

# Complementi di Algoritmi

Massimo Perego

# Indice

<b>1 Complessità Computazionale</b>	<b>2</b>
1.1 Riducibilità polinomiale tra problemi di decisione . . . . .	3
1.2 Classi $\mathcal{P}$ e $\mathcal{NP}$ . . . . .	8
1.3 Problemi $\mathcal{NP}$ -completi . . . . .	9

## Capitolo 1

# Complessità Computazionale

## 1.1 Riducibilità polinomiale tra problemi di decisione

Un **problema di decisione**  $X$  su  $\{0, 1\}$  è una coppia  $(\mathcal{I}, q)$  dove

- $\mathcal{I} \subseteq \{0, 1\}^*$  è l'insieme delle descrizioni (come stringhe binarie) delle **istanze valide del problema**
- $q : \mathcal{I} \rightarrow \{0, 1\}$  è la **funzione di decisione**, che associa ogni istanza  $I \in \mathcal{I}$  al suo valore, ovvero  $q(I) = 1$  se positiva o  $q(I) = 0$  se negativa

Un algoritmo  $A$  risolve un problema  $X = (\mathcal{I}, q)$  se restituisce il valore corretto della funzione  $q$  su ogni istanza  $I \in \mathcal{I}$ .

Indichiamo con  $|I|$  la **lunghezza della codifica** in bit di un'istanza  $I \in \mathcal{I}$ . In genere, non specifichiamo il modo con il quale un'istanza viene rappresentata come una stringa di bit, ma resta inteso che le istanze vengano rappresentate secondo rappresentazioni “ragionevoli” (e.g. lista di adiacenza per un grafo).

Se  $A$  è un algoritmo che risolve il problema  $X = (\mathcal{I}, q)$ , indichiamo con  $T_A(I)$  il tempo di calcolo di  $A$  quando l'input è  $I \in \mathcal{I}$ . Definiamo ora

$$T_A(n) = \max \{T_A(I) \mid |I| \leq n\}$$

ovvero il massimo dei tempi di calcolo per istanze  $I \in \mathcal{I}$  di lunghezza al più  $n$ . Diciamo che  $A$  risolve  $X$  in tempo polinomiale se esiste un intero  $k$  tale che  $T_A(n) = \mathcal{O}(n^k)$ .

Un algoritmo ha accesso a un **oracolo** per un problema  $X = (\mathcal{I}, q)$  quando può ottenere in tempo unitario il valore  $q(I)$  per istanze arbitrarie  $I \in \mathcal{I}$ .

Un problema  $Y$  è **polynomialmente riducibile** a un altro problema  $X$ , denotato con  $Y \preceq_p X$  quando

1. Esiste un algoritmo che risolve  $Y$  in tempo polinomiale con accesso a un oracolo per  $X$
2. L'oracolo viene interrogato una sola volta e l'algoritmo termina fornendo la risposta dell'oracolo

Questa nozione di riducibilità polinomiale è anche nota come riducibilità secondo Karp. Esiste anche una nozione più generale, nota come riducibilità secondo Turing, dove l'algoritmo può accedere all'oracolo un numero polinomiale di volte.

Equivalentemente, la riducibilità polinomiale (secondo Karp) può essere vista come “l’input di  $Y$  può essere trasformato in tempo polinomiale in input di  $X$ ” (per poi risolvere il problema in tempo unitario grazie all’oracolo, coincide con la prima definizione).

Se  $Y \preceq_p X$  ed esiste un algoritmo che risolve  $X$  in tempo polinomiale, allora esiste un algoritmo che risolve  $Y$  in tempo polinomiale (la polinomialità è robusta rispetto alla composizione). Viceversa, se  $Y \preceq_p X$  e non esiste un algoritmo che risolve  $Y$  in tempo polinomiale, allora non esiste un algoritmo che risolve  $X$  in tempo polinomiale. Intuitivamente,  $Y \preceq_p X$  significa che il problema  $Y$  non è più *difficile* del problema  $X$ . Se  $X \preceq_p Y$  e  $Y \preceq_p X$ , ovvero  $X$  e  $Y$  sono polinomialmente riducibili l’uno con l’altro, allora scriviamo  $X \equiv_p Y$ .

Sia  $G = (V, E)$  un grafo semplice (non diretto, non pesato, senza loop e archi multipli). Un **insieme indipendente** (independent set) in  $G$  è un sottoinsieme  $S \subseteq V$  di vertici non adiacenti, ovvero  $\forall i, j \in S, (i, j) \notin E$ . Una **copertura** (vertex cover) di  $G$  è un sottoinsieme  $S' \subseteq V$  di vertici tali che ogni arco di  $G$  ha almeno un estremo in  $S'$ , ovvero  $\forall (i, j) \in E$  esiste  $k \in S'$  tale che  $k = i$  o  $k = j$ .

**Fatto 1.1.1.** *Sia  $G = (V, E)$  un grafo semplice. Allora  $S$  è un insieme indipendente se e solo se  $S' = V \setminus S$  è una copertura.*

*Dimostrazione.* Sia  $S$  un insieme indipendente e sia  $S' = V \setminus S$ . Allora dato un qualsiasi  $(i, j) \in E$ , deve valere  $i \notin S$  oppure  $j \notin S$ . Quindi  $i \in S'$  oppure  $j \in S'$ , da cui ne segue che  $S'$  è una copertura di  $G$ .

Viceversa, sia  $S'$  una copertura di  $G$  e  $S = V \setminus S'$ . Allora dati  $i, j \in S$  arbitrari  $(i, j) \notin E$ , altrimenti allora  $S'$  non sarebbe una copertura; da questo otteniamo che  $S$  è un independent set.  $\square$

Sia *Independent Set* il problema di decisione le cui istanze sono coppie  $I = (G, k)$  dove  $G$  è un grafo semplice e  $k$  è un intero. La funzione di decisione  $q_{IS}$  è tale che  $q_{IS}(I) = 1$  se e solo se  $G$  contiene un insieme indipendente di taglia almeno  $k$ . Le istanze del problema **Vertex Cover** sono le stesse di Independent Set, ma la funzione di decisione  $q_{VC}$  è tale che  $q_{VC}(I) = 1$  se e solo se  $G$  contiene una copertura di taglia al più  $k$ . Conseguenza immediata del Fatto 1.1.1 è che Independent Set e Vertex Cover sono polinomialmente riducibili l’uno con l’altro.

**Corollario 1.1.1.** *Independent Set  $\equiv_p$  Vertex Cover*

La relazione  $\equiv_p$  gode di transitività.

**Fatto 1.1.2.** *Se  $Z \preceq_p Y$  e  $Y \preceq_p X$ , allora  $Z \preceq_p X$ .*

*Dimostrazione.* Dato un oracolo per  $X$  possiamo risolvere un'istanza di  $Z$  nel modo seguente: eseguiamo l'algoritmo per risolvere  $Z$  usando l'oracolo per  $Y$ , ma ogni volta che dovrebbe essere invocato l'oracolo di  $Y$ , questo viene simulato eseguendo l'algoritmo per risolvere  $Y$ , il quale usa l'oracolo per  $X$ .  $\square$

Introduciamo ora il problema di decisione **Set Cover**: le istanze sono della forma  $I = (U, \mathcal{S}, k)$  dove  $U$  è un insieme finito,  $\mathcal{S} \equiv \{S_1, \dots, S_m\}$  è una collezione di sottoinsiemi di  $U$  e  $k$  è un intero. Allora  $qSC(I) = 1$  se e solo se esistono al più  $k$  sottoinsiemi in  $\mathcal{S}$  tali che la loro unione sia tutto  $U$ .

**Fatto 1.1.3.** *Vertex Cover  $\preceq_p$  Set Cover*

*Dimostrazione.* Supponiamo di avere un oracolo per Set Cover e definiamo un algoritmo polinomiale per Vertex Cover.

Data un'istanza  $I = (G, k)$  di Vertex Cover con  $G = (V, E)$  costruiamo un'istanza  $I'$  di Set Cover dove  $U \equiv E$  e  $\mathcal{S} \equiv \{S_i \mid i \in V\}$  dove  $S_i$  è l'insieme degli archi di  $G$  incidenti su  $i$  (ottenibile in tempo lineare partendo dalla descrizione di  $G$ ). Quindi ogni  $u \in U$  è contenuto in esattamente due elementi di  $\mathcal{S}$ .

Verifichiamo ora che  $U$  è l'unione di al più  $k$  insiemi  $S_1, \dots, S_{|V|}$  se e solo se  $G$  ha una copertura con vertici di taglia al più  $k$ .

Supponiamo che l'unione di  $S_{i_1}, \dots, S_{i_r}$  sia  $U$ , con  $r \leq k$ . Allora, ogni arco in  $G$  è incidente a uno dei vertici  $i_1, \dots, i_r$ . Quindi  $\{i_1, \dots, i_r\}$  è una copertura con vertici di taglia al più  $k$ . Viceversa, se  $\{i_1, \dots, i_r\}$  è una copertura con vertici di taglia al più  $k$  allora la gli insiemi corrispondenti  $S_{i_1}, \dots, S_{i_r}$  hanno  $U$  come unione.

Quindi possiamo definire un algoritmo che implementa la funzione di decisione  $qVC$  per Vertex Cover nel modo seguente: data un'istanza di  $I$  Vertex Cover, l'algoritmo costruisce in tempo polinomiale un'istanza  $I'$  di Set Cover nel modo descritto sopra. Quindi chiama l'oracolo un'unica volta per ottenere  $qSC(I')$  che viene restituita in output.  $\square$

Così come possiamo vedere Set Cover come una generalizzazione di Vertex Cover, introduciamo ora Set Packing come generalizzazione di Independent Set. Le istanze di **Set Packing** sono le stesse di Set Cover, ovvero  $I = (U, \mathcal{S}, k)$  dove  $U$  è un insieme finito,  $\mathcal{S} \equiv \{S_1, \dots, S_m\}$  è una collezione di sottoinsiemi di  $U$  e  $k$  è un intero. La funzione di decisione  $q_{SP}$  è tale che  $q_{SP} = 1$  se e solo se esistono almeno  $k$  sottoinsiemi in  $\mathcal{S}$  i quali sono a due a due disgiunti. Anche se Set Packing è apparentemente più generale di Independent Set, si può dimostrare che i due problemi sono equivalenti.

**Fatto 1.1.4.** *Independent Set  $\equiv_p$  Set Packing.*

*Dimostrazione.* Data un'istanza  $\mathcal{S} \equiv \{S_1, \dots, S_m\}$  di Set Packing costruiamo in tempo polinomiale un grafo  $G = (V, E)$  dove  $V \equiv \{v_S \mid S \in \mathcal{S}\}$  e  $(v_S, v_T) \in E$  se e solo se  $S \cap T \neq \emptyset$ . Allora ogni insieme indipendente in  $G$  corrisponde a un packing della stessa taglia.

Viceversa, dato un grafo  $G = (V, E)$  possiamo costruire in tempo polinomiale la collezione  $\mathcal{S} \equiv \{S_i \mid i \in V\}$  dove  $S_i$  è l'insieme degli archi di  $G$  incidenti su  $i$ . Allora ogni packing in  $\mathcal{S}$  corrisponde a un insieme indipendente in  $G$  della stessa taglia.  $\square$

In molte discipline bisogna spesso risolvere problemi di ottimizzazione combinatoria vincolata. In questi problemi si cerca un assegnamento di valori per un insieme di variabili discrete in modo da soddisfare un dato insieme di vincoli. In astratto, problemi di questo tipo sono formulati come problemi di soddisfacibilità su variabili booleane.

Sia  $\mathcal{X}$  un insieme di variabili booleane  $x_1, \dots, x_n$ . Un assegnamento di valori di verità a  $\mathcal{X}$  è una funzione  $\pi : \mathcal{X} \rightarrow \{0, 1\}$ . Un letterale  $\ell_i$  è la variabile  $x_i$  o la sua negazione  $\bar{x}_i$ . Una clausola  $C = \ell_{i_1} \vee \dots \vee \ell_{i_k}$  è una disunione di letterali. Un assegnamento  $\pi$  soddisfa una clausola  $C$  se e solo se c'è almeno un letterale della forma  $\ell_i = x_i$  e  $\pi(x_i) = 1$  oppure  $\ell_j = \bar{x}_j$  e  $\pi(x_j) = 0$ .

Sia **SAT** il problema di decisione le cui istanze  $I$  sono insiemi di clausole  $\mathcal{C}$  su un insieme  $\mathcal{X}$  di variabili booleane. Allora  $q(I) = 1$  se e solo se esiste un assegnamento  $\pi : \mathcal{X} \rightarrow \{0, 1\}$  tale che soddisfi tutte le clausole in  $\mathcal{C}$ .

Una versione ridotta di SAT è **3-SAT**, le cui istanze sono insiemi di clausole ciascuna contenente esattamente 3 letterali. Esempio di istanza di 3-SAT:

- $\mathcal{X} \equiv \{x_1, x_2, x_3, x_4\}$

- $\mathcal{C} \equiv \{(\bar{x}_1 \vee x_2 \vee \bar{x}_3), (\bar{x}_2 \vee x_3 \vee x_4), (x_1 \vee x_2 \vee \bar{x}_4)\}$

Un assegnamento che soddisfa tutte le clausole è  $\pi(x_i) = 0$  per  $i = 1, \dots, 4$ .

**Teorema 1.1.1.**  $3\text{-SAT} \preceq_p \text{Independent Set.}$

*Dimostrazione.* Una definizione equivalente di 3-SAT:  $\mathcal{C}$  è soddisfacibile se e solo se è possibile scegliere esattamente un letterale in ciascuna clausola in modo tale che tra i letterali scelti non compaiano simultaneamente  $x_i$  e  $\bar{x}_i$  per nessuna delle variabili  $x_i \in \mathcal{X}$ .

Data un'istanza  $I = \mathcal{C}$  di 3-SAT, costruiamo un'istanza  $I' = (G, |\mathcal{C}|)$  di Independent Set tale che  $q_{3\text{-SAT}}(I) = q_{IS}(I')$ . Sia  $k = |\mathcal{C}|$ . Il grafo  $G = (V, E)$  ha  $3k$  vertici, uno per ogni letterale in  $\mathcal{C}$ . I tre vertici corrispondenti ai letterali di ciascuna clausola formano una clique. Inoltre, per ogni  $x_i \in \mathcal{X}$ , se  $x_i$  e  $\bar{x}_i$  sono letterali in clausole distinte allora c'è un arco in  $G$  fra i vertici corrispondenti. Si noti che possiamo costruire  $G$  a partire da  $\mathcal{C}$  in tempo polinomiale rispetto a  $|\mathcal{C}|$ .

Per costruzione di  $G$ , l'unico modo per ottenere un insieme indipendente di taglia  $k$  nel grafo  $G$  è quello di scegliere un nodo in ciascuna delle  $k$  clique in modo che non ci siano archi fra il nodo scelto in una clique e nodi scelti appartenenti a un'altra clique. Per costruzione, questo avviene se e solo se è possibile scegliere esattamente un letterale in ciascuna clausola in modo tale che tra i letterali scelti non compaiano simultaneamente  $x_i$  e  $\bar{x}_i$  per nessuna delle variabili  $x_i \in \mathcal{X}$ .

Possiamo quindi costruire un algoritmo che risolve 3-SAT in tempo polinomiale usando un oracolo per Independent Set: l'algoritmo riceve un'istanza  $I$  di 3-SAT, costruisce l'istanza corrispondente  $I'$  di Independent Set e accede all'oracolo ottenendo  $q_{IS}(I')$ , per poi restituire  $q_{IS}(I')$ .  $\square$

## 1.2 Classi $\mathcal{P}$ e $\mathcal{NP}$

Sia  $\mathcal{P}$  la classe dei problemi di decisione  $X$  risolvibili in tempo polinomiale. Ovvero, per ogni problema  $X \in \mathcal{P}$  esiste un algoritmo che lo risolve in tempo polinomiale nella lunghezza delle istanze.

La funzione di decisione  $q$  di un problema  $X$  caratterizza una determinata proprietà delle sue istanze (per esempio, quei grafi che contengono un insieme indipendente abbastanza grande). Un **certificatore polinomiale** per  $X$  è un algoritmo  $B : \mathcal{I} \times \{0, 1\}^* \rightarrow \{0, 1\}$  tale che

1. Esiste un polinomio  $p(\cdot)$  tale che, per ogni istanza  $I \in \mathcal{I}$ ,  $q(I) = 1$  se e solo se esiste una stringa  $z \in \{0, 1\}^{p(|I|)}$  tale che  $B(I, z) = 1$
2.  $B$  termina in tempo polinomiale in  $|I|$  e  $|z|$

Possiamo pensare alla stringa  $z$  come un certificato del fatto che  $q(I) = 1$ . Per esempio, nel problema Independent Set la stringa  $z$  denota il sottinsieme di vertici che costituisce un insieme indipendente di cardinalità almeno  $k$ . Nel problema SAT, la stringa  $z$  denota un assegnamento che soddisfa tutte le clausole.

L'accesso a un certificatore polinomiale permette di verificare rapidamente se una stringa  $z$  è un certificato valido per un'istanza  $I$  di un problema. D'altra parte, se volessimo usare il certificatore per trovare un certificato qualora esso esista, ovvero stabilire il valore di  $q(I)$ , saremmo obbligati a eseguire  $B(I, z)$  esaustivamente su tutte le  $2^{p(|I|)}$  stringhe  $z$  tali che  $|z| \leq p(|I|)$ .

Introduciamo ora la classe  $\mathcal{NP}$  di problemi di decisione  $X$  che posseggono un certificatore polinomiale. Si noti che  $\mathcal{P} \subseteq \mathcal{NP}$ . Infatti, se  $X \in \mathcal{P}$  allora esiste un algoritmo che calcola la funzione di decisione  $q$  in tempo polinomiale in  $|I|$ . Possiamo usare questo algoritmo per implementare un certificatore polinomiale  $B$  come segue: dati  $I, z \in \mathcal{I} \times \{0, 1\}^*$ ,  $B$  restituisce  $q(I)$  ignorando  $z$ .

Quindi se  $q(I) = 1$ , abbiamo che  $B(I, z) = 1$  per ogni  $z \in \{0, 1\}^*$  (in particolare per  $z$  limitati in lunghezza da un polinomio in  $|I|$ ). Invece, se  $q(I) = 0$ , allora  $B(I, z) = 0$  per ogni  $z \in \{0, 1\}^*$ .

Anche se risolvere un'istanza di un problema in  $\mathcal{P}$  non significa necessariamente trovare un certificato  $z$ , possiamo comunque interpretare  $\mathcal{NP}$  come la classe che contiene quei problemi la cui soluzione (o certificato) è *verificabile* in tempo polinomiale, in contrasto con i problemi in  $\mathcal{P}$ , i quali problemi sono *risolvibili* in tempo polinomiale.  $\mathcal{P} \equiv \mathcal{NP}$  è un problema aperto.

### 1.3 Problemi $\mathcal{NP}$ -completi

Un problema di decisione  $X$  è  $\mathcal{NP}$ -completose  $x \in \mathcal{NP}$  e per qualunque  $Y \in \mathcal{NP}$  vale  $Y \preceq_p X$ . Intuitivamente, i problemi di decisione  $\mathcal{NP}$ -completi sono i “più difficili” in  $\mathcal{NP}$ , infatti vale il seguente.

**Fatto 1.3.1.** *Sia  $X$  un qualunque problema  $\mathcal{NP}$ -completo. Allora  $X \in \mathcal{P}$  se e solo se  $P \equiv \mathcal{NP}$ .*

*Dimostrazione.* Se  $P \equiv \mathcal{NP}$  allora  $X \in \mathcal{P}$ . D’altra parte, se  $X$  è  $\mathcal{NP}$ -completo, allora dato un qualunque  $Y \in \mathcal{NP}$  vale  $Y \preceq_p X$ . Per ipotesi,  $X$  è risolvibile in tempo polinomiale, di conseguenza lo è  $Y$ , quindi  $Y \in \mathcal{P}$ , da cui otteniamo  $\mathcal{NP} \subseteq \mathcal{P}$ . Abbiamo già dimostrato come  $\mathcal{P} \subseteq \mathcal{NP}$ , quindi  $\mathcal{P} \equiv \mathcal{NP}$ .  $\square$

Non è chiaro che esistano problemi  $\mathcal{NP}$ -completi. Potrebbero esserci due problemi  $X', X'' \in \mathcal{NP}$  tali che non esiste nessun altro  $X \in \mathcal{NP}$  tale che  $X' \preceq_p X$  e  $X'' \preceq_p X$ . Oppure, potrebbe esserci una sequenza infinita di problemi in  $\mathcal{NP}$  del tipo  $X_1 \preceq_p X_2 \preceq_p \dots$  tale che non esista un problema più difficile degli altri.

Un **circuito** è un grafo diretto aciclico (senza loop, archi multipli e pesi) avente un unico nodo senza archi uscenti chiamato nodo di output. I nodi senza archi entranti possono avere un valore di verità preassegnato. I nodi senza archi entranti e senza valori preassegnati sono chiamati nodi di input. I nodi rimanenti hanno uno o due archi entranti e sono etichettati da un operatore booleano  $\wedge, \vee, \neg$  in modo tale che nodi  $\wedge$  e  $\vee$  abbiano esattamente due archi entranti e nodi  $\neg$  abbiano esattamente un arco entrante.

Un circuito calcola una funzione booleana  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  dove  $n$  è il numero dei nodi di input. Dato un assegnamento  $(b_1, \dots, b_n) \in \{0, 1\}^n$  di valori di verità ai nodi di input, calcoliamo  $f(b_1, \dots, b_n)$  come il valore di verità del nodo di output ottenuto valutando in cascata i valori di verità di ciascun nodo. La valutazione di un nodo avviene applicando l’operatore logico che lo etichetta ai valori di verità dei nodi all’altro capo degli archi entranti. Se la valutazione di ogni nodo avviene in tempo costante, allora l’intero circuito viene valutato in tempo lineare nel numero dei nodi.

Sostanzialmente, si tratta del parsing tree di una funzione booleana.

Il problema di decisione **Circuit Satisfiability** (CS) ha istanze che rappresentano circuiti. La funzione di decisione  $q$  è tale che  $q(I) = 1$  se e solo

se esiste un assegnamento ai nodi di input del circuito  $I$  tale che il nodo di output assume valore 1. In altre parole,  $q(I) = 1$  se e solo se la funzione  $f$  calcolata dal circuito è tale che  $f(b_1, \dots, b_n) = 1$  per un qualche  $(b_1, \dots, b_n) \in \{0, 1\}^n$ .

**Teorema 1.3.1** (Cook-Levin). *CS è  $\mathcal{NP}$ -completo.*

*Dimostrazione.* La dimostrazione completa è omessa in quanto lunga e tecnicamente complessa. Si basa sul fatto che ogni algoritmo con un tempo di esecuzione polinomiale nella lunghezza dell'input può essere implementato da una famiglia di circuiti  $C_1, C_2, \dots$  dove  $C_n$  ha  $n$  nodi di input e un numero polinomiale di altri nodi. Per simulare l'algoritmo su un input  $I$  di lunghezza  $n$ , calcoliamo l'output del circuito  $C_n$  con input  $I$ .

Per dimostrare che un problema  $X \in \mathcal{NP}$  è polinomialmente riducibile a CS consideriamo un certificatore polinomiale  $B$  per  $X$ , il quale esiste in quanto  $X \in \mathcal{NP}$ . Costruiamo la famiglia di circuiti  $C'_1, C'_2, \dots$  tale che  $C'_n$  ha  $n + p(n)$  nodi senza archi entranti e un numero polinomiale in  $n$  di altri nodi, dove  $p(\cdot)$  è il polinomio che limita la lunghezza dei certificati di  $B$ . Per ogni  $n$ ,  $C'_n$  simula  $B$  su istanze  $I$  di lunghezza  $n$ . Data un'istanza  $I \in \mathcal{I}$  di lunghezza  $n$ , sia  $C'_n(I, \cdot)$  il circuito  $C'_n$  dove i valori dei primi  $n$  nodi senza archi entranti sono preassegnati ai bit di  $I$ , mentre i rimanenti  $p(n)$  nodi sono di input. Allora  $C'_n(I, \cdot)$  simula il certificatore polinomiale  $B(I, \cdot)$ . Ovvero,  $C'_n(I, \cdot)$  è soddisfacibile se e solo se esiste  $z \in \{0, 1\}^*$  con  $|z| \leq p(|I|)$  tale che  $B(I, z) = 1$ .  $\square$

Dimostriamo ora un caso particolare del teorema di Cook-Levin, ovvero che un particolare problema di decisione è polinomialmente riducibile a CS. Il problema è 2-IS, con istanze  $I$  che rappresentano grafi semplici, mentre la funzione di decisione  $q$  è tale che  $q(I) = 1$  se e solo se il grafo  $I$  contiene un insieme indipendente di tagli almeno 2.

**Teorema 1.3.2.** *2-IS  $\preceq_p$  CS.*

*Dimostrazione.* Sia  $G = (V, E)$  un grafo su  $n$  vertici. Gli archi del grafo  $G$  possono essere codificati con una stringa  $I_G \in \{0, 1\}^N$  dove  $N = \binom{n}{2}$ , ogni bit rappresenta una coppia di vertici e un arco fra una coppia di vertici è codificato ponendo a 1 il bit corrispondente. Indichiamo con  $B$  il certificatore polinomiale per 2-IS, il quale esiste in quanto  $2\text{-IS} \in \mathcal{NP}$ .

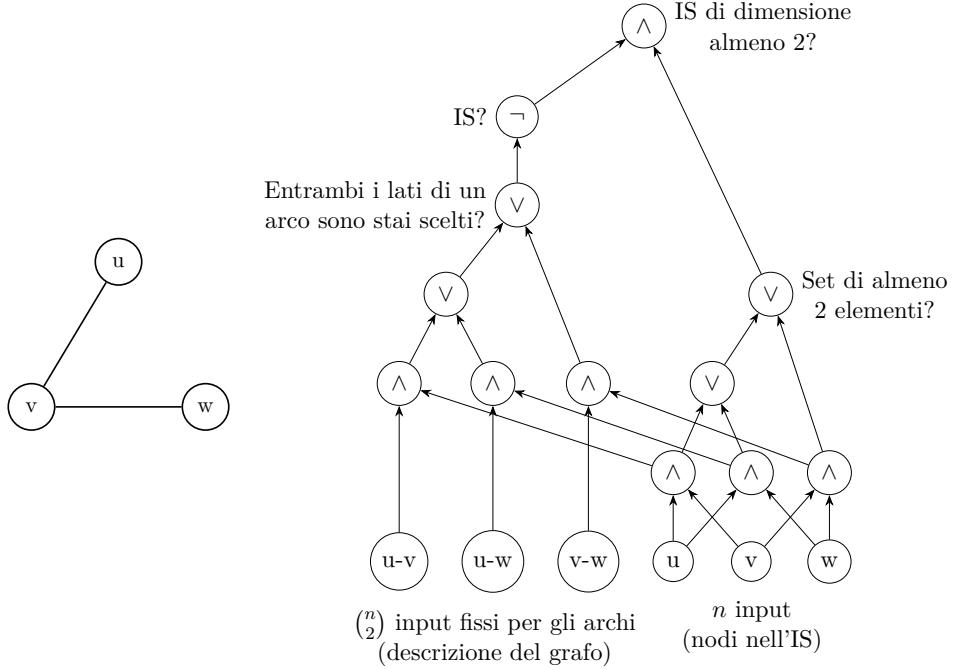


Figure 1.1: Riduzione da 2-IS a CS

Dimostriamo ora come costruire in tempo polinomiale nella descrizione di  $G$  un circuito  $C$  con  $\binom{n}{2} + n$  nodi senza archi entranti tale che  $C(I_G, \cdot)$  è soddisfacibile se e solo se esiste una stringa  $z$  di lunghezza  $n$  tale che  $B(I_G, z) = 1$ . Questo completa la riduzione, in quanto per calcolare la funzione di decisione di  $2 - IS$  su un'istanza  $I_G$  è sufficiente costruire  $C$ , invocare l'oracolo per CS e produrre in output la risposta dell'oracolo.

Notiamo che i certificati per 2-IS possono essere codificati con una stringa  $z \in \{0, 1\}^n$  che ha almeno due occorrenze di 1 nelle posizioni corrispondenti ai vertici che formano un insieme indipendente nel grafo.

Costruiamo quindi un circuito con  $\binom{n}{2} + n$  nodi di input tale che i primi  $\binom{n}{2}$  vengono utilizzati per codificare  $I_G$  e i successivi  $n$  nodi vengono utilizzati per codificare un certificato  $z$ . A questo punto, usiamo  $2\binom{n}{2} - 1$  per verificare che  $z$  contenga almeno due occorrenze di 1 e usiamo  $2\binom{n}{2}$  nodi interni per verificare che non ci sia un arco fra i nodi scelti dal certificato.  $\square$

In Figura 1.1 si può vedere un esempio:

- Al di sopra dei nodi di input per  $z$ , ogni nodo  $\wedge$  chiede “sono stati scelti questi due nodi?”; la restante parte destra dell’albero controlla solamente che siano stati scelti almeno due nodi
- Il risultato della “scelta” viene messo in  $\wedge$  con la codifica degli archi presenti
- I risultati vengono messi in  $\vee$ , quindi risulta 1 se e solo se sono stati scelti due nodi che hanno un arco presente tra loro; viene invertito per avere 1 nel caso di Independent Set
- La radice si occupa di controllare che l’input rappresenti un Independent set (sottoalbero di sinistra) di dimensione almeno 2 (sottoalbero di destra)

Una volta che abbiamo stabilito che un certo problema è  $\mathcal{NP}$ -completo, possiamo trovarne molti altri usando la seguente osservazione.

**Fatto 1.3.2.** *Se  $Y$  è  $\mathcal{NP}$ -completo e  $X \in \mathcal{NP}$  è tale che  $Y \preceq_p X$ , allora anche  $X$  è  $\mathcal{NP}$ -completo.*

*Dimostrazione.* Sia  $Z \in \mathcal{NP}$  qualunque. Allora  $Z \preceq_p Y$ . Ma dato che  $Y \preceq_p X$ , allora  $Z \preceq_p X$ , il che implica che  $X$  è  $\mathcal{NP}$ -completo.  $\square$

Possiamo subito applicare questa osservazione dimostrando quanto segue.

**Teorema 1.3.3.**  *$CS \preceq_p 3\text{-SAT}$ .*

*Dimostrazione.* Omessa.  $\square$

Dato che  $CS$  è  $\mathcal{NP}$ -completo, ne segue che anche  $3\text{-SAT}$  è  $\mathcal{NP}$ -completo. Ricordando inoltre che  $3\text{-SAT} \preceq_p$  Independent Set  $\preceq_p$  Vertex Cover  $\preceq_p$  Set Cover, ne deduciamo che tutti questi problemi sono  $\mathcal{NP}$ -completi. Al contrario di  $3\text{-SAT}$  che è  $\mathcal{NP}$ -completo,  $2\text{-SAT}$  (ovvero usando clausole da esattamente due letterali) è un problema risolvibile in tempo lineare.

Si noti che ogni formula booleana può essere equivalentemente rappresentata come un circuito (ovvero un’istanza di  $CS$ ) o come una formula CNF (ovvero un’istanza di  $SAT$ ) in modo che tali istanze abbiano lunghezza polinomiale nella lunghezza della formula. Inoltre, ogni istanza  $I$  di  $SAT$  può essere rappresentata come un’istanza di  $3\text{-SAT}$  di lunghezza polinomiale in

$|I|$  (queste dimostrazioni sono il processo per la dimostrazione omessa precedentemente).

Al contrario, non è sempre possibile rappresentare una formula booleana in DNF in modo che la DNF abbia lunghezza polinomiale nella lunghezza della formula. Se questo fosse possibile avremmo che  $\mathcal{P} \equiv \mathcal{NP}$ . La soddisfacibilità di una DNF è decidibile in tempo lineare controllando che esista almeno una congiunzione della formula che è soddisfacibile da un qualche assegnamento.

Si noti che la definizione di  $\mathcal{NP}$  è asimmetrica: dato un problema  $X = (\mathcal{I}, q) \in \mathcal{NP}$ , se  $q(I) = 1$  allora esiste un certificato polinomiale  $z$  tale che  $B(I, z) = 1$  dove  $B$  denota il certificatore polinomiale per  $X$ . Se invece  $q(I) = 0$  allora la definizione ci garantisce solo che  $B(I, z) = 0$  per un numero esponenziale di certificati  $z$ . Questa asimmetria si riscontra quanto consideriamo problemi  $\bar{X} = (\mathcal{I}, \bar{q})$  che sono complementari di problemi  $X = (\mathcal{I}, q)$ . Per *complementare* di  $X$  si intende che  $\bar{q}(I) = 0$ , ma non sappiamo se esista un certificato polinomiale che certifichi  $\bar{q}(I) = 1$ . Per esempio, se  $X$  è Independent Set, allora per certificare  $\bar{q}(I) = 1$  dovremmo trovare un certificato polinomiale che attesti che il grafo **non** contenga un insieme indipendente di taglia  $k$ .

Quindi non è chiaro se  $\mathcal{NP} \equiv \text{co-}\mathcal{NP}$ , dove co- $\mathcal{NP}$  indica la classe dei problemi di decisione complementari di problemi in  $\mathcal{NP}$  (se  $\mathcal{NP}$  è l'insieme dei problemi di cui posso verificare una soluzione/certificato in tempo polinomiale, per i problemi in co- $\mathcal{NP}$  non è possibile avere una soluzione verificabilmente corretta in tempo polinomiale, è invece possibile verificare in tempo polinomiale che una soluzione restituisce 0).

Lo scenario è differente per  $\mathcal{P}$ :  $X \in \mathcal{P}$  se e solo se  $\bar{X} \in \mathcal{P}$ . Infatti  $X = (\mathcal{I}, q) \in \mathcal{P}$  implica che esiste un algoritmo per calcolare  $q$  in tempo polinomiale. Ma allora si può anche calcolare  $\bar{q}$  in tempo polinomiale semplicemente calcolando  $q$  e complementando l'output. Quindi  $\mathcal{P} \equiv \text{co-}\mathcal{P}$ .

Dimostrare che  $\text{co-}\mathcal{NP} \not\equiv \mathcal{NP}$  sarebbe un progresso ancora maggiore che dimostrare  $\mathcal{P} \not\equiv \mathcal{NP}$ . Vale infatti la seguente cosa.

**Fatto 1.3.3.** *Se  $\text{co-}\mathcal{NP} \not\equiv \mathcal{NP}$  allora  $\mathcal{P} \not\equiv \mathcal{NP}$ .*

*Dimostrazione.* Dimostriamo la contrapositiva, ovvero che  $\mathcal{P} \equiv \mathcal{NP}$  implica  $\text{co-}\mathcal{NP} \equiv \mathcal{NP}$ . Intuitivamente, dato che  $\mathcal{P}$  è chiuso rispetto all'operazione di complemento, se  $\mathcal{P} \equiv \mathcal{NP}$  allora deve essere che  $\text{co-}\mathcal{NP} \equiv \mathcal{NP}$ . Formal-

mente

$$X \in \mathcal{NP} \iff X \in \mathcal{P} \iff \bar{X} \in \mathcal{P} \iff \bar{X} \in \mathcal{NP} \iff X \in \text{co-}\mathcal{NP}$$

Il che conclude la dimostrazione.  $\square$

Possiamo caratterizzare i problemi  $X = (\mathcal{I}, q) \in \text{co-}\mathcal{NP}$  tramite l'esistenza di un polinomio  $p(\cdot)$  e di un certificatore polinomiale  $B$ , calcolabile in tempo polinomiale, tale che  $q(I) = 0$  se e solo se esiste una stringa  $z \in \{0, 1\}^{p(|I|)}$  tale che  $B(I, z) = 0$ . Si noti che  $\mathcal{P} \subseteq \text{co-}\mathcal{NP}$ , infatti, potendo implementare  $q$  in tempo polinomiale, si può calcolare  $B(I, z)$  ignorando  $z$  e restituendo  $q(I)$ .

Una classe particolarmente interessante è quella dei problemi in  $\mathcal{NP} \cap \text{co-}\mathcal{NP}$ . Sia  $X = (\mathcal{I}, q)$  un tale problema

- Dato che  $X \in \mathcal{NP}$  esiste un polinomio  $p(\cdot)$  e un certificatore polinomiale  $B$  tale che  $q(I) = 1$  se e solo se esiste una stringa  $z \in \{0, 1\}^{p(|I|)}$  tale che  $B(I, z) = 1$
- Dato che  $X \in \text{co-}\mathcal{NP}$  esiste un polinomio  $p'(\cdot)$  e un certificatore polinomiale  $B'$  tale che  $q(I) = 0$  se e solo se esiste una stringa  $z' \in \{0, 1\}^{p'(|I|)}$  tale che  $B'(I, z') = 0$

Quindi i problemi in  $\mathcal{NP} \cap \text{co-}\mathcal{NP}$  sono tali che per ogni istanza  $I$  esiste un certificato polinomiale, sia quando  $q(I) = 1$  sia quando  $q(I) = 0$ .

Si noti che se  $X \in \mathcal{P}$  allora  $X \in \mathcal{NP}$  ed anche  $X \in \text{co-}\mathcal{NP}$ . Quindi  $\mathcal{P} \subseteq \mathcal{NP} \cap \text{co-}\mathcal{NP}$ . D'altra parte non si sa se  $\mathcal{P} \not\equiv \mathcal{NP} \cap \text{co-}\mathcal{NP}$ . Ovvero non si sa se esistono problemi le cui istanze hanno sempre certificati brevi ma tuttavia non sono risolvibili in tempo polinomiale.