$, $\vee$, $\neg$, (e), che rappresentano se stessi.
2. La variabile x_i è rappresentata dal simbolo x seguito dalla rappresentazione binaria di i. Gli indici usati vanno da 1 in su, e non possono essere saltati degli interi.

Nota (Lunghezza delle espressioni codificate)**.** La lunghezza di una espressione codificata è pari a circa il numero di posizioni (occorrenze di variabili) dell'espressione originale.

Se l'espressione ha m posizioni può avere al massimo m variabili distinte, di cui l'ultima avrà indice m. La codifica di una variabile generica non richiederà quindi più di $O(\log(m))$ simboli.

Con m posizioni possiamo quindi avere una espressione lunga $O(m \log(m))$ simboli, e la differenza tra m e $m \log(m)$ è senz'altro limitata da un polinomio.

Teorema (*Cook*). SAT è NP-completo.

Dimostrazione. Dimostriamo che SAT $\in$ **NP** illustrando una NTM che accetta SAT:

1. *Parte non-deterministica:* Una NTM può congetturare l'assegnamento di valori di verità per le variabili dell'espressione data E. Se E ha lunghezza n questa fase richiede tempo $O(n)$.
   Questo perchè in n mosse la NTM può raggiungere 2^n ID, corrispondenti ad altrettanti tentativi di assegnamento di valori di verità (ossia tutti quelli possibili per un espressione di n variabili distinte, potendo una variabile assumere solo due valori).

2. *Parte deterministica:* Dato un assegnamento di valori di verità T, dalla fase precedente, valutiamo se il valore di E risulta vero, ossia se $E(T) = 1$. In tal caso accettiamo.
   Questa valutazione può essere fatta in tempo $O(n^2)$ su una NTM multinastro, oppure $O(n^4)$ su una NTM a nastro singolo.

Dato che SAT può essere accettato in tempo polinomiale da una NTM, SAT $\in$ **NP**.

Dobbiamo ora dimostrare che per ogni $P \in$ **NP** esiste una riduzione polinomiale da P a SAT.

Scegliamo arbitrariamente $P \in$ **NP**, e sia M la NTM a nastro singolo associata a P. Sia w una istanza arbitraria di P. Supponiamo inoltre che M non impieghi più di $p(n)$ passi su un input di lunghezza n.

Senza perdere generalità questa dimostrazione assume che M non scriva mai un simbolo di blank, e non sposti mai la testina a sinistra della posizione

iniziale. Si assume inoltre che q_0 sia lo stato iniziale di M e che q_1 sia il suo unico stato finale.

Ideeremo qui di seguito un algoritmo polinomiale per trasformare l'istanza w di P in un istanza di SAT, avente come input M e w.

Essendo P un problema generico, possiamo solo dire che se M accetta w, allora eseguirà una serie di transizioni partendo dallo stato iniziale fino ad arrivare allo stato accettante. Queste transizioni, insieme al contenuto del nastro, possono essere descritte da una serie di ID:

$$\alpha_0 \vdash \alpha_1 \vdash \alpha_2 \vdash \ldots \vdash \alpha_{p(n)}$$

dove α_0 descrive la situazione di partenza e $\alpha_{p(n)}$ è una ID accettante.

Possiamo immaginare queste ID strutturate in una tabella di dimensione $p(n) \times p(n)$, in quanto il numero di mosse eseguite da M nel caso peggiore sono p(n), e cosi la lunghezza della sequenza di ID. La lunghezza massima di una ID è anch'essa di p(n) posizioni.

Questa tabella avrà quindi la forma seguente:

α_0	X_{00}	X_{01}						$X_{0p(n)}$
α_1	X_{10}	X_{11}						$X_{1p(n)}$
α_i				X_{ij-1}	X_{ij}	X_{ij+1}		
α_{i+1}				X_{i+1j-1}	X_{i+1j}	X_{i+1j+1}		
$\alpha_{p(n)}$	$X_{p(n)0}$	$X_{p(n)1}$						$X_{p(n)p(n)}$

Indicheremo le posizioni nelle varie ID col nome di X_{ij}, dove i indica il numero della ID, ossia la riga della tabella, mentre j indica la posizione nella ID, ossia la colonna della tabella. Queste variabili X_{ij} possono contenere uno stato di M oppure un simbolo di nastro.

Dato che le istanze di SAT sono espressioni booleane, non possiamo usare direttamente le variabili X_{ij} nell'espressione che andremo a costruire. Useremo le variabili X_{ij} solo come riferimento, e descriveremo il loro contenuto attraverso variabili booleane y_{ijz}.

Il significato di y_{ijz} è il seguente: y_{ijz} vale 1 se e solo se la variabile X_{ij} nella nostra tabella contiene z, dove z può essere un simbolo di nastro o uno stato di M.

Useremo queste variabili booleane per costruire una espressione finale avente forma:

$$E_{M,w} = U \wedge S \wedge N \wedge F$$

soddisfacibile se e solo se M accetta w entro $p(n)$ mosse. Vedremo ora come costruire i vari componenti di questa espressione.

U: unicity

U è formato da l'AND logico di tutti i termini della forma $\neg(y_{ij\alpha} \wedge y_{ij\beta})$ con $\alpha \neq \beta$. Il numero di termini è $O(p^2(n))$.

Questa espressione impedisce che ogni posizione in ogni ID contenga più di un simbolo.

S: start ID

Questa espressione vincola la ID iniziale. La ID iniziale deve avere come primo simbolo q_0, lo stato iniziale, cui devono seguire i simboli di $w = a_1 a_2 a_3 \ldots a_n$. Le posizioni rimanenti devono contenere il simbolo di blank B.

L'espressione utilizzata per S è la seguente:

$$S = y_{00q_0} \wedge y_{01a_1} \wedge y_{02a_2} \wedge \ldots y_{0na_n} \wedge y_{0n+1B} \wedge \ldots y_{0p(n)B}$$

La lunghezza di S è $O(p(n))$.

F: final ID

Questa espressione verifica che lo stato della ID finale, in posizione $p(n)$, sia accettante.

Ricordiamo che abbiamo assunto per M un solo stato finale accettante q_1. Assumiamo inoltre che se lo stato q_1 è raggiunto in una ID precedente all'ultima, questa ID finale viene ripetuta senza alterazioni fino

alla posizione $p(n)$.

F è l'OR logico di $p(n)$ variabili:

$$F = y_{p(n)0q_1} \vee y_{p(n)1q_1} \vee y_{p(n)2q_1} \vee \ldots y_{p(n)p(n)q_1}$$

La lunghezza di F è $O(p(n))$.

N: next ID

L'espressione N assicura che, partendo da una ID valida, la ID successiva sia anch'essa valida.

N è formata da l'AND logico di una serie di espressioni N_i, con $i = 0, 1, 2, \ldots, p(n) - 1$. Ogni N_i garantisce che α_{i+1} sia una delle ID valide che possono seguire a α_i in M.

Le espressioni N_i si suddividono a loro volta in due sotto-espressioni:

$$N_i = A_{ij} \vee B_{ij}$$

con $j = 0, 1, 2, \ldots, p(n)$.

L'espressione B_{ij} è cosi formata:

$$\begin{aligned} B_{ij} =& (y_{ijz_1} \wedge y_{i+1jz_1}) \vee (y_{ijz_2} \wedge y_{i+1jz_2}) \vee \ldots \vee (y_{ijz_r} \wedge y_{i+1jz_r}) \vee \\ & (y_{ij-1q_1} \vee y_{ij-1q_2} \vee \ldots \vee y_{ij-1q_m}) \vee \\ & (y_{ij+1q_1} \vee y_{ij+1q_2} \vee \ldots \vee y_{ij+1q_m}) \end{aligned}$$

dove $q_1, q_2, \ldots, q_m$ sono gli stati di M, e $z_1, z_2, \ldots, z_r$ sono i simboli di nastro.

La prima riga della formula di B_{ij} verifica se la posizione X_{ij} nella ID α_i contiene un simbolo di nastro, e in tal caso verifica che questo simbolo venga trascritto senza modifiche nella stella posizione della ID successiva.

Le ultime due righe della formula per B_{ij} consentono un eccezione a quanto appena detto: se una delle due posizioni adiacenti (X_{ij-1} e

X_{ij+1}) a quella del simbolo analizzato (X_{ij}) contiene uno stato invece che un simbolo di nastro, queste sotto espressioni rendono vera B_{ij}.

Nel caso invece che X_{ij} contenga uno stato, la sotto-espressione B_{ij} risulterà falsa, e sarà' compito di A_{ij} rendere l'espressione vera.
È da notare che per i casi B_{i0} e $B_{ip(n)}$ sono necessarie alcune piccole modifiche: descrivendo queste i limiti di una ID alcune posizioni non sono accessibili. Lasciamo le modifiche allo studente.

Abbiamo detto che se uno stato è adiacente a un simbolo di nastro in una ID, quella parte di ID può non essere trascritta immutata nella ID successiva. In generale è possibile conoscere il simbolo X_{i+1j} nella ID $\alpha_i + 1$, osservando i tre simboli sopra di esso (X_{ij-1}, X_{ij} e X_{ij+1}).

La porzione di ID interessata è illustrata dalla tabella seguente:

α_i	...	X_{ij-1}	X_{ij}	X_{ij+1}	...
α_{i+1}	...	X_{i+1j-1}	X_{i+1j}	X_{i+1j+1}	...

Se la posizione X_{ij} contiene uno stato è la sotto-espressione A_{ij} che verifica la correttezza delle posizioni X_{i+1j-1}, X_{i+1j} e X_{i+1j+1} nella ID successiva.

La forma di A_{ij} è un OR di termini, un termine per ogni insieme di sei variabili che formano un assegnamento valido. Un assegnamento è valido se:

1. X_{ij} è uno stato, ma X_{ij-1} e X_{ij+1} sono simboli di nastro.

2. esiste una mossa in M per la quale X_{ij-1}, X_{ij} e X_{ij+1} diventano X_{i+1j-1}, X_{i+1j} e X_{i+1j+1}.

Per esempio, supponiamo in M sia presente una transizione:

$$\delta(q_2, z_1) = (q_3, z_2, L)$$

La sezione di nastro interessata subirà queste variazioni:

α_i	$\ldots$	z_3	q_2	z_1	$\ldots$
α_{i+1}	$\ldots$	q_3	z_3	z_2	$\ldots$

Questo si traduce nell'espressione:

$$y_{ij-1z_3} \ \wedge \ y_{ijq_2} \ \wedge \ y_{ij+1z_1} \ \wedge$$
$$y_{i+1jq_3} \ \wedge \ y_{i+1j-1z_3} \ \wedge \ y_{i+1j+1z_2}$$

In modo simmetrico è possibile costruire l'espressione nel caso la transizione muova la testina a destra e non a sinistra. Inoltre, come per le B_{ij}, bisogna modificare leggermente le A_{ij} che descrivono situazioni ai limiti di una ID.

A_{ij} è formato da l'OR logico di tutte le espressioni aventi la forma illustrata forma, che descrivono una possibile transizione di M.

Per quanto riguarda la lunghezza di A_{ij} e B_{ij}, questa può essere molto grande per macchine con numerosi stati e simboli di nastro, ma resta costante rispetto alla lunghezza dell'input w.

Di conseguenza la lunghezza di N_i è $O(p(n)$, mentre quella di N è $O(p^2(n)$.

Abbiamo descritto come convertire una istanza di un arbitrario $P \in \mathbf{NP}$ in una espressione istanza di SATdi forma:

$$E_{M,w} = U \wedge S \wedge N \wedge F$$

Questa espressione dipende sia da M che da w, anche se solo S dipende direttamente da w, mentre le altre parti dipendono solo da M de dalla lunghezza di w.

Quindi per ogni NTM che gira in tempo polinomiale $p(n)$ possiamo ideare un algoritmo che prende un input w di lunghezza n e produce $E_{M,w}$. Il tempo di esecuzione su una TM deterministica multinastro è di $O(p^2(n))$, convertibile in una TM a nastro singolo che impiega un tempo $O(p^4(n)$.

L'output dell'algoritmo è un espressione booleana soddisfacibile se e solo se M accetta l'input w. Quindi SAT è NP-completo.

□

Avendo aggiunto al nostro repertorio un problema NP-completo, possiamo dimostrare l'NP-completezza dei successivi problemi attraverso riduzioni polinomiali.

Esiste però un problema intermedio, detto 3SAT, che spesso risulta più semplice da ridurre ad alcune classi di problemi rispetto a SAT, perchè sfrutta espressioni aventi una forma molto precisa: congiunzioni logiche di clausole.

Vedremo qui di seguito come dimostrare che 3SATè NP-completo, e quale sia con precisione il formato delle istanze di 3SAT.

Definizione (Letterale)**.** Un letterale è una variabile booleana o una variabile booleana negata (e.g. x_1 o $\neg x_2$).

Definizione (Clausola)**.** Una clausola è la disgiunzione logica di uno o più letterali (e.g. $(x_1 \vee \neg x_2 \vee x_3)$).

Definizione (Espressione in forma normale congiuntiva)**.** Una espressione booleana si dice in forma normale congiuntiva (CNF, conjunctive normal form) se è la congiunzione logica di una o più clausole.
Un esempio di espressione in CNF è $(x_1 \vee \neg x_2 \vee x_3) \wedge x_4 \wedge (\neg x_1 \vee x_2)$.

Per arrivare a 3SATci appoggeremo a un problema intermedio, chiamato CSAT. Per far questo dobbiamo ridurre le istanze di SATa CSAT, che verifica la soddisfacibilità di espressioni in forma normale congiuntiva.

Sappiamo dalla logica che è sempre possibile convertire espressioni booleane generiche in espressioni equivalenti in CNF. Questo approccio però, in alcuni

casi, produce espressioni in CNF che sono esponenziali rispetto alla lunghezza delle espressioni generiche originali, e questo non è accettabile se si vuole ottenere una riduzione polinomiale.

In realtà i vincoli che dobbiamo soddisfare sono meno forti: è' sufficiente convertire un istanza E di SAT in un istanza F di CSAT, tale che F sia soddisfacibile se e solo se anche E lo è.

Questa è la strategia usata dalla riduzione che andremo a costruire.

Teorema. Per ogni espressione booleana E esiste una espressione booleana F in cui le negazioni compaiono solo nei letterali.

Inoltre la lunghezza di F è lineare rispetto alla lunghezza di E, ed è costruibile in tempo polinomiale.

Dimostrazione. Dimostriamo il teorema per induzione sul numero di operatori di E. Dimostriamo inoltre che se E contiene $n \geq 1$ operatori, l'espressione equivalente F ne contiene al più $2n - 1$.

Base Se E contiene un solo operatore è già in CNF.

Induzione Supponiamo l'enunciato sia vero per espressioni con meno operatori di E.

Rispetto all'operatore di livello più alto, la forma di E può essere:

$E = E_1 \vee E_2$ Per ipotesi induttiva esistono F_1 e F_2, equivalenti a E_1 e E_2, con negazioni solo nei letterali. Poniamo quindi $F = F_1 \vee F_2$.

Supponiamo E_1 e E_2 abbiano rispettivamente n_1 e n_2 operatori, e che quindi E ne abbia $n_1 + n_2 + 1$.

Per ipotesi induttiva F_1 e F_2 ne hanno al massimo $2n_1 - 1$ e $2n_2 - 1$. Perciò F ne ha al massimo $2(n_1 + n_2 + 1) - 1$.

$E = E_1 \wedge E_2$ Come visto sopra possiamo porre $F = (F_1) \wedge (F_2)$. Anche in questo caso F ha al massimo $2(n_1 + n_2 + 1) - 1$ operatori.

$E = \neg E_1$ Consideriamo allora l'operatore più esterno di E_1:

Se $E_1 = \neg E_2$ Allora $E = \neg(\neg(E_2))$. Per ipotesi induttiva possiamo trovare una F equivalente a E_2 che rispetta l'enunciato.

Se $E_1 = E_2 \vee E_3$, allora per De Morgan, $E = \neg(E_2 \vee E_3) = \neg(E_2) \wedge \neg(E_3)$.

Sia $\neg(E_2)$ che $\neg(E_3)$ per ipotesi induttiva hanno espressioni equivalenti F_2 e F_3 con negazioni solo nei letterali. Poniamo quindi $F = (F_2) \wedge (F_3)$.
Supponiamo E_2 e E_3 abbiano rispettivamente n_2 e n_3 operatori, e che quindi E ne abbia $n_2 + n_3 + 2$. Per ipotesi di induzione sappiamo che F_2 e F_3 ne hanno al massimo $2(n_2+1)-1$ e $2(n_3+1)-1$.

Quindi F non ne ha più di $2n_2 + 2n_3 + 3 = 2(n_2 + n_3 + 2) - 1$.
Se infine $E_1 = E_2 \wedge E_3$ si procede come nel caso precedente, applicando De Morgan.

□

Teorema. Sia E un espressione booleana di lunghezza n, le cui negazioni compaiono solo nei letterali. Allora esiste una espressione F tale che:

1. F è in CNF ed è formata da non più di n clausole.
2. F si può costruire da E in tempo non superiore a $c|E|^2$, con c costante.
3. un assegnamento di valori di verità T rende E vera se e solo se T può essere esteso a un assegnamento S che rende F vera.

Dimostrazione. Per induzione sul numero di simboli di E.

Base se E ha al massimo due simboli allora è un letterale e soddisfa già l'enunciato.

Induzione Ipotizziamo che ogni espressione più corta di E si possa convertire in un espressione che rispetti l'enunciato.

A seconda dell'operatore principale di E distinguiamo i seguenti casi:

1. $E = E_1 \wedge E_2$ Per ipotesi di induzione esistono due espressioni F_1 e F_2, derivate rispettivamente da E_1 e E_2, che rispettano l'enunciato.
   Senza perdere generalità assumiamo che gli insiemi di variabili di F_1 e F_2 siano disgiunti, salvo per le variabili che figurano in E.
   Costruiamo F come: $F = F_1 \wedge F_2$.
   F è in CNF. Dobbiamo dimostrare che un assegnamento di valori di verità T rende E vera se e solo se può essere esteso ad S che rende F vera.

   (a) Se T soddisfa E allora esiste S che soddisfa F, con S estensione di T:
   Sia T un assegnamento di valori di verità che soddisfa E. Siano T_1 e T_2 come T, ma ristretti a E_1 e E_2.
   Allora, per ipotesi induttiva, esistono S_1 e S_2, estensioni di T_1 e T_2, che soddisfano F_1 e F_2.
   Possiamo costuire S in accordo con S_1 e S_2, in quanto le variabili comuni di S_1 e S_2 hanno valori concordi. Poichè F è l'AND di F_1 e F_2, S soddisfa F.

   (b) Se S soddisfa F, allora esiste T che soddisfa E, con S estensione di T:
   Sia S un assegnamento di valori di verità che soddisfa F. SianoS_1 e S_2 le restrizioni di S rispettivamente su F_1 e F_2.
   Per ipotesi induttiva esistono T_1 e T_2 che soddisfano rispettivamente E_1 e E_2, con S_1 estensione di T_1 e S_2 estensione di T_2.
   Poichè F è l'AND di F_1 e F_2, allora S_1 e S_2 hanno variabili concordi. Allora anche T_1 e T_2 hanno variabili comuni concordi.
   Possiamo allora costruire T in accordo a T_1 e T_2.
   Essendo E l'AND di E_1 e E_2, T soddisfa E.

2. $E = E_1 \vee E_2$ Per ipotesi induttiva esistono F_1 e F_2 che rispettano l'enunciato rispetto a E_1 e E_2. Supponiamo $F_1 = g_1 \wedge g_2 \wedge \ldots \wedge g_p$ e $F_2 = h_1 \wedge h_2 \wedge \ldots \wedge h_q$, dove le g e le h sono clausole.

Introduciamo una nuova variabile y e costruiamo F come:

$$\begin{aligned} F \ = & (y \vee g_1) \ \wedge \ (y \vee g_2) \ \wedge \ \ldots \ \wedge \ (y \vee g_p) \ \wedge \\ & (\neg y \vee h_1) \ \wedge \ (\neg y \vee h_2) \ \wedge \ \ldots \ \wedge \ (\neg y \vee h_q) \end{aligned}$$

Dobbiamo dimostrare che un assegnamento di valiri di verità T soddisfa E se e solo se S soddisfa F, con S estensione di T.

(a) *Solo se*: Supponiamo T soddisfi E. Siano T_1 e T_2 come E, ma ristretti a E_1 e E_2.

Poichè $E = E_1 \vee E_2$, T soddisfa almeno uno tra E_1 e E_2. Supponiamo soddisfi E_1.

Allora per ipotesi induttiva T_1 può essere esteso a S_1 che soddisfa F_1.

Costruiamo S per soddisfare F come:

i. per ogni variabile x di F_1 imponiamo $S(x) = S_1(X)$
ii. $S(y) = 0$, per rendere vere le clausole di F_2
iii. per ogni variabile x di F_2 che non appartiene a F_1, $S(x) = T(x)$ se definito, oppure assegnamo un valore arbitrario a $S(x)$.

Con queste regole tutte le clausole di F sono soddisfatte, quindi S soddisfa F.

Se T soddisfa solo E_2, la procedura di dimostrazione è simile, imponendo $S(y) = 1$.

(b) *Se*: Sia T un assegnamento di valori di verità per E, sia S un estensione di T per F e supponiamo S soddisfi F. Dobbiamo dimostrare che T soddisfa E.

Abbiamo due casi: $S(y)$ può valere 1 o 0.

Supponiamo $S(y) = 0$. Allora le clausole derivate dalle h sono vere.

Dato che $S(y) = 0$, allora S deve rendere vere le g_i nelle clausole $(y \vee g_i)$ di F_1.

Sia S_1 come S ma ristretto a F_1. Dato che S_1 soddisfa F_1, per ipotesi induttiva T_1, la ristrizione di T a E_1, soddisfa E_1. Allora T soddisfa $E = E_1 \vee E_2$.

Il caso $S(y) = 1$ è simmetrico.

Dobbiamo ora dimostrare che il tempo necessario per costruire F da E è al massimo quadratico in $n = |E|$.

La separazione di E in E_1 e E_2 e la costruzione di F da F_1 e F_2 richiedono tempo lineare rispetto a $|E|$.

Sia dn un limite superiore di questo tempo, nel caso 1 e nel caso 2.

Abbiamo equazioni di ricorrenza:

$$\begin{aligned} T(1) &= T(2) \leq e \textit{ costante} \\ T(n) &\leq dn + c \max_{0 \leq i \leq n-1} (T(i) + T(n-1-i)) \end{aligned}$$

con c tale che $T(n) \leq cn^2$.

Per una forma del teorema master $T(n) = O(n^2)$.

□

Teorema (NP-completezza di CSAT)**.** CSAT è NP-completo.

Dimostrazione. CSAT è un sottoinsieme di SAT, percui è possibile adattare la NTM di SAT per accettare CSAT in tempo polinomiale nondeterministico. Quindi CSAT è in **NP**.

Dobbiamo ora dimostrare che esiste una riduzione polinomiale da SAT a CSAT.

Come primo passo prendiamo una generica istanza E di SAT e trasformiamola in una espressione F equivalente con tutte le negazioni nei letterali. Abbiamo dimostrato che ciò è possibile in tempo lineare rispetto alla lunghezza di E.

Dobbiamo ora trasformare F in un espressione G in CNF, tale che G è soddisfacibile se e solo se F è soddisfacibile.

Per fare questo applichiamo l'ultimo teorema dimostrato, che soddisfa condizioni più forti di quelle che ci occorrono in questo caso.

La costruzione di G è possibile in tempo polinomiale, quindi CSAT è NP-completo.

□

Teorema (NP-completessa di 3SAT)**.** 3SATè NP-completo.

Dimostrazione. Possiamo utilizzare la stessa NTM che congettura assegnamenti alle variabili di SAT per dimostrare che 3SAT è in **NP**.

Per dimostrare che 3SATè anche NP-completo riduciamo CSAT a 3SAT.

Data una espressione in CNF $E = e_1 \wedge e_2 \wedge \ldots \wedge e_k \wedge$ formiamo una nuova espressione F sostituendo le clausole e_i in base alla loro lunghezza:

1. Se e_i è formata da un singolo letterale x, introduciamo due nuove variabili u e v.

   Sostituiamo la clausola e_i con:

   $$(x \vee u \vee v) \wedge x \vee u \vee \neg v) \wedge x \vee \neg u \vee v) \wedge x \vee \neg u \vee \neg v)$$

   Poichè u e v compaiono in tutte le combinazioni, l'unico modo per soddisfare tutte le clausole è rendere vera x.

   Di conseguenza tutti gli assegnamenti che soddisfano E, e solo quelli, si possono estendere ad assegnamenti che soddisfano F.

2. Sia e_i la somma di due letterali $(x \vee y)$. Introduciamo z e trasformiamo e_i come:

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

Come nel caso (1), il solo modo per soddisfare le due clausole è soddisfare $(x \vee y)$.

3. Se e_i è la somma di 3 letterali, allora è già nella forma richiesta.

4. Sia $e_i = (x_1 \vee x_2 \vee \ldots \vee x_m)$, con $m \geq 4$. Introduciamo $y_1, y_2, \ldots, y_{m-3}$. Sostituiamo e_i con:

$$\begin{array}{l} (x_1 \ \vee \ \ x_2 \ \ \vee \ y_1) \wedge \\ (x_3 \ \vee \ \neg y_1 \ \vee \ y_2) \wedge \\ (x_4 \ \vee \ \neg y_2 \ \vee \ y_3) \wedge \\ \ldots \\ (x_{m-2} \ \vee \ \neg y_{m-4} \ \vee \ y_{m-3}) \wedge \\ (x_{m-1} \ \vee \ \ x_m \ \ \vee \ \neg y_{m-3}) \end{array}$$

Un assegnamento T che soddisfa E deve rendere vero almeno un letterale di e_i. Sia x_j questo letterale.

Se dichiariamo $y_1, y_2, \ldots, y_{j-2}$ vere e $y_{j-1}, y_j, \ldots, y_{m-3}$ false, allora tutte le clausole della nuova espressione risultano vere. Perciò T può essere estesa per saddisfarle.

Viceversa se in T tutte le x sono false, non esiste una estensione di T che può soddisfare l'espressione che abbiamo costruito.

Abbiamo mostrato come costruire F a partire da E, in modo che F sia soddisfacibili se e solo se anche E lo è.

La costruzione impiega tempo lineare, in quanto il caso più costoso, il primo, allunga la lunghezza della sotto-espressione e_i di un fattore costante (pari a 32/3).

□

Nota (Presentare un problema NP-completo)**.** Nel presentare nuovi problemi NP-completi adottiamo il seguente schema di definizione:

1. **Nome** del problema, spesso associato ad una abbreviazione
2. **Input** del problema
3. **Output** del problema, ossia quando deve essere "si"
4. **Problema** da cui si compie la riduzione

Definizione (Insieme indipendente)**.** Sia G un grafo non orientato. Un sottoinsieme I di nodi di G si dice indipendente se nessuna coppia di nodi di I è collegata da un lato di G.

Definizione (Insieme indipendente massimale)**.** Un insieme indipendente di G si dice massimale se ha almeno tanti nodi quanti ogni altro insieme indipendente di G.

Definizione (Problema dell'insieme indipendente, IS)**.** Definiamo il problema dell'insieme indipendente con lo schema appena indicato:

1. **Nome**: Insieme indipendente (IS, indipendent set).
2. **Input**: Un grafo G e un limite inferiore k, con k compreso fra 1 e il numero di nodi di G.
3. **Output**: "Si" se e solo se G ha un insieme indipendente di k nodi.
4. **Riduzione da**: 3SAT.

Teorema (NP-completezza di IS)**.** IS è NP-completo.

Dimostrazione. IS è in **NP** perchè dato un grafo G e un intero k, una NTM può congetturare un insieme di k nodi di G e verificare se sono indipendenti, in tempo polinomiale.

Riduciamo ora 3SAT a IS.

Sia $E = (e_1) \wedge (e_2) \wedge \ldots \wedge (e_m)$ un espressione in 3-CNF. A partire da E costruiamo un grafo G con $3m$ nodi. A questi nodi diamo nome $[i, j]$, con $1 \leq i \leq m$ e $j = 1, 2, 3$. Il nodo $[i, j]$ rappresenta il j-esimo letterale della clausola e_i.

Per quanto riguarda i lati di G, seguiamo 2 regole:

1. vogliamo che si possa scegliere al massimo un nodo per clausola; allora uniamo con un lato tutte le coppie di nodi derivati dalla stessa clausola di E.

2. dobbiamo impedire che nodi che rappresentano letterali complementari (i.e. x e $\neg x$) siano entrambi nell'insieme indipendente; uniamo allora con un lato tutti i nodi che rappresentano letterali complementari.

Il limite k, per G costruito secondo queste due regole, è m.

La costruzione del grafo G sopra descritta è possibile in tempo quadratico rispetto alla lunghezza di E.

Rimano ora da dimostrare che E è soddisfacibile se e solo se G ha un insieme indipendente di m nodi.

Se : Sia I un insieme indipendente di m nodi di G.

Per la regola (1), I non può contenere più di un nodo per clausola, e avendo m nodi ne ha esattamente uno per clausola.

I non può nemmeno avere al suo interno nodi che rappresentano letterali complementari, perchè, per la regola (2), altrimenti non sarebbe indipendente.

Pertanto I genera un assegnamento di valori di verità che soddisfa E. Definiamo $T(x) = 1$ se il nodo associato a x è in I, $T(x) = 0$ altrimenti.

Dato che ogni clausola è resa vera da un letterale, T soddisfa E.

Solo se : Sia T un assegnamento di valori di verità che soddisfa E.

Allora T rende vero almeno un letterale per ogni clausola di E.

Formiamo I scegliendo uno solo dei letterali veri da ogni clausola di E. Dobbiamo dimostrare che I è indipendente.

Se un lato unisce due nodi derivati dalla stessa clausola, questo nodi

non possono essere entrambi in I, per come abbiamo sclto i nodi di I.

Se un lato unisce una variabile alla sua negata, non possono trovarsi entrambe in I, perchè abbiamo scelto solo nodi corrispondenti a letterali veri rispetto a T.

Quindi se E è soddisfacibile, allora G ha un insieme indipendente di dimensione m.

□

Definizione (Copertura per nodi)**.** Si definisce copertura per nodi di un grafo G, un sottoinsieme di nodi C di G, tale che ogni lato di G ha almeno un estremo in C.

Definizione (Copertura per nodi minimale)**.** Una copertura per nodi C di un grafo G si dice minimale, se ogni altra copertura di G non ha un numero di nodi inferiore quelli di C.

Definizione (Problema della copertura per nodi)**.** Schema per il problema della copertura per nodi:

1. **Nome**: Copertura per nodi (NC, node cover).
2. **Input**: Un grafo G e un limite superiore k, con k compreso fra 0 e il numero di nodi di G meno uno.
3. **Output**: "Si" se e solo se G ha una copertura per nodi di non più di k nodi.
4. **Riduzione da**: IS.

Teorema (NP-completezza di NC)**.** NC è NP-completo.

Dimostrazione. NC è in **NP**, dato che una NTM è in grado di congetturare un insieme di k nodi di un grafo G e verificare se è una copertura per nodi, in tempo polinomiale.

Dobbiamo ora ridurre IS a NC.

Sappiamo gi'ạlla teoria dei grafi che il complemento di un insieme indipendente è una copertura per nodi per lo stesso grafo.

Dati G e k, con G avente n nodi, gli elementi di una istanza di IS, siano G e $n-k$ gli elementi dell'istanza di NC che vogliamo costruire. Questa costruzione è possibile in tempo lineare.

Proviamo che G ha un insieme indipendente di dimensione k se e solo se G ha una copertura per nodi di dimensione $n-k$:

Se : Sia N l'insieme di nodi di G, e C la copertura per nodi di G avente dimensione $n-k$.

Affermiamo allora che $N-C$ è un insieme indipendente.

Per assurdo supponiamo che i nodi v e w in $N-C$ siano uniti da un lato.

Poichè ne v ne w sono in C, il lato (v,w) di G non è coperto da C.
Ma questo è assurdo perchè per ipotesi C è una copertura per nodi di G.

Solo se : Sia I un insieme indipendente di G avente k nodi.

Affemiamo che $N-I$ è una copertura per nodi di G di $n-k$ nodi.

Per assurdo supponiamo il lato (v,w) non sia coperto da $N-I$.
Allora v e w sono in I, ma questo è assurdo perchè I è indipendente per ipotesi.

□