3.4. Le classi P, NP, co-NP e NPC

Dopo aver visto come risolvere alcuni problemi in tempo polinomiale e aver proposto per altri solo algoritmi esponenziali, dobbiamo fare un po' di ordine nella teoria. Questo ci aiuterà a impostare organicamente i problemi anche se lascerà aperte alcune questioni fondamentali per cui mancano risultati definitivi.

Anzitutto ogni ente di cui trattiamo dovrà essere rappresentato in modo efficiente nel senso tecnico del termine (capitolo 2); cioè i dati, gli algoritmi, e ogni struttura che questi possano costruire durante il calcolo avranno dimensione logaritmica nel numero di elementi della classe cui appartengono. In particolare un numero N sarà rappresentato con $\Theta(\log N)$ caratteri, o precisamente con $\lceil \log_2 N \rceil$ cifre binarie. Un insieme di n elementi sarà rappresentato con $\Theta(n)$ caratteri nell'ipotesi che ogni elemento abbia dimensione costante d. Notiamo però che in questo modo si possono rappresentare al più a^d elementi distinti, ove a è la cardinalità dell'alfabeto. Poiché il valore di a^d per quanto grande è costante, all'aumentare di n l'insieme conterrà necessariamente elementi ripetuti: se si vuole evitare questa situazione ogni elemento dovrà essere rappresentato con $\Theta(\log n)$ caratteri portando a $\Theta(n \log n)$ la dimensione complessiva dell'insieme. L'ipotesi di rappresentazione efficiente degli enti in gioco è implicita e non sarà più ripetuta.

Abbiamo affermato che l'inerente difficoltà di un problema è in genere già presente nella sua forma decisionale che chiede di rispondere a un quesito binario sull'istanza di volta in volta considerata. Concretamente anziché *determinare* una soluzione relativa all'istanza si chiede di stabilire se esiste una soluzione che goda di una data proprietà: in caso affermativo diremo che l'istanza è *accettabile*. La teoria si rivolge inizialmente ai problemi decisionali dividendoli in due classi, dette **P** e **NP**, in relazione alla complessità degli algoritmi di soluzione. La prima definizione è semplice:

P è la classe dei problemi decisionali risolubili con un algoritmo polinomiale.

Per esempio $\mathcal{P}_{ric}(k, l) \in \mathbf{P}$. Parimenti è in **P** il problema $\mathcal{P}_{coprimi}(N, M)$ che chiede di stabilire se due interi N, M sono primi tra loro, perché si può calcolare $mcd(N, M)$ con l'algoritmo polinomiale `Euclid`(N, M). Non sappiamo invece se il problema $\mathcal{P}_{primo}(N)$ (stabilire se un intero N è primo) appartiene a **P** perché abbiamo indicato solo un algoritmo esponenziale per risolverlo.

La definizione della seconda classe è molto più sottile. Un problema $\mathcal{P}$ per cui non si conosce un algoritmo polinomiale è praticamente irrisolubile al crescere della lunghezza della sequenza x che rappresenta i suoi dati d'ingresso. È però possibile che un'informazione addizionale y, detta *certificato* di x, consenta di risolvere più efficientemente $\mathcal{P}$ attraverso un *algoritmo di verifica* $\mathcal{A}(x, y)$. Perché il discorso non appaia troppo astratto consideriamo come esempio l'istanza del problema $\mathcal{P}_{zaino}(I, b, c)$ presentata nel paragrafo precedente: la sequenza d'ingresso x rappresenta l'insieme $I = \{5, 8, 12, 6, 6, 7, 25, 4, 2, 9\}$ e i limiti $b = 20$, $c = 24$. Abbiamo visto che il sottoinsieme $T = \{8, 6, 7, 2\}$ è una soluzione che soddisfa la condizione dello zaino, quindi l'esistenza di T dimostra che l'istanza x è accettabile. Orbene in questo caso un certificato y è la stessa sequenza che rappresenta T, poiché esiste un algoritmo di verifica che scandisce $y = T$, calcola la somma dei suoi elementi, confronta tale somma con i valori di b e c in $x = I$, e stabilisce così che per x esiste una soluzione.

Affinché questo approccio abbia interesse dobbiamo chiarire alcuni punti importanti. Anzitutto abbiamo affermato che l'esistenza di un certificato y e di un algoritmo $\mathcal{A}$ di verifica permette di attestare che l'istanza x è accettabile: in questo caso porremo (convenzionalmente) $\mathcal{A}(x, y) = 1$; ma il risultato opposto $\mathcal{A}(x, y) = 0$ *non* indica che una soluzione per x non esiste, ma solo che y non ne è un certificato. Definiremo perciò il certificato solo per le istanze accettabili: in sostanza un certificato è un attestato di esistenza della soluzione, ma non è facile costruire attestati di non esistenza. Inoltre, per un dato problema $\mathcal{P}$, è necessario che:

1. ogni istanza accettabile x di lunghezza n ammetta un certificato y di lunghezza polinomiale in n;
2. esista un algoritmo di verifica $\mathcal{A}$ *polinomiale* in n e applicabile a ogni coppia $\langle x, y \rangle$.

Se queste due condizioni sono valide si dice che $\mathcal{A}$ *verifica* $\mathcal{P}$ *in tempo polinomiale*. Possiamo allora porre la definizione:

NP è la classe dei problemi decisionali verificabili in tempo polinomiale.[8]

[8] L'acronimo **NP** sta per «nondeterministico polinomiale», ed è nato nell'ambito del calcolo non deterministico ove questa teoria ha avuto origine.

Per quanto abbiamo già visto $\mathcal{P}_{zaino}(I, b, c) \in \mathbf{NP}$. Parimenti è in **NP** il problema $\mathcal{P}_{ham}(G)$; infatti, per ogni grafo G (instanza x) che ammette un ciclo hamiltoniano, esiste un certificato y che specifica la permutazione di vertici associata al ciclo.

Poiché in pratica un certificato contiene un'informazione molto prossima alla soluzione del problema, possiamo chiederci quale sia l'interesse di aver introdotto questo concetto. In fondo stiamo solo cercando di eludere la difficoltà: se la soluzione è (esponenzialmente) difficile da raggiungere ammettiamo di averla già! Il dubbio è assolutamente legittimo: come ora vedremo la teoria della verifica è utilissima per far luce sulle gerarchie di complessità dei problemi ma non aggiunge nulla alla possibilità di risolverli efficientemente.

Notiamo anzitutto che ogni problema $\mathcal{P} \in \mathbf{P}$ è banalmente verificabile in tempo polinomiale. Infatti si può costruire un certificato vuoto y per ogni sequenza d'ingresso x e definire un algoritmo $\mathcal{A}(x, y)$ che ignora y e verifica l'accettabilità di x risolvendo direttamente $\mathcal{P}$ in tempo polinomiale. Ne segue:

$$\mathbf{P} \subseteq \mathbf{NP}$$

Non si sa se $\mathbf{P} \neq \mathbf{NP}$ anche se vi sono forti motivi per ritenere che ciò sia vero. Dal punto di vista teorico questo è il principale problema aperto nello studio della complessità. In pratica si ammette che sia $\mathbf{P} \neq \mathbf{NP}$ fino a una (improbabile) prova contraria. Vi sono moltissimi problemi in **NP**, tra cui per esempio $\mathcal{P}_{zaino}$ e $\mathcal{P}_{ham}$, per cui non è noto alcun algoritmo polinomiale di soluzione e si ritiene che questa situazione non sia destinata a cambiare. In gergo i problemi in **P** sono detti *trattabili* e i problemi in $\mathbf{NP} - \mathbf{P}$ sono detti *intrattabili*, anche se questi ultimi, come vedremo, presentano presumibilmente diversi gradi di «difficoltà».

Un altro punto importante è la profonda differenza tra certificare l'esistenza (compito possibilmente semplice) o la non esistenza (compito spesso difficile) di una soluzione, come si può ben comprendere su un esempio. Il problema $\mathcal{P}_{primo}(N)$ può essere risolto con un algoritmo enumerativo esponenziale. Si può in effetti dimostrare che $\mathcal{P}_{primo}(N) \in \mathbf{NP}$, ma la cosa non è banale perché è difficile costruire un certificato di primalità per N. Il problema $\mathcal{P}_{composto}(N)$, che chiede di stabilire se N *non* è primo, può ovviamente essere risolto con lo stesso algoritmo complementandone la risposta. Tuttavia per questo problema ogni istanza N accettabile ammette un semplice certificato costituito da un divisore R di N, quindi si può banalmente concludere che $\mathcal{P}_{composto}(N) \in \mathbf{NP}$.

La questione generale è nei seguenti termini. Per ogni problema decisionale $\mathcal{P}$ si definisce *problema complementare* $\overline{\mathcal{P}}$ quello per cui sono accettabili le istanze non accettabili per $\mathcal{P}$, e viceversa. Per esempio $\mathcal{P}_{primo}(N)$ e

$\mathcal{P}_{composto}(N)$ sono l'uno complementare dell'altro. Si pone quindi:

co-NP è la classe dei problemi decisionali $\mathcal{P}$ per cui $\overline{\mathcal{P}} \in$ **NP**.

Poiché $\mathcal{P}_{composto}(N) \in$ **NP** si ha: $\mathcal{P}_{primo}(N) \in$ **co-NP**. Quindi per quanto abbiamo visto sopra: $\mathcal{P}_{primo}(N) \in$ **NP** $\cap$ **co-NP**.

Con ragionamento analogo ai precedenti si vede anche immediatamente che **P** $\subseteq$ **co-NP**, quindi **P** $\subseteq$ **NP** $\cap$ **co-NP**; ma non si sa in che relazione **NP** e **co-NP** stiano tra loro, anche se si ammette che sia **NP** $\neq$ **co-NP**.[9]

Veniamo ora a un concetto che ci permetterà di aggiungere un tassello importante alla teoria. Si dice che un problema $\mathcal{P}_1$ *si riduce in tempo polinomiale* a un problema $\mathcal{P}_2$, e si scrive $\mathcal{P}_1 \leqslant_p \mathcal{P}_2$, se esiste un algoritmo polinomiale $\mathcal{R}$ che trasforma ogni istanza x_1 di $\mathcal{P}_1$ in un'istanza x_2 di $\mathcal{P}_2$ in modo che x_1 è accettabile in $\mathcal{P}_1$ se e solo se x_2 è accettabile in $\mathcal{P}_2$. L'esistenza di un algoritmo di riduzione $\mathcal{R}$ permette di risolvere $\mathcal{P}_1$ in tempo polinomiale se si sa risolvere $\mathcal{P}_2$ in tempo polinomiale: infatti per ogni istanza x_1 si può calcolare la corrispondente istanza $x_2 = R(x_1)$ e poi risolvere $\mathcal{P}_2$ su questa. Possiamo così definire una fondamentale sottoclasse di **NP**:

NPC è la classe dei problemi decisionali $\mathcal{P}$ tali che

$\mathcal{P} \in$ **NP**;
per ogni $\mathcal{P}' \in$ **NP** si ha $\mathcal{P}' \leqslant_p \mathcal{P}$.

I problemi in **NPC** sono detti *NP-completi* a indicare che possiedono tutte le (peggiori) caratteristiche della classe **NP**. Essi sono infatti i «più difficili» problemi di quella classe poiché se si scoprisse un algoritmo polinomiale per risolverne uno, tutti i problemi in **NP** potrebbero essere risolti in tempo polinomiale attraverso l'algoritmo di riduzione. D'altra parte se si scoprisse che per risolvere anche uno solo dei problemi in **NP** è indispensabile un tempo esponenziale, tutti i problemi in **NPC** richiederebbero tale tempo e si sarebbe dimostrata la congettura **P** $\neq$ **NP**.

Un caposaldo nella teoria della complessità fu posto con l'individuazione del primo problema *NP*-completo: successivamente attraverso il processo di riduzione sono stati riconosciuti come *NP*-completi moltissimi altri problemi, tra cui per esempio $\mathcal{P}_{zaino}(I, b, c)$ e $\mathcal{P}_{ham}(G)$. Altri problemi sono in una situazione intermedia poiché o non ne è stata ancora dimostrata l'appartenenza a **NPC**, o la loro appartenenza a tale classe implicherebbe che **P** = **NP** ed è quindi ritenuta estremamente improbabile. Tra questi ultimi vi

[9] Si può dimostrare che la congettura **NP** $\neq$ **co-NP** implica la **P** $\neq$ **NP** ma non viceversa. Potrebbero perciò valere contemporaneamente **P** $\neq$ **NP** e **NP**=**co-NP**.

è il problema $\mathcal{P}_{primo}(N)$ particolarmente rilevante in crittografia, che potrebbe quindi essere «più facile» dei problemi *NP*-completi anche se non si conosce un algoritmo polinomiale per risolverlo.[10] Complessivamente le classi di complessità di cui abbiamo trattato si trovano nella situazione riportata nella figura 3.1 sotto le ipotesi $\mathbf{P} \neq \mathbf{NP}$ e $\mathbf{NP} \neq \mathbf{co\text{-}NP}$. Si immagina anche che sia $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$: l'ipotesi contraria implicherebbe, per esempio, che $\mathcal{P}_{primo}(N)$ sia risolubile in tempo polinomiale.[11]

Questo studio deve essere approfondito nei testi citati in bibliografia: poiché però abbiamo definito le classi di complessità solo per problemi decisionali, vediamo come si possono collocare in questa teoria tutti i problemi. A tale scopo dobbiamo estendere la definizione di riduzione polinomiale $\mathcal{P}_1 \leqslant_p \mathcal{P}_2$ imponendo che, oltre l'algoritmo per trasformare le istanze di $\mathcal{P}_1$ in istanze di $\mathcal{P}_2$, esista un algoritmo polinomiale per trasformare le soluzioni di $\mathcal{P}_2$ in corrispondenti soluzioni di $\mathcal{P}_1$. Possiamo allora porre un'ultima definizione:

NPH è la classe dei problemi $\mathcal{P}$ tali che per ogni $\mathcal{P}' \in \mathbf{NP}$ si ha $\mathcal{P}' \leqslant_p \mathcal{P}$.

I problemi in **NPH** sono detti *NP-hard*, o *NP-ardui* in italiano: per essi non è richiesta l'appartenenza a **NP**, quindi non sono necessariamente problemi decisionali. Confrontando le definizioni di **NPC** e **NPH** si vede subito che per stabilire se un problema $\mathcal{P}$ è *NP*-arduo è sufficiente controllare che un qualunque problema *NP*-completo si riduca a $\mathcal{P}$. Per esempio si considerino il problema *NP*-completo $\mathcal{P}_{zaino}(I, b, c)$ e la sua versione $\mathcal{P}_{zaino.max}(I, c)$ che chiede di determinare il sottoinsieme di I di somma s massima con l'unica condizione che sia $s \leqslant c$. Si tratta per esempio di riempire al meglio un disco di capacità c con file tratti da un insieme I. Dalla soluzione di $\mathcal{P}_{zaino.max}(I, c)$ si ricava immediatamente (e polinomialmente) una soluzione per $\mathcal{P}_{zaino}(I, b, c)$ poiché l'istanza I, b, c di questo è accettabile se e solo se $s \geqslant b$. Dunque il problema $\mathcal{P}_{zaino.max}(I, c)$ è *NP*-arduo. Similmente si dimostra che il problema del commesso viaggiatore $\mathcal{P}_{cv}(G)$ è *NP*-arduo.

I problemi che si incontrano nella pratica richiedono di *determinare* una soluzione con particolari proprietà e non semplicemente di stabilirne l'esistenza. Un problema *NP*-arduo $\mathcal{P}$ è in genere «più difficile» del corrispondente problema decisionale $\mathcal{P}_D$ nel senso che un eventuale algoritmo polinomiale per risolvere $\mathcal{P}$ permetterebbe di risolvere anche $\mathcal{P}_D$ in tempo poli-

[10] Sappiamo che $\mathcal{P}_{primo}(N) \in \mathbf{NP} \cap \mathbf{co\text{-}NP}$: per i problemi in questa situazione vi è ancora speranza di trovare un algoritmo polinomiale senza creare particolari turbamenti alla gerarchia delle classi di complessità.

[11] Nel 2002, dopo la pubblicazione della prima edizione di questo testo, è stato dimostrato che $\mathcal{P}_{primo}(N) \in \mathbf{P}$. Come sottolineato nella nota precedente ciò non ha alcuna conseguenza sulle classi di complessità; in particolare non implica che sia $\mathbf{P} = \mathbf{NP} \cap \mathbf{co\text{-}NP}$.

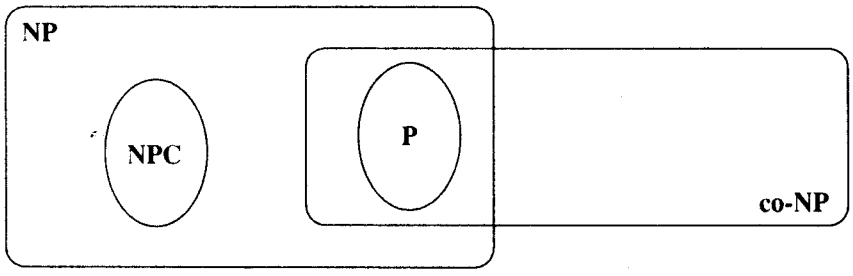

Figura 3.1
Relazione tra classi di complessità nell'ipotesi **P** ≠ **NP** e **NP** ≠ **co-NP**. Esempi di appartenenza: $\mathcal{P}_{ric} \in$ **P**; $\mathcal{P}_{coprimi} \in$ **P**; $\mathcal{P}_{primo} \in$ **NP** ∩ **co-NP**; $\mathcal{P}_{zaino} \in$ **NPC**; $\mathcal{P}_{ham} \in$ **NPC**.

nomiale attraverso la trasformazione delle istanze da $\mathcal{P}_D$ a $\mathcal{P}$ e dei risultati da $\mathcal{P}$ a $\mathcal{P}_D$. Quindi *se* $\mathcal{P}_D$ è intrattabile anche $\mathcal{P}$ è intrattabile. È comunque possibile che $\mathcal{P}$ sia «non più difficile» di $\mathcal{P}_D$, nel senso che ciascun problema si possa trasformare nell'altro in tempo polinomiale. In conclusione tutti i problemi in **NPH** sono «non più facili» di quelli in **NP**, ma possono essere «molto» più difficili. Consideriamo un caso limite con cui concluderemo questa carrellata sulle classi di complessità.

Prendiamo nuovamente in esame il problema $\mathcal{P}_{diof}(E)$ delle equazioni diofantee che chiede di decidere se l'equazione algebrica E, a coefficienti interi, di grado arbitrario e in un numero arbitrario di variabili, ammette una soluzione in cui tutte le variabili hanno valori interi. Nel capitolo 2 abbiamo visto che $\mathcal{P}_{diof}(E)$ è algoritmicamente indecidibile, ma questo non implica che siano indecidibili alcune sue forme particolari. Immaginiamo che le variabili in E siano due, indicate con x e y, e che l'equazione abbia la forma lineare $ax + by + c = 0$, o la forma quadratica $ax^2 + by + c = 0$ (cioè rappresenti rispettivamente una retta o una parabola nel piano). Indicheremo con $\mathcal{P}_{diof.lin}(E)$ e $\mathcal{P}_{diof.quad}(E)$ le forme assunte rispettivamente dal problema. I dati sono costituiti nei due casi dai coefficienti a, b, c, e ponendo che questi siano dello stesso ordine di grandezza avremo una dimensione n dell'ingresso logaritmica nella descrizione di ciascuno di essi.

Lasciamo al lettore il compito di verificare che l'equazione $ax + by + c = 0$ ammette una soluzione intera (cioè la retta corrispondente passa per un punto del piano a coordinate intere) se e solo se c è divisibile per $mcd(a, b)$. Per esempio l'equazione $2x - 6y + 10 = 0$ ammette la soluzione intera $x = -2$, $y = 1$; ma la $2x - 6y + 9 = 0$ non ammette soluzioni intere. Poiché il calcolo di $mcd(a, b)$ e di $c/mcd(a, b)$ si può eseguire in tempo polinomiale in n, abbiamo $\mathcal{P}_{diof.lin}(E) \in$ **P**.

Il problema $\mathcal{P}_{diof.quad}(E)$ è più complesso. Il lettore potrà facilmente dimostrare che, in questo caso, se E ammette una soluzione intera con $x = \bar{x}$, allora ne ammette anche una con $x = \bar{x} + b$. Dunque si può risolvere il problema sostituendo in E tutti i valori interi di x tra 0 e $b - 1$ e controllando se,

per uno di essi, y assume valore intero: E ammatterà una soluzione intera se e solo se ciò si verifica. Questo calcolo richiede tempo proporzionale al valore di b, quindi esponenziale in n: in effetti è possibile dimostrare che $\mathcal{P}_{diof.quad}(E)$ è un problema NP-completo e quindi non si può far meglio. Tuttavia è chiaro che un eventuale algoritmo $\mathcal{A}$ per risolvere $\mathcal{P}_{diof}(E)$ nella sua forma generale risolverebbe immediatamente anche $\mathcal{P}_{diof.quad}(E)$: dunque $\mathcal{P}_{diof}(E)$, benché indecidibile, è NP-arduo (e MOLTO più difficile dei problemi in **NP**).