Tecniche di Derandomizzazione: il caso di Arthur-Merlin

Dario Balboni

19 Aprile 2018

Indice

1 Introduzione ai modelli di computazione

Nel seguito definiamo in maniera intuitiva i principali modelli di computazione. Tali modelli saranno tutti orientati a problemi decisionali, ovvero data una stringa binaria di lunghezza finita decidere se essa appartiene o meno ad un certo linguaggio. Ricordiamo che per **linguaggio** si intende un qualunque sottoinsieme di $\{0,1\}^*$.

1.1 Macchine di Turing

Una macchina di Turing è una macchina con un numero finito di stati, che può leggere da un nastro di input, scrivere su un nastro di output ed ha vari nastri di lavoro che funzionano come delle memorie su cui può scrivere ciò che vuole. Tutti i nastri hanno un numero infinito di celle sia verso destra che verso sinistra. Ad ogni passo la macchina può leggere un simbolo da un nastro di lavoro o da quello di input, spostare le teste di ogni nastro di un posto a destra o a sinistra e dire qual è lo stato successivo della macchina.

Esiste uno stato iniziale da cui la macchina parte ed uno stato finale in cui la macchina si ferma. Prima dell'inizio della macchina sul nastro di input può venire scritta una certa stringa, ed al termine della procedura il risultato dell'elaborazione è la stringa presente sul nastro di input. Nel caso dei problemi decisionali, sul nastro di output verrà scritto un solo simbolo: 1 se la stringa in input è accettata, 0 se essa viene rigettata.

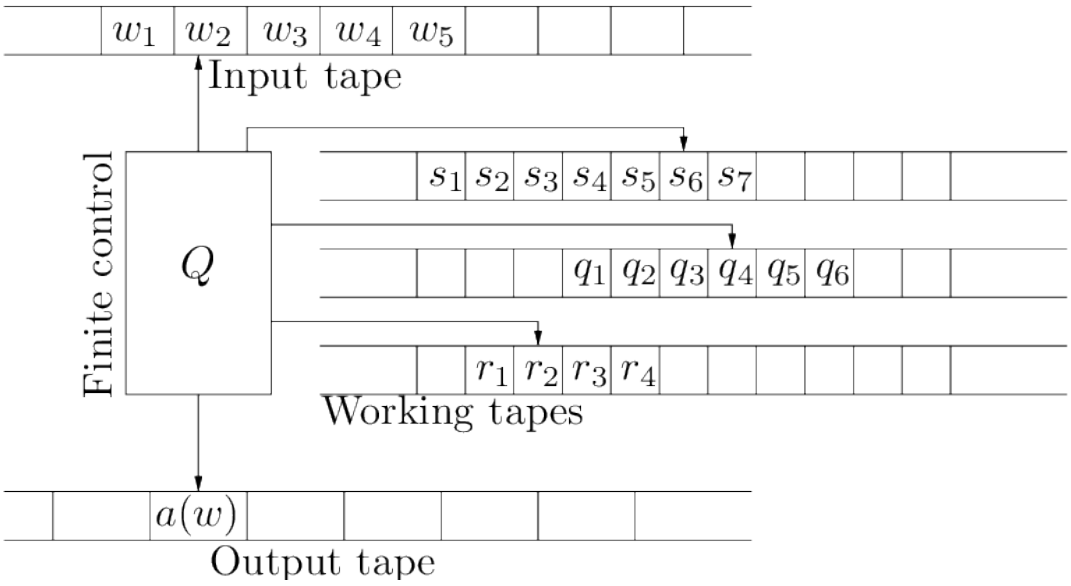

Figura 1: Macchina di Turing deterministica multinastro

Esistono diverse varianti di tipi di macchine di Turing:

- **deterministiche**: Sono quelle descritte fino ad adesso.
- **nondeterministiche**: Ad ogni passo la macchina può spostarsi verso un insieme di nuovi stati (e non solamente verso uno solo). La stringa si intende accettata se esiste uno stato di arrivo che è accettante.
- **co-nondeterministiche**: Ad ogni passo la macchina può spostarsi verso un insieme di nuovi stati. La stringa si intende accettata se tutti gli stati a cui si arriva sono accettanti.

- **con suggerimenti**: La macchina ha un nastro aggiuntivo da cui può solo leggere che può contenere delle informazioni aggiuntive. Viene ad esempio utilizzato per modellizare la possibilità di scelte casuali della macchina.

1.2 Principali classi di Complessità

Nel seguito diamo la definizione di alcune classi di complessità per problemi decisionali.

1.2.1 Classi senza interazione

- **DTIME**$(t(n))$: Un linguaggio $L \in$ **DTIME**$(t(n))$ se esiste una macchina di Turing *deterministica* M tale che il linguaggio è l'insieme delle stringhe accettate da M in un *numero di passi* $\leq t(n)$ (dove n indica la dimensione dell'input).

  Da questa si definiscono varie classi:

  - **P**: È la classe di linguaggi deterministici decidibili in tempo polinomiale, ovvero l'unione di **DTIME**$(p(n))$ al variare di p polinomio.
  - **E**: È l'unione di **DTIME**(2^{cn}) al variare di $c \in \mathbb{R}$.

- **DSPACE**$(t(n))$: Un linguaggio $L \in$ **DSPACE**$(t(n))$ se esiste una macchina di Turing *deterministica* M tale che il linguaggio è l'insieme delle stringhe accettate da M utilizzando al più $t(n)$ *celle di nastro.*

  In particolare si definisce **PSPACE** come l'unione di **DSPACE**$(p(n))$ al variare di p polinomio.

- **NTIME**$(t(n))$: Come **DTIME**$(t(n))$ ma la macchina di Turing è *nondeterministica.*

  Analogamente a **P** ed **E** si definiscono **NP** ed **NE**.

- **coNTIME**$(t(n))$: Come **DTIME**$(t(n))$ ma la macchina di Turing è *co-nondeterministica.*

  Analogamente a **P** ed **E** si definiscono **coNP** e **coNE**.

- **BPTIME**$(t(n))$: La macchina di Turing è *deterministica con suggerimento.* Affinché una stringa x sia accettata serve che al variare di ogni possibile stringa di suggerimento, la probabilità con cui x viene accettata è superiore ai $\frac{2}{3}$. Inoltre affinché una stringa x venga rigettata, la probabilità che ciò accada al variare del suggerimento deve essere inferiore ad $\frac{1}{3}$.

  Analogamente a **P** si definisce **BPP**.

1.2.2 Classi di complessità con interazione

Reinterpretazione di NP Affinché una stringa x sia accettata da una macchina di Turing nondeterministica M serve che vi sia un percorso valido sull'insieme degli stati di M che porta la macchina ad accettare. Tale percorso può essere codificato in una stringa w che viene chiamata testimone dell'appartenenza di x al linguaggio.

Possiamo pensare a ciò come un gioco tra Peggy e Victor:

- Peggy deve dimostrare a Victor che $x \in L$
- Victor deve poter verificare che Peggy dica la verità e poter scoprire se invece sta barando utilizzano poche risorse

Per fare ciò Peggy invia a Victor la stringa w ed egli utilizza le sue risorse per verificarla. Nel caso di NP le risorse che Victor ha a disposizione sono P, ovvero dato x e w deve impiegare al più tempo polinomiale per verificare $x \in L$.

Merlin-Arthur (Monete private) Protocollo interattivo con un solo messaggio:

- Merlin manda ad Arthur una "dimostrazione" che $x \in L$
- Arthur decide se accettare o meno utilizzando una computazione **BPP** (tempo probabilistico polinomiale)

Questo è molto simile alla reinterpretazione di **NP**, tranne per il fatto che per la verifica Arthur può usare algoritmi probabilistici. Ovvero se per tutte le stringhe nel linguaggio, Merlin può mandare ad Arthur una dimostrazione polinomiale per convincerlo di questo fatto con alta probabilità, mentre per ogni stringa che non appartiene al linguaggio, non esiste una dimostrazione che possa convincere Arthur con grande probabilità.

Formalmente, un linguaggio è in **MA** se esiste una macchina di Turing deterministica M polinomiale e due polinomi p, q tali che per ogni stringa x si abbia:

- $x \in L \implies \exists z \in \{0,1\}^{q(n)} : \Pr_{y \in \{0,1\}^{p(n)}} [M(x,y,z) = 1] \geq \frac{2}{3}$
- $x \notin L \implies \forall z \in \{0,1\}^{q(n)} : \Pr_{y \in \{0,1\}^{p(n)}} [M(x,y,z) = 1] \leq \frac{1}{3}$

Arthur-Merlin (Monete Pubbliche) La classe Arthur-Merlin generalizza la precedente interpretazione come "gioco" di **NP**. È l'insieme dei linguaggi che possono essere decisi in tempo polinomiale da un protocollo interattivo tra Arthur e Merlin con due messaggi:

- Arthur lancia qualche moneta random e manda il risultato di *tutti* i suoi lanci a Merlin
- Merlin risponde con una supposta dimostrazione
- Arthur in maniera deterministica verifica la dimostrazione, utilizzando al più i lanci di moneta noti a Merlin

Questo procedimento deve avere le seguenti proprietà:

- Se la stringa è nel linguaggio, deve esistere una dimostrazione di Merlin che molto probabilmente (rispetto ai lanci di moneta di Arthur) convince Arthur della cosa
- Se la stringa non è nel linguaggio, indipendentemente da quale stringa "dimostrazione" scelga Merlin, Arthur molto probabilmente scoprirà l'inganno

Formalmente, un linguaggio è in **AM** se esiste una macchina di Turing deterministica M che gira in tempo polinomiale e due polinomi p, q tali che per ogni stringa in input x di lunghezza $n = |x|$ si abbia:

- $x \in L \implies \Pr_{y \in \{0,1\}^{p(n)}} \left[\exists z \in \{0,1\}^{q(n)} \quad M(x,y,z) = 1 \right] \geq \frac{2}{3}$

- $x \notin L \implies \Pr_{y \in \{0,1\}^{p(n)}} \left[\exists z \in \{0,1\}^{q(n)} \quad M(x,y,z) = 1 \right] \leq \frac{1}{3}$

La classe di complessità $\mathbf{AM}[k]$ è l'insieme di problemi che può essere deciso in tempo polinomiale, con k domande e risposte. **AM** come definito sopra corrisponde ad $\mathbf{AM}[2]$. L'ultimo messaggio deve sempre andare da Merlin ad Arthur, visto che non può aiutare Arthur mandare un messaggio a Merlin subito prima di decidere.

Relazioni tra AM ed MA

- Sia **AM** che **MA** rimangono invariate se si richiede una completezza perfetta, ovvero che Arthur accetti con probabilità 1 quando $x \in L$ (invece che con probabilità $\frac{2}{3}$)
- Per ogni costante $k \geq 2$ si ha $\mathbf{AM}[k] = \mathbf{AM}[2] = \mathbf{AM}$: vedere Babai and Moran [1988].
- $\mathbf{MA} \subseteq \mathbf{AM}[3] = \mathbf{AM}$ siccome Arthur può, dopo aver ricevuto il certificato da Merlin, tirare un opportuno numero di monete, mandarle a Merlin, ed ignorare la risposta
- I linguaggi riconosciuti non cambiano se si permette ad Arthur di tenere nascosti i lanci delle sue monete.

1.3 Circuiti

Un circuito è un grafo diretto aciclico dove i nodi intermedi sono porte logiche $(\wedge, \vee, \neg)$. Ha un po' di celle di input $(x_1, \ldots, x_n)$, un po' di porte "di scelta" $(y_1, \ldots, y_k)$ ed un po' di output $(o_1, \ldots, o_s)$. Il valore delle uscite si calcola in modo univoco dati i valori in input e quelli delle porte di scelta. In quest'ultimo senso essi precisano un modello di computazione.

Per fissare la notazione chiameremo $\mathbf{x} = (x_1, \ldots, x_n)$ e simili e diremo che $\mathbf{o} = \mathcal{C}(\mathbf{x}, \mathbf{y})$. Inoltre si dice dimensione del circuito il numero di porte utilizzate.

Vediamo nel dettaglio i tipi di circuiti che ci servirà conoscere e in che modo rappresentano dei linguaggi:

- **deterministici**: Hanno un unico bit di output e nessuna porta di scelta e $x \in L_{\mathcal{C}} \Leftrightarrow \mathcal{C}(x) = 1$
- **nondeterministici**: Essi hanno un unico bit di output e $x \in L_{\mathcal{C}} \Leftrightarrow \exists \mathbf{y} : \mathcal{C}(x, y) = 1$
- **co-nondeterministici**: Hanno un unico bit di output e $x \in L_{\mathcal{C}} \Leftrightarrow \forall \mathbf{y} : \mathcal{C}(x, y) = 1$
- **SV-nondeterministici**: Hanno due bit di output (il secondo è da interpretarsi come "flag" di validità). Per loro deve valere che $\forall \mathbf{x} : \exists o_1 \in \{0,1\} : \forall \mathbf{y} : \mathcal{C}(x,y) = (\alpha, 1) \implies \alpha = o_1$ e definiscono il linguaggio $x \in L_{\mathcal{C}} \Leftrightarrow \exists \mathbf{y} : \mathcal{C}(x, y) = (1, 1)$.

2 Alcuni problemi in AM

2.1 Non-isomorfismo di Grafi

Dati due grafi G_0, G_1 si chiede di dimostrare che essi **non** sono isomorfi. Diamo di seguito un possibile modo di farlo se i lanci di monetà di Arthur fossero privati (ovvero se fossimo in **MA**). Si può fare anche con lanci pubblici ma ciò è più lungo da mostrare.

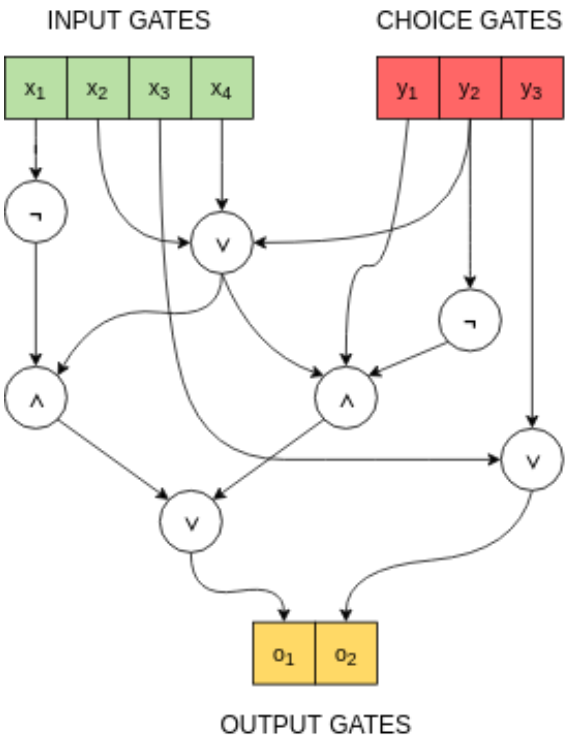

Figura 2: Esempio di circuito con porte di scelta

Versione con i lanci privati: Supponiamo che G_0 e G_1 abbiano lo stesso numero di vertici n (altrimenti il problema è banalmente vero). Arthur lancia una moneta che ha come risultato $b \in \{0, 1\}$ e successivamente lancia altre monete per generare una permutazione casuale $\pi \in S_n$. Ora Arthur presenta a Merlin il grafo $\pi(G_b)$ e gli chiede di dire se esso è uguale a G_0 oppure a G_1.

Se i due grafi sono effettivamente distinti Merlin può riuscire a riconoscere quale grafo è $\pi(G_b)$ e dare la risposta ad Arthur con probabilità di successo $= 1$. Se i due grafi sono isomorfi Merlin può solo indovinare quale dei due Arthur ha preso e quindi ha probabilità di successo $= \frac{1}{2}$.

2.2 Esistenza di uno zero comune di equazioni polinomiali

Si può mostrare (vedere Koiran [1996]) che, supponendo vera l'ipotesi di Riemann estesa (ERH), anche il seguente problema è in **AM**: Dati $f_1, \ldots, f_s \in \mathbb{Z}[x_1, \ldots, x_k]$ polinomi a coefficienti interi in più variabili, stabilire se essi hanno uno zero complesso in comune, ovvero se $\exists \mathbf{y} \in \mathbb{C}^k : f_i(\mathbf{y}) = 0$ per ogni $i = 1, \ldots, s$.

3 Le idee della derandomizzazione

Vediamo qualche idea di base per la derandomizzazione delle varie classi di complessità.

3.1 Enumerazione

Proposizione 1: Se $A(x,r)$ è un algoritmo randomizzato per un linguaggio L che termina in tempo $t(n)$ e richiede $r(n)$ bit random allora $L \in \mathbf{DTIME}(t(n) \cdot 2^{r(n)})$.

Basato sul fatto che

$$\Pr_{r \in \{0,1\}^{r(n)}} [A(x,r) = 1] = \frac{1}{2^{r(n)}} \sum_{r \in \{0,1\}^{r(n)}} A(x,r)$$

e l'espressione a destra si può calcolare per enumerazione di r.

Questo consente di dire che $\mathbf{BPP} \subseteq \mathbf{EXP}$ ma anche che se un algoritmo $f \in \mathbf{BPP}$ utilizza al più $O(\log n)$ bit casuali allora può essere simulato in $\mathbf{P}$. $\mathbf{BPP} \subseteq \mathbf{EXP}$ è attualmente l'unica derandomizzazione di $\mathbf{BPP}$ nota che non dipenda da congetture.

3.2 Derandomizzazione non costruttiva / non uniforme

Proposizione 2: Se $A(x,r)$ è un algoritmo randomizzato per un linguaggio L che ha probabilità di errore più piccola di 2^{-n} sugli input x di lunghezza n, allora per ogni n esiste una sequenza di stringhe r_n tali che $A(x,r_n)$ è corretto per tutti gli $x \in \{0,1\}^n$.

Dimostrazione. Scelto r_n uniformemente da $\{0,1\}^{r(n)}$ si ha

$$\begin{aligned}\Pr_{r_n \in \{0,1\}^{r(n)}} [\exists x \in \{0,1\}^n : A(x,r_n) \text{ incorretto su } x] &\leq \sum_{x \in \{0,1\}^n} \Pr_{r_n \in \{0,1\}^{r(n)}} [A(x,r_n) \text{ incorretto su } x] \\ &< 2^n \cdot 2^{-n} = 1\end{aligned}$$

e quindi $\exists r_n \in \{0,1\}^{r(n)}$ tale che $\forall x \in \{0,1\}^n : A(x,r_n)$ è corretto. □

Si può vedere che ogni linguaggio in $\mathbf{BPP}$ ammette un algoritmo con probabilità di errore arbitrariamente piccola e quindi la proposizione è generale. Inoltre, una volta trovate le stringhe r_n, l'algoritmo risultante è deterministico e polinomiale. L'esistenza delle stringhe r_n è non-costruttiva e non è chiaro come ottenerle in tempo meno che esponenziale.

4 Derandomizzazione di Arthur-Merlin utilizzando gli Hitting sets

Nel seguito analizziamo nel dettaglio una procedura più macchinosa di derandomizzazione. In particolare vogliamo arrivare a dimostrare il seguente risultato:

Teorema 3 (Miltersen-Vinodchandran)**:** Sia $\varepsilon > 0$ una costante. Se esiste un linguaggio $L \in \mathbf{coNE} \cap \mathbf{NE}$ tale che per tutti tranne al più un numero finito di n, $L \cap \{0,1\}^n$ abbisogna di circuiti SV-nondeterministici di dimensione almeno $2^{\varepsilon n}$, allora $\mathbf{AM} = \mathbf{NP}$.

4.1 Definizione ed idea generale

Definizione 1 (Hitting Set)**:** Un sottoinsieme $H \subseteq \{0,1\}^n$ si dice Hitting Set di soglia δ per i circuiti di tipo T se vale l'implicazione

$$\Pr_{x \in \{0,1\}^n} [\mathcal{C}(x) = 1] \geq \delta \implies \exists x \in H : \mathcal{C}(x) = 1$$

per ogni circuito $\mathcal{C}$ di tipo T.

Il motivo principale che ci spinge a cercare di costruire degli hitting sets è che essi ci permettono di derandomizzare le varie classi di complessità, come possiamo vedere più nel dettaglio nel lemma seguente.

Lemma 4: Se c'è una procedura SVNP che, sull'input 1^n dà in output un hitting set in $\{0,1\}^n$ di soglia $\frac{1}{2}$ per i circuiti co-nondeterministici di dimensione n, allora $\mathbf{AM} = \mathbf{NP}$.

Dimostrazione. **AM** corrisponde a

$$\begin{aligned} x \in L &\implies \Pr_{y \in \{0,1\}^{p(n)}} \left[\exists z \in \{0,1\}^{p(n)} : C(x,y,z) = 1 \right] = 1 \\ &\implies \forall y \in \{0,1\}^{p(n)} : \exists z \in \{0,1\}^{p(n)} : C(x,y,z) = 1 \end{aligned}$$

$$\begin{aligned} x \notin L &\implies \Pr_{y \in \{0,1\}^{p(n)}} \left[\exists z \in \{0,1\}^{p(n)} : C(x,y,z) = 1 \right] \leq \frac{1}{2} \\ &\implies \Pr_{y \in \{0,1\}^{p(n)}} \left[\forall z \in \{0,1\}^{p(n)} : C(x,y,z) = 0 \right] \geq \frac{1}{2} \\ &\implies \exists y \in H_{p(n)} : \forall z \in \{0,1\}^{p(n)} : C(x,y,z) = 0 \end{aligned}$$

avendo utilizzato l'hitting set H nell'ultima implicazione.

Prendendone le contronomiali segue che:

$$\begin{aligned} \forall y \in H_{p(n)} : \exists z \in \{0,1\}^{p(n)} : C(x,y,z) = 1 &\implies x \in L \\ \exists y \in \{0,1\}^{p(n)} : \forall z \in \{0,1\}^{p(n)} : C(x,y,z) = 0 &\implies x \notin L \end{aligned}$$

da cui si vede, indebolendo l'ultima a $\exists y \in H_{p(n)} \ldots$, che si ha $x \in L \Leftrightarrow \forall y \in H_{p(n)} : \exists z \in \{0,1\}^{p(n)} : C(x,y,z) = 1$. Tale problema sta in **NP** in quanto $H_{p(n)}$ ha un numero polinomiale in n di elementi e si può enumerare su di esso, mentre la restante condizione è un problema in **NP**. □

4.2 Lemmi preliminari

Lemma 5 (Estensioni di grado basso)**:** Dati $S_1, \ldots, S_k \subseteq \mathbb{F}$ sottoinsiemi finiti ed $f : S_1 \times \ldots \times S_k \to \mathbb{F}$ una funzione qualunque, esiste un'unica estensione $\tilde{f} : \mathbb{F} \times \ldots \times \mathbb{F} \to \mathbb{F}$ tale che essa sia un polinomio di grado minore stretto di $|S_i|$ nella i esima variabile e che $\tilde{f}\,|_{S_1 \times \ldots S_k} \equiv f$.

Dimostrazione. Sia $u = (u_1, \ldots, u_k) \in S = S_1 \times \ldots \times S_k$. Il polinomio

$$g_u(x) = \prod_{i=1}^{m} \prod_{s \in S_i \setminus \{u_i\}} (x_i - s)$$

è tale che $g_u(u) \neq 0$ mentre $g_u(x) = 0$ per tutti gli $x \in S \setminus \{u\}$. Chiaramente ogni funzione da S in $\mathbb{F}$ è combinazione lineare delle funzioni g_u al variare di $u \in S$ e questo dimostra la parte dell'esistenza. L'unicità può essere facilmente dimostrata per induzione sul numero delle variabili. □

Definizione 2 (Vettore associato ad un polinomio)**:** Dato un polinomio $p \in \mathbb{F}[x]$ su un campo finito, denotiamo con L_p il vettore $(p(i))_{i \in \mathbb{F}} \in \mathbb{F}^{|\mathbb{F}|}$ delle sue valutazioni.

Lemma 6 (Proiezioni iniettive di spazi di polinomi)**:** Sia $\mathbb{F}$ un campo finito con più di due elementi e consideriamo $\mathcal{L} = \left\{ L_p \mid p \text{ è un polinomio di grado } \leq |\mathbb{F}|^{\frac{1}{2}} \right\}$. Allora, per ogni insieme $Z \subseteq \mathbb{F}^{|\mathbb{F}|}$, esiste un insieme di indici $S \subseteq \{1, \ldots, |\mathbb{F}|\}$, con $|S| \leq \lceil \log |Z| \rceil$ tale che la proiezione $\pi_S : \mathcal{L} \cap Z \to \mathbb{F}^S$ è 1-1.

Dimostrazione. Osserviamo che per $x \neq x \in \mathcal{L}$, essi coincidono su meno di $\frac{1}{4}$ degli indici. Sia $r = \lceil \log |Z| \rceil$ e costruiamo induttivamente $S = \{j_1, \ldots, j_r\}$: dato $S_i = \{j_1, \ldots, j_i\}$ costruiamo $S_{i+1} = S_i \cup \{j_{i+1}\}$ come segue.

Definiamo $Y_i = \{(x, y) \mid x, y \in \mathcal{L} \cap Z, x \neq y, \pi_{S_i}(x) = \pi_{S_i}(y)\}$. Per ogni $(x, y) \in Y_i$ fissato si ha, come già notato, che esistono almeno $\frac{3}{4}$ degli indici su cui $x_j \neq y_j$. Quindi esiste almeno un indice j che separa almeno i $\frac{3}{4}$ delle coppie di polinomi. Quindi si ha $j_{i+1} = j$ e $|Y_{i+1}| \leq \frac{1}{4} |Y_i|$.

Da $\frac{1}{4^r} \binom{|Z|}{2} < 1$ segue la tesi (siamo riusciti a separare tutte le coppie di polinomi). □

4.3 Amplificazione di un Hitting Set

In realtà è abbastanza ottenere un Hitting Set di soglia anche molto vicina ad uno, come mostrano i lemmi seguenti.

Definizione 3 (Dispersore)**:** Un dispersore di soglia t è un grafo bipartito $G = (U, V, E)$ tale che per ogni sottoinsieme $S \subseteq U$ con $|S| \geq t$, più della metà dei vertici di V sono adiacenti ad S.

Definizione 4 (Dispersore esplicito)**:** Per costanti fissate $\delta, \gamma > 0$ e $k \geq 1$, un (n^δ, n^γ) dispersore esplicito è una famiglia di dispersori $G_n = (U_n, V_n, E_n)$ per $n = 1, \ldots$ con $U_n = \{0,1\}^n, V_n = \{0,1\}^{\lceil n^\gamma \rceil}$ e di soglia $t_n = 2^{n^\delta}$ tale che esista un algoritmo in tempo deterministico polinomiale che, dato in input $x \in U_n$, enumera i vertici in V_n adiacenti ad x.

Teorema 7 (Saks-Srinivasan-Zhou)**:** Per ogni $\delta > 0$ fissato esiste un $\gamma > 0$ tale che esiste un (n^δ, n^γ) dispersore esplicito.

Dimostrazione. Per la dimostrazione vedere Saks, Srinivasan, and Zhou [1998]. □

Lemma 8 (Amplificazione di un Hitting Set)**:** Per ogni costante $\delta \geq 0$, esistono costanti $q \geq 1, \gamma > 0$ tali che vi è una procedura polinomiale che, dato in input un hitting set $H \subseteq \{0,1\}^n$ di soglia $1 - 2^{-n+n^\delta}$ per i circuiti co-nondeterministici di dimensione n^q, dà in output un hitting set in $\{0,1\}^{n'}$ con soglia $\frac{1}{2}$ per i circuiti co-nondeterministici di dimensione $n' = \lceil n^\gamma \rceil$.

Dimostrazione. Si fissi $\delta > 0$. Per quanto già detto, esiste un (n^δ, n^γ) dispersore esplicito. Sia esso $G_n = (U_n, V_n, E_n)$ dove $n' = \lceil n^\gamma \rceil$, $U_n = \{0,1\}^n$, $V_n = \{0,1\}^{n'}$. Sia inoltre $H \subseteq \{0,1\}^n = U_n$ un hitting set di soglia $1 - 2^{-n+n^\delta}$ per i circuiti co-nondeterministici di dimensione n^q, con q da determinare.

Sia H' l'insieme dei vertici in V_n adiacenti a H. Siccome il dispersore è esplicito, H' può essere generato in tempo polinomiale da H. Il claim è che H' sia un hitting set di soglia $\frac{1}{2}$ per i circuiti co-nondeterministici di dimensione n'.

Infatti, si prenda un qualunque circuito co-nondeterministico C' di dimensione n' con n' inputs tale che $|Z(C')| \leq 2^{n'-1}$. Dobbiamo mostrare che $H' \not\subseteq Z(C')$. Per fare ciò costruiamo un circuito co-nondeterminisio C come segue: $C(x) = 1 \Leftrightarrow \exists y : (x,y) \in E_n \wedge C'(y)$. Siccome il dispersore è esplicito, la dimensione di questo circuito può essere resa polinomiale. Si fissi ora q in modo che n^q sia un limite superiore alla sua dimensione.

Ora deve valere $|Z(C)| < 2^{n^\delta}$: altrimenti, essendo G_n un dispersore, i vicini in V_n di $Z(C)$ sarebbero più della metà di V_n e quindi i vicini dovrebbero intersecare $V_n \setminus Z(C')$. Ovvero, per qualche y adiacente a $x \in Z(C)$ si ha $C'(y) = 1$ che implica $C(x) = 1$, contraddicendo $x \in Z(C)$.

Siccome $|Z(C)| < 2^{n^\delta}$, la probabilità di accettazione di C è di almeno $1 - 2^{-n+n^\delta}$. Quindi, siccome H è un hitting set, per qualche $x \in H$ si ha $C(x) = 1$. Ciò significa che per qualche $y \in V_n$, adiacente a qualche $x \in H$ si ha $C'(y) = 1$. Ma tale y per definizione è in H', ovvero H' è un hitting set per C'. □

4.4 Costruzione effettiva

Teorema 9 (Costruzione di un Hitting Set)**:** Nel seguito sia f una funzione che non può essere computata da circuiti SV-nondeterministici di dimensione minore di $2^{\varepsilon m}$ per quasi tutti gli m. Per ogni $\varepsilon > 0$ e $q \geq 1$ esiste una procedura polinomiale P che, data in input la tabella di verità di una tale $f : \{0,1\}^{kl} \to \{0,1\}$, P produce in output un hitting set $H_f \subseteq \{0,1\}^n$ per i circuiti co-nondeterministici di dimensione n^q con soglia $1 - 2^{-n+n^\delta}$, dove $n = (2l)2^{2l}$.

Dimostrazione. I valori k e δ saranno fissati in seguito in dipendenza di ε e di q.

Prima di tutto diamo una procedura per costruire H_f:

- Consideriamo f come mappa $f : \left(\{0,1\}^l\right)^k \to \{0,1\}$
- Sia $\mathbb{F}$ il campo finito con 2^{2l} elementi e consideriamo un embedding $\{0,1\}^l \to \{0,1\}^{2l}$ in modo che le operazioni aritmetiche siano efficienti
- Estendiamo f utilizzando le estensioni di grado basso ad una mappa $\tilde{f} : \mathbb{F}^k \to \mathbb{F}$
- Definiamo H_f come le tabulazioni delle restrizioni di $\tilde{f}$ ad ogni retta parallela agli assi in $\mathbb{F}^k$, ovvero $H_i = \{(f(a_1, \ldots, a_{i-1}, j, a_{i+1}, \ldots, a_k))_{j \in \mathbb{F}} \mid a_1, \ldots, a_{i-1}, a_{i+1}, \ldots a_k \in \mathbb{F}\}$ e $H = \cup_i H_i$.

- Generare H_f prende tempo polinomiale nella tabella di verità di f: $H_f \simeq k \cdot \mathbb{F}^k \cdot \text{val}(f) \leq k \cdot w^{4lk}$ e quindi $H_f \leq k \cdot \text{Tab}(f)^4$, visto che $\text{Tab}(f) = 2^{lk}$.

Vediamo ora la correttezza: supponiamo per assurdo che H_f non sia un hitting set per i circuiti co-nondeterministici e mostreremo che f ha un circuito SV-nondeterministico più piccolo di ciò che si assume avere.

- Sia C il circuito che viola l'hitting set, ovvero $C : \{0,1\}^n \to \{0,1\}$, ha dimensione n^q e denotiamo con Z l'insieme dei suoi zeri (ovvero tali che c'è un qualche settaggio delle sue porte di scelta nondeterministiche che lo fa valutare a zero). Allora $|Z| \leq 2^{n^\delta}$ e $H_f \subseteq Z$.
- Sia $\mathcal{L} \subseteq \mathbb{F}^{2^{2l}}$ i vettori delle valutazioni dei polinomi univariati di grado minore di 2^l. Vedendo $\mathcal{L}$ come sottoinsieme di $\{0,1\}^n$ si ha per costruzione $H_f \subseteq \mathcal{L}$. Inoltre per costruzione $H_f \subseteq Z$. Per il lemma sulle mappe iniettive di polinomi si ha l'esistenza di un insieme S di indici di dimensione $\leq n^\delta$ tale che $\pi_S : \mathcal{L} \cap Z \to \mathbb{F}^S$ è 1-1. Si fissi tale S.
- Ora daremo una procedura SV-nondeterministica efficiente che computa $\tilde{f}$ dati i seguenti indizi non-uniformi: il circuito C, l'insieme S ed una tabella delle restrizioni di $\tilde{f}$ ad S^k. Molto semplicemente per computare $\tilde{f}(a_1, \ldots, a_k)$ lo eseguiamo in k stadi: detto T_i la tabella di valori $\tilde{f}(a_1, \ldots, a_i, S^{k-i})$, noi abbiamo T_0 e vogliamo computare T_k.
- La procedura per costruire T_{i+1} dato T_i, C, S è la seguente:

  1. Per ogni $u \in S^{k-i}$ si esegue
  2. Indovina $v \in \mathbb{F}^{2l}$
  3. Verificare che $v \in \mathcal{L}$, $C(v) = 0$, $\pi_S(v) = \pi_S(\tilde{f}(a_1, \ldots, a_{i-1}, j, u)_{j \in \mathbb{F}})$
  4. Se tutte le verifiche vanno a buon fine allora v è il valore di $\tilde{f}(a_1, \ldots, a_i, u)$ in T_i

  Infatti supponiamo di avere un vettore v che soddisfi le condizioni nel passo 3. $v \in \mathcal{L}$ e $C(v) = 0$ assicurano che $v \in \mathcal{L} \cap Z$. Inoltre, per il lemma sulle funzioni iniettive di polinomi, per ogni altro vettore dell'insieme $\mathcal{L} \cap Z$, le proiezioni sull'insieme di indici S sono diverse. Per costruzione dell'Hitting set si ha anche $(\tilde{f}(a_1, \ldots, a_{i-1}, j, u)_{j \in \mathbb{F}}) \in \mathcal{L} \cap Z$ e quindi le verifiche ci assicurano che $v_{a_i} = \tilde{f}(a_1, \ldots, a_i, u)$.
- Ora stimiamo la dimensione del circuito per $\tilde{f}$ che può essere costruito dalla procedura. Per computare $\tilde{f}(a_1, \ldots, a_k)$, la procedura viene chiamata k volte. Per ognuna di tali chiamate, la complessità della procedura è limitata dal tempo richiesto per fare meno di $|S|^{k-1}$ verifiche di un v valore. Ciascuna di queste verifiche richiede il tempo di valutare un circuito di dimensione n^q, il tempo richiesto per controllare che una tabella di dimensione n è un polinomio di grado basso (che è limitato da n^2) e confrontare $|S|$ valori in $\mathbb{F}$ (che è limitato da n). Quindi, inserendo l'avviso nel circuito, possiamo convertire tutta la procedura in un circuito SV-nondeterministico. La dimensione del circuito è limitata dall'alto da $O\left(\left(n^\delta\right)^k n^q\right)$.

Scegliendo δ e k tali che $O\left(\left(n^\delta\right)^k n^q\right) < 2^{\varepsilon kl}$ otteniamo un assurdo. □

5 Il problema dei lower bound sulla complessità circuitale

Il risultato che abbiamo mostrato, come molti altri simili nel campo, hanno come premessa congetture sull'esistenza di funzioni in certe classi di complessità che siano difficili da computare per circuiti di un certo tipo. Dimostrare lower bound sulla complessità circuitale è un problema aperto da tempo sul quale si stanno facendo progressi solo per particolari tipi di circuiti (ad es. monotoni).

In particolare, il miglior bound attuale per funzioni in **NP** è che necessitano di circuiti di più di $\mathbf{5}n$ porte. Si hanno invece bound polinomiali e superpolinomiali per funzioni nella gerarchia polinomiale.

Si sospetta che tali bound esistano anche in virtù del fatto che la maggior parte delle funzioni ha bisogno di circuiti di taglia esponenziale. L'approccio di Razborov-Rudich consente di delineare molteplici approcci dimostrativi che **NON** possono funzionare.

Riferimenti bibliografici

Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

László Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

László Babai, Lance Fortnow, Leonid A Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32. ACM, 1991.

Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68. ACM, 1986.

Pascal Koiran. Hilbert's nullstellensatz is in the polynomial hierarchy. *Journal of complexity*, 12(4): 273–286, 1996.

Peter Bro Miltersen and N Variyam Vinodchandran. Derandomizing arthur–merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

Michael Saks, Aravind Srinivasan, and Shiyu Zhou. Explicit or-dispersers with polylogarithmic degree. *Journal of the ACM (JACM)*, 45(1):123–154, 1998.