Complementi di Algoritmi e Strutture Dati

Complessità computazionale

Docente: *Nicolò Cesa-Bianchi* versione 10 marzo 2025

Attenzione: Questo è un riassunto di alcune lezioni messo a disposizione dal docente. Pertanto, questo materiale non è in alcun modo da ritenersi sostitutivo del testo di riferimento (Jon Kleinberg, Éva Tardos. Algorithm Design. Pearson International Edition, 2006. Acronimo KT nel testo) per questa parte del programma.

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;
- $q : \mathcal{I} \to \{0,1\}$ è la **funzione di decisione**, che determina se $I \in \mathcal{I}$ è un'instanza **positiva** ($q(I) = 1$) o **negativa** ($q(I) = 0$) del problema di decisione.

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

Per esempio, $\mathcal{I}$ è l'insieme delle coppie $I = (G, k)$ dove $G = (V, E)$ è un grafo non orientato e k un intero positivo. La funzione q è definita in modo tale che $q(I) = 1$ con $I = (G, k)$ se e solo se G contiene una clique (sottografo completo) su almeno k vertici.

Indichiamo con $|I|$ la lunghezza della codifica in bit di un'istanza $I \in \mathcal{I}$. In genere, non specifichiamo il modo col quale un'istanza viene rappresentata come una stringa di bit. Resta però sottinteso che le istanze vengono codificate usando rappresentazioni "ragionevoli" (per esempio, una 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 $I \in \mathcal{I}$ (il tempo di calcolo è definito rispetto all'esecuzione di un interprete per un dato linguaggio di programmazione di riferimento). Definiamo ora $T_A(n) = \max\{T_A(I) : |I| \leq n\}$, 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 ad 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 **polinomialmente riducibile** ad un altro problema X, denotato con $Y \leq_{\mathrm{p}} X$, quando:

1. esiste un algoritmo che risolve Y in tempo polinomiale con accesso ad 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.

Se $Y \leq_{\mathrm{p}} X$ ed esiste un algoritmo che risolve X in tempo polinomiale, allora esiste un algoritmo che risolve Y in tempo polinomiale. Viceversa, se $Y \leq_{\mathrm{p}} X$ e non esiste un algoritmo che risolve Y

in tempo polinomiale, allora non esiste un algoritmo che risolve X in tempo polinomiale. Intuitivamente, $Y \leq_{\mathrm{p}} X$ significa che il problema Y non è più difficile del problema X. Se $X \leq_{\mathrm{p}} Y$ e $Y \leq_{\mathrm{p}} X$, ovvero X e Y sono polinomialmente riducibili l'uno con l'altro, allora scriviamo $X \equiv_{\mathrm{p}} Y$.

Sia $G = (V, E)$ un grafo semplice (cioè 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, per ogni coppia i, j di vertici distinti in S vale che $(i, j) \notin E$. Una **copertura** (vertex cover) di G è un sottoinsieme $S' \subseteq V$ di vertici tale che ogni arco di G ha almeno un estremo in S'. Ovvero, per ogni $(i, j) \in E$ esiste un $k \in S'$ tale che $k = i$ o $k = j$.

Fatto 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 $(i, j) \in E$ arbitrario, 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, se $(i, j) \in E$ allora S' non sarebbe una copertura. Quindi $(i, j) \notin E$ da cui otteniamo che S è un insieme indipendente. □

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 $q_{\mathrm{IS}}(I) = 1$ 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 in questo caso la funzione di decisione q_{VC} è tale che $q_{\mathrm{VC}}(I) = 1$ se e solo se G contiene una copertura di taglia al più k. Conseguenza immediata del Fatto 1 è che Independent Set e Vertex Cover sono polinomialmente riducibili l'uno con l'altro, come enunciato dal corollario seguente.

Corollario 2 *Independent Set $\equiv_{\mathrm{p}}$ Vertex Cover.*

Una proprietà utile della relazione $\leq_{\mathrm{p}}$ è la transitività.

Fatto 3 *Se $Z \leq_{\mathrm{p}} Y$ e $Y \leq_{\mathrm{p}} X$ allora $Z \leq_{\mathrm{p}} X$.*

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 l'oracolo viene invocato lo simuliamo eseguendo l'algoritmo per risolvere Y usando l'oracolo per X. □

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

Fatto 4 *Vertex Cover $\leq_{\mathrm{p}}$ 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 $\mathcal{S} \equiv \{S_i : i \in V\}$ dove S_i è l'insieme degli archi di G incidenti su i (si noti che questo può essere fatto in tempo lineare nella descrizione di G). Quindi, in particolare, ogni $u \in U$ è contenuto esattamente in due elementi di $\mathcal{S}$. Verifichiamo ora che U è l'unione di al più k insiemi $S_1, \ldots, S_{|V|}$ se e solo se G ha una copertura con vertici di taglia al più k.

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

Quindi possiamo definire un algoritmo che implementa la funzione di decisione q_{VC} 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 $q_{\mathrm{SC}}(I')$ che viene restituita in output. □

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, $\mathcal{S} \equiv \{S_1, \ldots, S_m\}$ è una collezione di sottoinsiemi di U e k è un intero. La funzione di decisione q_{SP} è tale che $q_{\mathrm{SP}}(I) = 1$ se e solo se esistono almeno k sottoinsiemi in $\mathcal{S}$ che sono a due a due disgiunti. Anche se Set Packing è apparentemente più generale di Independent Set, con una dimostrazione simile a quella del Fatto 4 possiamo dimostrare che i due problemi sono equivalenti.

Fatto 5 *Independent Set* $\equiv_{\mathrm{p}}$ *Set Packing.*

Dimostrazione. Data un'istanza $\mathcal{S} \equiv \{S_1, \ldots, S_m\}$ di Set Packing costruiamo in tempo polinomiale un grafo $G = (V, E)$ dove $V \equiv \{v_S : S \in \mathcal{S}\}$ e $(v_S, v_T) \in E$ se e solo se $S \cap T \not\equiv \emptyset$. 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 $\mathcal{S} \equiv \{S_i : i \in V\}$ 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. □

In molte discipline, come ad esempio in intelligenza artificiale, bisogna spesso risolvere problemi di ottimizzazione combinatorica vincolata. In questi problemi si cerca un assegnamento di valori ad 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 variabli booleane $x_1, \ldots, x_n$. Un assegnamento di valori di verità a $\mathcal{X}$ è una funzione $\pi : \mathcal{X} \to \{0, 1\}$. Un letterale ℓ_i è la variable x_i o la sua negazione $\bar{x}_i$. Una clausola $C = \ell_{i_1} \vee \cdots \vee \ell_{i_k}$ è una disgiunzione di letterali. Un assegnamento π 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 c'è almeno un letterale della forma $\ell_j = \bar{x}_j$ 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 $\pi : \mathcal{X} \to \{0, 1\}$ che soddisfa tutte le clausole in $\mathcal{C}$. Una versione ridotta di SAT è il problema **3-SAT**, le cui istanze sono insiemi di clausole ciascuna contenente esattamente tre letterali.

Un'istanza di 3-SAT è $\mathcal{X} \equiv \{x_1, x_2, x_3, x_4\}$ con $\mathcal{C} \equiv \big\{(\bar{x}_1 \vee x_2 \vee \bar{x}_3), (\bar{x}_2 \vee x_3 \vee x_4), (x_1 \vee x_2 \vee \bar{x}_4\big\}$. Un assegnamento che soddisfa tutte le clausole è $\pi(x_i) = 0$ per $i = 1, \ldots, 4$.

Teorema 6 *3-SAT* $\leq_{\mathrm{p}}$ *Independent Set*

DIMOSTRAZIONE. È facile verificare che la seguente è 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 fra 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 $q_{3\mathrm{SAT}}(I) = q_{\mathrm{IS}}(I')$. 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 in $|\mathcal{C}|$.

Per come abbiamo costruito 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 il nodo scelto in un'altra clique. Ora, per costruzione, questo avviene se e solo se è possibile scegliere esattamente un letterale in ciascuna clausola in modo tale che fra i letterali scelti non compaiano simultaneamente x_i e $\bar{x}_i$ per nessuna delle variabili $x_i \in \mathcal{X}$. Ma allora possiamo 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 $q_{\mathrm{IS}}(I')$; infine, restituisce $q_{\mathrm{IS}}(I')$ in output. □

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 $B : \mathcal{I} \times \{0,1\}^* \to \{0,1\}$ 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 a un certificato del fatto che $q(I) = 1$. Per esempio, nel problema Indipendent Set la stringa z denota il sottoinsieme di vertici che costituisce un insieme indipendente di cardinalità almeno k. Nel problema Satisfiability, la stringa z denota un assegnamento che soddisfa tutte le clausole. Si noti che avere accesso ad 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 $q(I)$—dovremmo eseguire $B(I, z)$ 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 $z \in \{0,1\}^*$ (e, in particolare, per quegli z di lunghezza limitata 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 di decisione la cui soluzione (o certificato) è **verificabile** in tempo polinomiale in contrasto coi problemi in $\mathcal{P}$ che sono invece **risolvibili** in tempo polinomiale. Il più importante problema aperto in informatica teorica è stabilire se vale $\mathcal{P} \not\equiv \mathcal{NP}$.

Un problema di decisione X è $\mathcal{NP}$-completo se $X \in \mathcal{NP}$ e per qualunque $Y \in \mathcal{NP}$ vale $Y \leq_{\mathrm{p}} X$.[1] Intuitivamente, i problemi di decisione $\mathcal{NP}$-completi sono i più difficili in $\mathcal{NP}$. Infatti vale il fatto seguente.

Fatto 7 *Sia X un qualunque problema $\mathcal{NP}$-completo. Allora $X \in \mathcal{P}$ se e solo se $\mathcal{P} \equiv \mathcal{NP}$.*

Dimostrazione. Se $\mathcal{P} \equiv \mathcal{NP}$ allora evidentemente $X \in \mathcal{P}$. D'altra parte, se X è $\mathcal{NP}$-completo allora dato un qualunque $Y \in \mathcal{NP}$ vale $Y \leq_{\mathrm{p}} X$. Dato che per ipotesi X è risolubile in tempo polinomiale, allora anche Y è risolubile in tempo polinomiale e quindi $Y \in \mathcal{P}$ da cui otteniamo $\mathcal{NP} \subseteq \mathcal{P}$. Dato che vale anche $\mathcal{P} \subseteq \mathcal{NP}$ concludiamo che $\mathcal{P} \equiv \mathcal{NP}$. □

Non è chiaro che esistano problemi $\mathcal{NP}$-completi. Infatti, potrebbero esserci due problemi $X', X'' \in \mathcal{NP}$ tali che non esiste nessun altro $X \in \mathcal{NP}$ tale che $X' \leq_{\mathrm{p}} X$ e $X'' \leq_{\mathrm{p}} X$. Oppure, potrebbe esserci una sequenza infinita di problemi in $\mathcal{NP}$ del tipo $X_1 \leq_{\mathrm{p}} X_2 \leq_{\mathrm{p}} \cdots$ in modo che non esista un problema più difficile di tutti gli 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 $\vee$ (OR), $\wedge$ (AND), $\neg$ (NOT) in modo tale che nodi AND e OR abbiano esattamente due archi entranti e nodi NOT abbiano esattamente un arco entrante (si veda la Figura 1).

Un circuito calcola una funzione booleana $f : \{0,1\}^n \to \{0,1\}$ dove n è il numero dei nodi di input. Dato un assegnamento $(b_1, \ldots, b_n) \in \{0,1\}^n$ di valori di verità ai nodi di input, calcoliamo $f(b_1, \ldots, b_n)$ 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 di nodi.

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 $f(b_1, \ldots, b_n) = 1$ per un qualche $(b_1, \ldots, b_n) \in \{0,1\}^n$.

Teorema 8 (Cook-Levin) *CS è $\mathcal{NP}$-completo.*

[1] Una definizione alternativa di $\mathcal{NP}$-completezza può essere data usando la nozione di riducibilità secondo Turing. Non è tuttavia noto se le due nozioni di $\mathcal{NP}$-completezza siano equivalenti.

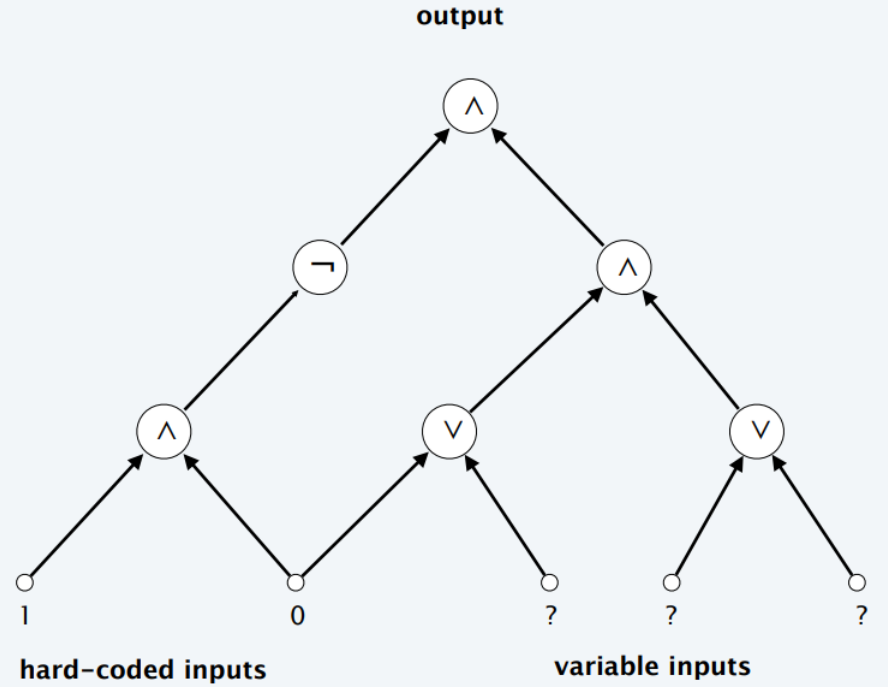

Figura 1: Un circuito.

La dimostrazione di questo teorema è omessa dato che è lunga e tecnicamente complessa. Essa 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 $C_1, C_2, \ldots$ 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 (questo esiste perché $X \in \mathcal{NP}$). Costruiamo la famiglia di circuiti $C'_1, C'_2, \ldots$ 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 $B(I, z) = 1$.

Come esercizio, dimostriamo ora un caso particolare del teorema di Cook-Levin, ovvero che un particolare problema di decisione è polinomialmente riducibile a CS. Il problema di decisione 2-IS ha 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 taglia almeno 2.

Teorema 9 *2-IS* $\leq_{\mathrm{p}}$ *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

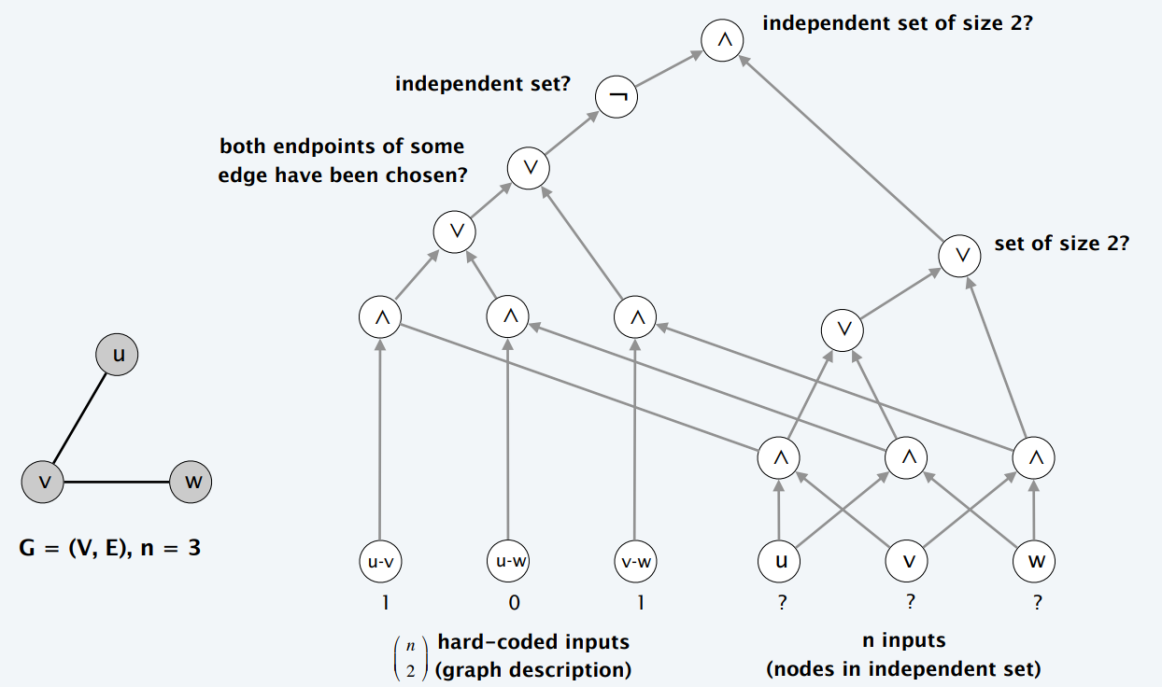

Figura 2: Riduzione da 2-IS a CS.

un arco fra una coppia di vertici è codificato ponendo a 1 il bit corrispondente. Indichiamo con B il certificatore polinomiale per 2-IS (sappiamo che esiste perché 2-IS è in $\mathcal{NP}$). 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.

Per prima cosa 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 $\binom{n}{2}$ nodi vengono utilizzati per codificare I_G e i successivi n nodi vengono utilizzati per codificare un certificato z. A questo punto, usiamo $2\binom{n}{2} - 1$ nodi interni 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 (si veda la Figura 2). □

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

Fatto 10 *Se Y è $\mathcal{NP}$-completo e $X \in \mathcal{NP}$ è tale che $Y \leq_{\mathrm{p}} X$, allora anche X è $\mathcal{NP}$-completo.*

Dimostrazione. Sia $Z \in \mathcal{NP}$ qualunque. Allora $Z \leq_{\mathrm{p}} Y$. Ma dato che $Y \leq_{\mathrm{p}} X$ allora $Z \leq_{\mathrm{p}} X$, il che implica che X è $\mathcal{NP}$-completo. □

Possiamo subito applicare questa osservazione dimostrando quanto segue (dimostrazione omessa).

Teorema 11 *CS* $\leq_{\mathrm{p}}$ *3-SAT.*

Dato che CS è $\mathcal{NP}$-completo, ne segue che anche 3-SAT è $\mathcal{NP}$-completo. Ricordando inoltre che 3-SAT $\leq_{\mathrm{p}}$ Independent Set $\leq_{\mathrm{p}}$ Vertex Cover $\leq_{\mathrm{p}}$ Set Cover, ne deduciamo che tutti questi problemi sono $\mathcal{NP}$-completi. Al contrario di 3-SAT che è $\mathcal{NP}$-completo, 2-SAT (ovvero il problema di soddisfacibilità dove ogni clausola contiene esattamente due letterali) è un problema risolvibile addirittura in tempo lineare.

Si noti che ogni formula booleana (formula ben formata costruita applicando gli operatori AND, OR e NOT a letterali) 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'instanza di 3-SAT di lunghezza polinomiale in $|I|$. Al contrario, non è sempre possibile rappresentare una formula booleana in una DNF (ovvero una disgiunzione di congiunzioni di letterali) in modo che la DNF abbia lunghezza polinomiale nella lunghezza della formula. Se questo fosse possibile, avremmo che $\mathcal{P} \equiv \mathcal{NP}$. Infatti, la soddisfacibilità di una DNF è decidibile in tempo lineare, semplicemente 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 quando consideriamo problemi $\overline{X} = (\mathcal{I}, \overline{q})$ che sono complementari di problemi $X = (\mathcal{I}, q)$. Per complementare di X intendiamo semplicemente che $\overline{q}(I) = 1 - q(I)$ per ogni $I \in \mathcal{I}$. Ora se $X \in \mathcal{NP}$ cosa possiamo dire di $\overline{X}$? L'esistenza di un certificatore polinomiale B per X ci garantisce che esiste un certificato polinomiale che certifica $\overline{q}(I) = 0$, ma non sappiamo se esista un certificato polinomiale che certifichi $\overline{q}(I) = 1$. Per esempio, se X è Independent Set, allora per certificare $\overline{q}(I) = 1$ 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 che sono complementari di problemi in $\mathcal{NP}$.

Lo scenario è differente per $\mathcal{P}$: $X \in \mathcal{P}$ se e solo se $\overline{X} \in \mathcal{P}$. Infatti, $X = (\mathcal{I}, q) \in \mathcal{P}$ implica che esiste un algoritmo per calcolare q in tempo polinomiale. Ma allora posso anche calcolare $\overline{q}$ in tempo polinomiale semplicemente calcolando q e complementando l'output (ovvero $\overline{q}(I) = 1 - q(I)$). Quindi $\mathcal{P} \equiv \text{co}{-}\mathcal{P}$.

Dimostrare che 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 12 *Se* 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 co$-\mathcal{NP} \equiv \mathcal{NP}$. Intuitivamente, dato che $\mathcal{P}$ è chiuso rispetto all'operazione di complemento (cioè co$-\mathcal{P} \equiv \mathcal{P}$), se $\mathcal{P} \equiv \mathcal{NP}$ allora dev'essere co$-\mathcal{NP} \equiv \mathcal{NP}$. Formalmente,

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

il che conclude la dimostrazione. □

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, se posso implementare q in tempo polinomiale, posso 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

$$q(I) = 1 \text{ se e solo se esiste una stringa } z \in \{0,1\}^{p(|I|)} \text{ tale che } B(I, z) = 1.$$

Dato $X \in \text{co}{-}\mathcal{NP}$ esiste un polinomio $p'(\cdot)$ e un certificatore polinomiale B' tale che

$$q(I) = 0 \text{ se e solo se esiste una stringa } z' \in \{0,1\}^{p'(|I|)} \text{ tale che } B'(I, z') = 0.$$

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 risolubili in tempo polinomiale.