

Liberamente estratto dal testo:

F.Luccio. La struttura degli algoritmi. Boringhieri, Torino 1982

1. Problemi decidibili e indecidibili

Il primo passo nello studio della complessità di algoritmi è anche il più drastico: ci chiediamo se esistano problemi *indecidibili*, tali cioè che non ammettano alcun algoritmo di soluzione.¹ Alla fine del 1800 un risultato di teoria degli infiniti dette al quesito una risposta genericamente affermativa, ma solo nel 1936 Alan Turing fornì la prima enunciazione di un problema indecidibile: questo problema può essere espresso informalmente nel modo seguente:

Problema della terminazione (o halting problem). Dato un algoritmo A e un dato D, entrambi arbitrari, stabilire se la computazione A(D) termina in un numero finito di passi.

Turing dimostrò che questo problema è indecidibile; ovvero non può esistere alcun algoritmo di decisione TER che, accettata una coppia A,D come dato d'ingresso, stabilisca *in tempo finito*, per qualunque valore della coppia, se la computazione A(D) termina anch'essa in tempo finito. Si noti che la computazione TER(A,D) non può semplicemente consistere nel simulare l'esecuzione di A su D e controllarne la terminazione, poiché se tale esecuzione non termina, TER(A,D) non risponde in tempo finito. L'argomentazione di Turing, riferita ad algoritmi di decisione, è schematicamente la seguente:

1. Ogni algoritmo A deve essere formulato attraverso un *programma* di lunghezza finita, in un sistema di calcolo prescelto. È quindi possibile associare ad A un numero intero che si ottiene per esempio rappresentando in forma binaria ogni simbolo presente nel programma, e interpretando poi come numero binario l'intera sequenza di simboli che rappresenta il programma. In tal modo ad algoritmi diversi corrispondono interi diversi: è così possibile ordinare gli algoritmi per interi crescenti e rinumerarli poi a partire da 1, stabilendo una corrispondenza biunivoca tra algoritmi e interi positivi. Identificheremo gli algoritmi con gli interi che li rappresentano.

2. Similmente i dati D possono essere messi in corrispondenza biunivoca con gli interi positivi. In tal modo lo stesso intero può rappresentare indipendentemente un algoritmo o un dato.

3. È dunque lecito, per due algoritmi A,B, considerare la computazione A(B) interpretando B come dato d'ingresso per A (in sostanza si chiede ad A di stabilire se B ha o meno una certa proprietà). È quindi anche lecito considerare la computazione A(A) secondo un meccanismo logico detto di *autoriferimento*.

4. Ammettiamo che esista un algoritmo di decisione TER per risolvere il problema della terminazione: dimostreremo che ciò conduce a un paradosso. Se TER esistesse potremmo applicarlo a un algoritmo arbitrario A per decidere la terminazione della computazione A(A). Avremmo:

$$\begin{aligned} \text{TER}(A, A) = \text{true} &\quad \Rightarrow \quad A(A) \text{ termina} \\ \text{TER}(A, A) = \text{false} &\quad \Rightarrow \quad A(A) \text{ non termina} \end{aligned}$$

¹ Il termine *indecidibile* si riferisce a problemi di decisione, il cui risultato è semplicemente *vero* o *falso*. Si tratta in sostanza di accertare se il problema ammette una soluzione con determinate proprietà senza richiedere di determinare la soluzione stessa. Per esempio, data un'equazione algebrica di forma determinata (lineare, quadratica ecc.) ma con coefficienti arbitrari, stabilire se essa ammette una soluzione in cui le variabili assumono valori interi senza richiedere di riportare tale soluzione, se esiste. Come vedremo sarà sufficiente riferirsi alla forma decisionale di molti problemi per determinarne l'intrinseca difficoltà. Gli algoritmi per la risoluzione di questi problemi sono detti a loro volta *algoritmi di decisione*.

5. Consideriamo ora il seguente algoritmo PAR (per paradosso) che accetta come dato d'ingresso un algoritmo arbitrario A:

```
PAR(A)
  p = true;
  while (p) p=TER(A,A);
```

La computazione PAR(A) termina se e solo se $p=TER(A,A)$ ha valore false; cioè, per il punto 4:

$PAR(A)$ termina $\Leftrightarrow A(A)$ non termina

6. Ovviamente PAR esiste se e solo se TER esiste, e in questo caso a PAR corrisponde un intero nella lista di tutti gli algoritmi. È dunque lecito eseguire la computazione PAR(PAR). Ma per quanto visto nel punto 5, sostituendo PAR al dato A abbiamo:

$PAR(PAR)$ termina $\Leftrightarrow PAR(PAR)$ non termina

Questa contraddizione ci obbliga a concludere che PAR non esiste, quindi non esiste TER.

Dai tempi di Turing sono stati scoperti molti altri problemi indecidibili costruiti in genere come trasformazione del problema della terminazione. Per esempio è indecidibile stabilire se un'equazione algebrica di grado arbitrario, in un arbitrario numero di variabili, possiede o meno una soluzione intera. In effetti si può dimostrare che, estratto un problema a caso, la probabilità che esso sia indecidibile è sostanzialmente 1: la nostra scarsa abilità nell'individuare problemi indecidibili si deve forse al fatto che i problemi che spontaneamente si pongono hanno una soluzione e, per quanto difficile, esiste un modo di trovarla. Tutto questo cade comunque nel campo della logica matematica e deve essere approfondito in testi specializzati: la nostra brevissima escursione presente serve unicamente a inquadrare il problema.

I problemi indecidibili sono dunque irrisolvibili in modo algoritmico. Vi sono però anche problemi decidibili che sono praticamente irrisolvibili poiché il tempo richiesto per risolverli cresce in modo esponenziale con le dimensioni dei dati. La divisione tra problemi *trattabili* e *intrattabili*, essenziale nella teoria degli algoritmi, si riferisce unicamente ai problemi decidibili e attiene all'esistenza di algoritmi polinomiali o esponenziali per la loro soluzione. Per esempio si può dimostrare che è decidibile stabilire se un'equazione algebrica di primo o secondo grado in due variabili ammette una soluzione intera, ma per equazioni di primo grado si conosce un algoritmo polinomiale di decisione mentre per equazioni di secondo grado si deve ricorrere a metodi enumerativi che richiedono un numero esponenziale di passi. Ci occuperemo ora di questo argomento e vedremo che un confine definitivo tra problemi trattabili e intrattabili non è ancora tracciabile con assoluta certezza.

2. Problemi polinomiali e esponenziali

La prima questione di trattabilità che vogliamo porre è se esistano problemi *intrinsecamente esponenziali*, tali cioè da richiedere, per essere risolti, tempo sicuramente esponenziale nella dimensione dei dati. Appartengono banalmente a questa famiglia tutti i problemi che richiedono risultati di lunghezza esponenziale: si può per esempio richiedere, dato un insieme di n elementi, che se ne costruiscano le $n!$ permutazioni. Ma problemi così formulati sono sostanzialmente non interessanti perché si chiede di produrre un risultato che non può poi essere realisticamente esaminato.

Il primo problema non banale intrinsecamente esponenziale fu individuato nel 1972 e da allora altri ne sono seguiti, afferenti in genere alla teoria dei linguaggi formali e alla logica matematica. Solo essi, oltre ai problemi indecidibili, sono dimostratamente intrattabili. Fortunatamente però questi problemi non sono in genere importanti in pratiche applicazioni, mentre si incontrano comunemente problemi decidibili che richiedono tempo esponenziale nel senso che non siamo capaci di risolverli meglio. Dirighiamo dunque su questi la nostra attenzione.

Partiamo di nuovo dai problemi di decisione, sufficienti a determinare la natura del fenomeno. Si dice che un tale problema è *risolvibile in tempo polinomiale* se esiste un algoritmo che stabilisce, in tempo polinomiale nella dimensione dei dati, se esiste una soluzione con la proprietà richiesta. Per esempio dato un grafo $G=(V,E)$ è possibile decidere algoritmicamente in tempo $O(|E| \log |V|)$ se esiste un percorso chiuso che attraversa tutti gli spigoli esattamente una volta (*ciclo Euleriano*). Si pone la definizione:

P è l'insieme dei problemi decisionali risolvibili in tempo polinomiale.

Il problema del ciclo Euleriano appartiene dunque a **P**.

Per molti altri problemi non è noto se esiste un algoritmo polinomiale di soluzione, ma un'informazione addizionale K , detta *certificato*, permette di *verificare* in tempo polinomiale se i dati D hanno la proprietà desiderata. Tecnicamente ciò significa che esiste un algoritmo di verifica $VER(K,D)$ che controlla in tempo polinomiale l'esistenza di una soluzione per D avente la proprietà richiesta. Per esempio dato un grafo $G=(V,E)$ possiamo chiederci se esiste un percorso chiuso che attraversa tutti i *vertici* (non gli spigoli) esattamente una volta (*ciclo Hamiltoniano*). Mentre non è noto se questo problema sia risolvibile in tempo polinomiale, un certificato K che specifichi l'ordine dei vertici incontrati nel ciclo, se questo esiste, permette di verificare in tempo polinomiale che G contiene un ciclo controllando se esistono gli spigoli tra i vertici consecutivi indicati in K . Si pone la definizione:

NP è l'insieme dei problemi decisionali verificabili in tempo polinomiale.

Il problema del ciclo Hamiltoniano appartiene dunque a **NP**.

Mentre la definizione della classe **P** è semplice e intuitiva, quella della classe **NP** è obiettivamente artificiale e richiede alcune riflessioni perché se ne comprenda la portata. Anzitutto il certificato K deve avere lunghezza polinomiale nella dimensione di D altrimenti non sarebbe esaminabile in tale tempo: si parla dunque in genere di *certificato polinomiale*. Inoltre la definizione di **NP** implica che sia costruibile un certificato K solo per i dati D che hanno la proprietà richiesta: si certifica cioè l'esistenza della proprietà ma non la sua inesistenza. Di nuovo tutto ciò sembra assai artificiale: se un grafo G possiede un ciclo Hamiltoniano il certificato attesta che tale ciclo esiste, ma se G non possiede un tale ciclo, non è richiesto (e non è comunque sicuro) che ciò sia verificabile. Queste apparenti asimmetrie si spiegano esaminando il modo in cui un problema in **NP** può essere praticamente risolto, al di là delle definizioni formali.

Per risolvere un problema in **NP** è necessario ricorrere a un metodo enumerativo che esegua un numero esponenziale di prove per verificare se almeno una di queste corrisponde a una soluzione accettabile. Per esempio in un grafo i cui vertici siano numerati da 1 a n , si possono considerare uno a uno tutti i vertici raggiungibili dal vertice 1 lungo uno spigolo, da ciascuno di questi tutti i vertici successivamente raggiungibili lungo un nuovo spigolo, e così via, costruendo tutti i cammini lunghi n e verificando man mano se uno di essi è un ciclo Hamiltoniano su cui l'algoritmo si arresta. Questo procedimento è evidentemente esponenziale in n , perché tanti possono essere i cammini generati. Un certificato per il problema è in genere un'informazione sulla sequenza di successive scelte di spigoli che corrispondono a un cammino Hamiltoniano, se esso esiste. Noto il certificato si possono seguire queste scelte raggiungendo direttamente la soluzione in n passi, e verificare quindi in tempo polinomiale che essa esiste. Se però un ciclo Hamiltoniano non esiste, l'algoritmo enumerativo deve generare tutti i cammini possibili prima di arrestarsi con risposta negativa, e a questo non corrisponde un certificato di lunghezza polinomiale. In sostanza un algoritmo enumerativo che incontra $p(n)$ punti nei quali eseguire una scelta tra due o più alternative (per esempio una **frase if** di un programma), ove $p(n)$ è un polinomio nella dimensione n dei dati, termina nel caso pessimo in un numero esponenziale di prove per raggiungere una soluzione, ma le scelte che portano a questa possono essere specificate in un certificato di lunghezza $p(n)$, il che giustifica la definizione formale della classe **NP**.

Possiamo ora osservare che per ogni problema in **P** si può verificare l'esistenza di una soluzione in tempo polinomiale applicando un certificato vuoto, ignorato nella verifica, e risolvendo direttamente il problema. Da ciò segue il risultato fondamentale:

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

Non è però mai stato dimostrato se vale il contenimento in senso stretto tra le due classi. Tale quesito è il più importante problema irrisolto nella teoria della complessità, anche se molti risultati collaterali e oltre trent'anni di intensa ricerca inducono a credere che **P** sia strettamente un sottoinsieme di **NP**; ovvero esistano problemi in **NP**, come quello del ciclo Hamiltoniano, che *non sono risolvibili* in tempo polinomiale. In ogni caso finché non sarà dimostrato il contrario dovremo considerare intrattabili i problemi in **NP** per cui non conosciamo un algoritmo polinomiale di soluzione. La situazione è comunque più complicata di così, come ora vedremo.

3. Il problema della soddisfattibilità

Un problema chiave nello studio dell'intrattabilità riguarda le forme normali congiuntive booleane. Date n variabili booleane x_1, x_2, \dots, x_n , una *clausola* è definita come OR di un sottoinsieme di esse, dirette o negate (dette genericamente *letterali*). Per esempio la clausola $C = (x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4)$. Una *forma normale congiuntiva* (CNF) è l'AND di diverse clausole; per esempio la forma $F = C_1 \wedge C_2 \wedge C_3$. Il *problema della soddisfattibilità* P_S è il seguente:

P_S : data un'arbitraria forma normale congiuntiva F su n variabili, stabilire se esiste un assegnamento di valori alle variabili che renda vera la F .

Per esempio la $F = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3)$ è soddisfattibile perché i valori $x_1=\text{true}$, $x_2=\text{false}$, $x_3=\text{false}$ rendono $F=\text{true}$.

Ovviamente P_S può essere risolto per enumerazione assegnando alle n variabili tutte le 2^n possibili combinazioni di valori. E in effetti P_S è in **NP** poiché, se la forma F è soddisfatta da un assegnamento A di valori, si può associare alla F un certificato K costituito da una sequenza di n bit, $K = b_1 b_2 \dots b_n$, con $b_i=0$ se $x_i=\text{true}$ nell'assegnamento A , $b_i=1$ se $x_i=\text{false}$; sostituendo questi valori nella F si verificherà che essa è soddisfatta. Vedremo ora che il problema P_S ha un ruolo centrale nella teoria della complessità.

Continua nei paragrafi 8.2.1 e 8.2.2 del testo citato in testa, riportati nella dispensa N. 21.