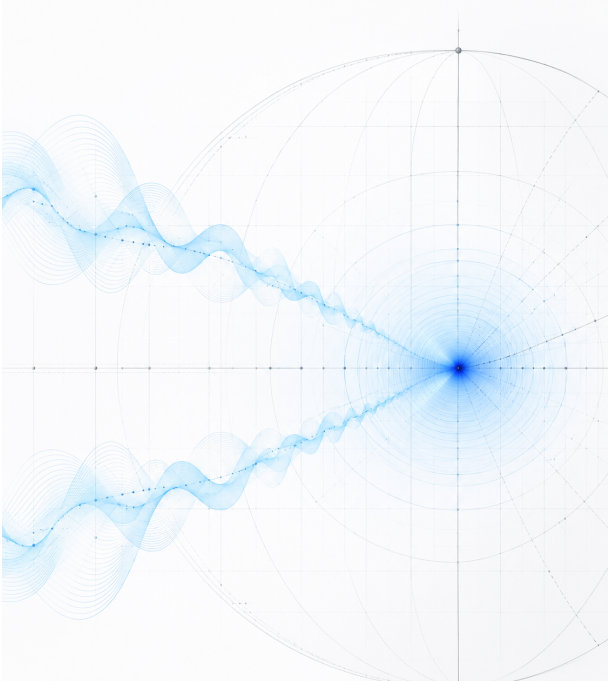
LEZIONE 17

Promise problems e modello a oracle

Una proprietà globale, poche interrogazioni

Metodi della Fisica Quantistica per l'Ingegneria

Prof. Vincenzo Berardi · Politecnico di Bari



Che cosa vogliamo determinare?

Non sempre serve ricostruire tutti i dati di un problema. Può bastare stabilire se possiedono una certa **proprietà globale**.

0	0	0	0	1	1	1	1

Domanda: i valori sono tutti uguali, oppure metà vale 0 e metà vale 1? Ci interessa *classificare* la configurazione, non necessariamente conoscere l'intera stringa.

Problema ordinario e promise problem

Problema ordinario: l'algoritmo deve rispondere correttamente per tutti gli input ammessi. **Promise problem:** è garantito a priori che l'input appartenga a una classe specificata di casi. La *promise* è una **ipotesi del problema**; non è una condizione che l'algoritmo debba verificare.

La promessa esclude configurazioni che, altrimenti, sarebbero possibili.

Il ruolo della promessa

Una condizione data in partenza può rendere sufficienti meno osservazioni. **Analogia con la fisica.** Se il testo di un esercizio assume l'assenza di attrito, questa ipotesi può essere usata nei calcoli: non si chiede di dimostrarla. Allo stesso modo, in un promise problem, gli input non compatibili con la promise sono **esclusi per definizione**.

La promessa cambia quali informazioni occorre acquisire.

Tre esempi dello stesso problema

1. **Sequenza di bit:** tutti uguali oppure tanti 0 quanti 1.
2. **Lotto di produzione:** tutti i componenti di un tipo oppure metà di tipo A e metà B.
3. **Rete di sensori:** tutti restituiscono lo stesso valore oppure esattamente metà restituisce 0 e metà 1.

In tutti i casi si cerca **una proprietà della configurazione**, non un elenco completo dei valori.

Otto dispositivi: la promise

Consideriamo otto dispositivi, numerati da 0 a 7. Ciascuno può trovarsi nello stato 0 oppure 1. Il problema garantisce **una e una sola** delle condizioni seguenti:

- **Uniforme:** tutti e otto i valori sono uguali.
- **Bilanciato:** quattro valori sono 0 e quattro sono 1.

Le configurazioni con sei zeri e due uni, ad esempio, **non sono ammesse**.

Obiettivo: distinguere uniforme da bilanciato, controllando un dispositivo alla volta.

Due configurazioni ammesse

Uniforme:

0	0	0	0	0	0	0	0

Bilanciata:

0	0	0	0	1	1	1	1

L'ordine dei dispositivi bilanciati può cambiare. La promise specifica **quanti** 0 e 1 sono presenti, non dove si trovano.

Quattro controlli non bastano

Supponiamo che i primi quattro dispositivi controllati restituiscano

$$0, \quad 0, \quad 0, \quad 0.$$

Queste osservazioni sono compatibili sia con

$$\underbrace{0000}_{\text{osservati}} \; 0000 \qquad \text{(uniforme)}$$

sia con

$$\underbrace{0000}_{\text{osservati}} \; 1111 \qquad \text{(bilanciato)}.$$

Con quattro controlli, la risposta è ancora indeterminata.

Il quinto controllo: caso 1

I primi quattro controlli hanno dato 0000. Il quinto restituisce **1**. Abbiamo osservato entrambi i valori:

$$0,\ 0,\ 0,\ 0,\ \boxed{1}.$$

Una configurazione uniforme non può contenere contemporaneamente 0 e 1. Poiché la promise ammette solo uniforme o bilanciato, la risposta è necessariamente

$$\boxed{\text{BILANCIATO}}.$$

Il quinto controllo: caso 2

I primi quattro controlli hanno dato 0000. Anche il quinto restituisce **0**. Abbiamo osservato cinque zeri:

$$0,\ 0,\ 0,\ 0,\ \boxed{0}.$$

Una configurazione bilanciata può contenere *al massimo quattro zeri in totale*. La configurazione è dunque necessariamente

$$\boxed{\text{UNIFORME}}.$$

Cinque interrogazioni nel caso peggiore

Se nei primi controlli compaiono sia 0 sia 1, possiamo concludere subito: la configurazione è bilanciata. Se invece i primi quattro valori coincidono, occorre un quinto controllo; esso permette di decidere in entrambi i casi.

$$\boxed{Q_{\text{classico}}^{\text{worst}} = 5 \quad \text{(otto dispositivi)}}$$

Perché non ne bastano quattro nel caso peggiore? Perché dopo quattro risultati uguali restano possibili sia una configurazione uniforme sia una bilanciata.

Che cosa succede senza la promise?

Supponiamo di aver osservato cinque zeri:

$$0,\ 0,\ 0,\ 0,\ 0,\ ?,\ ?,\ ?.$$

Con la promise: sappiamo che il sistema è uniforme. **Senza la promise:** gli ultimi tre dispositivi potrebbero avere valori diversi; i dati osservati non determinano la configurazione completa.

Il risparmio di interrogazioni dipende dall'ipotesi iniziale.

Dai dispositivi alla funzione

Associamo a ogni dispositivo un indice $x \in \{0, 1, \ldots, 7\}$ e definiamo

$$f(x) \in \{0, 1\}.$$

La **funzione costante** restituisce lo stesso valore per tutti gli input:

$$f(x) = 0 \quad \forall x \qquad \text{oppure} \qquad f(x) = 1 \quad \forall x.$$

La **funzione bilanciata** restituisce 0 su quattro input e 1 sugli altri quattro. Il problema è riconoscere *a quale classe appartiene* f.

Il problema in forma generale

Sia

$$f : \{0,1\}^n \longrightarrow \{0,1\}.$$

Esistono 2^n input possibili. Assumiamo che valga una delle due condizioni:

- f è **costante**: tutti i valori sono 0 oppure tutti sono 1;
- f è **bilanciata**: 2^{n-1} valori sono 0 e 2^{n-1} sono 1.

Obiettivo: stabilire quale delle due condizioni vale, senza costruire tutta la tabella di f.
Per $n = 3$ si ritrova l'esempio degli otto dispositivi.

Un limite classico nel caso peggiore

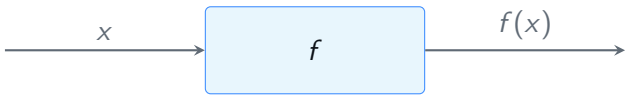
Con 2^n input, la promessa di bilanciamento ammette al massimo 2^{n-1} risultati uguali. Se i primi 2^{n-1} valori interrogati coincidono, non possiamo ancora distinguere una funzione costante da una bilanciata. L'interrogazione successiva decide:

$$\boxed{Q_{\text{classico}}^{\text{worst}} = 2^{n-1} + 1}$$

Il conteggio riguarda interrogazioni deterministiche all'oracle; non è il numero complessivo di porte del circuito.

Una query è un'interrogazione alla funzione

Nel modello a **oracle** l'algoritmo può scegliere un ingresso x e ottenere $f(x)$, senza conoscere l'implementazione interna della funzione.



Ogni interrogazione è una **query**: ad esempio, conoscere $f(3) = 0$ significa aver interrogato una sola volta la funzione, senza conoscere gli altri valori.

Qui si contano le query necessarie per distinguere una funzione costante da una bilanciata, non le porte impiegate per realizzarla.

Dall'*oracle* classico all'*oracle* quantistico

Per una funzione $f : \{0,1\}^n \to \{0,1\}$, l'*oracle* quantistico agisce su due registri secondo la regola

$$\boxed{U_f \left|x\right\rangle \left|y\right\rangle = \left|x\right\rangle \left|y \oplus f(x)\right\rangle .}$$

Se $f(x) = 0$ il secondo bit resta invariato; se $f(x) = 1$ viene invertito. Prepariamo il primo registro nello stato $\left|\phi\right\rangle = \sum_x \alpha_x \left|x\right\rangle$ e il secondo nello stato $\left|y\right\rangle$. Per distributività:

$$\left|\psi\right\rangle = \left|\phi\right\rangle \otimes \left|y\right\rangle = \Big(\sum_x \alpha_x \left|x\right\rangle \Big) \otimes \left|y\right\rangle = \sum_x \alpha_x \left|x\right\rangle \left|y\right\rangle .$$

Per **linearità** dell'operatore U_f e per la sua definizione,

$$\begin{aligned} U_f \left|\psi\right\rangle &= U_f \Big(\sum_x \alpha_x \left|x\right\rangle \left|y\right\rangle \Big) = \sum_x \alpha_x U_f(\left|x\right\rangle \left|y\right\rangle) \\ &= \boxed{\sum_x \alpha_x \left|x\right\rangle \left|y \oplus f(x)\right\rangle} . \end{aligned}$$

Una query su quattro ingressi: esempio

Scegliamo $n = 2$ e prepariamo quattro ingressi in sovrapposizione uniforme, con il secondo registro nello stato $|0\rangle$:

$$|\psi\rangle = \tfrac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)\,|0\rangle\,.$$

Supponiamo $f(00) = 0$, $f(01) = 1$, $f(10) = 1$, $f(11) = 0$. La definizione dell'*oracle* dà, su ciascun termine,

$$U_f\,|00\rangle\,|0\rangle = |00\rangle\,|0\rangle\,, \quad U_f\,|01\rangle\,|0\rangle = |01\rangle\,|1\rangle\,,$$
$$U_f\,|10\rangle\,|0\rangle = |10\rangle\,|1\rangle\,, \quad U_f\,|11\rangle\,|0\rangle = |11\rangle\,|0\rangle\,.$$

Quindi, per linearità,

$$\boxed{U_f\,|\psi\rangle = \tfrac{1}{2}\big(\,|00\rangle\,|0\rangle + |01\rangle\,|1\rangle + |10\rangle\,|1\rangle + |11\rangle\,|0\rangle\,\big).}$$

I quattro calcoli descrivono *un'unica applicazione* di U_f alla sovrapposizione, non quattro query distinte.

La sovrapposizione non rivela tutta la tabella

L'azione di U_f su una sovrapposizione non rende simultaneamente **leggibili** tutti i valori di $f(x)$. Il problema non è interrogare più ingressi *e poi leggerli tutti*, ma **progettare una sequenza di operazioni quantistiche** che, attraverso l'interferenza delle ampiezze, permetta di determinare la proprietà globale cercata.

1. Prepariamo una sovrapposizione degli ingressi.
2. Applichiamo l'*oracle* U_f.
3. Trasformiamo le ampiezze e misuriamo un'informazione utile.

La complessità in *query* conta le chiamate all'*oracle*, non il numero totale di porte del circuito.

Dov'è il vantaggio quantistico?

Promessa: $f : \{0,1\}^n \to \{0,1\}$ è costante oppure bilanciata; vogliamo distinguerle *con certezza*.

$$\boxed{\begin{aligned} Q_{\mathrm{C}}^{\mathrm{det,\,worst}}(n) &= 2^{n-1}+1, \\ Q_{\mathrm{Q}}^{\mathrm{esatto}}(n) &= 1. \end{aligned}}$$

Anticipazione: nelle prossime lezioni dimostreremo che una sola query quantistica basta a distinguere i due casi con certezza.

Lect18 — Deutsch ($n = 1$): 2 query classiche contro 1 quantistica.

Lect19 — Deutsch–Jozsa ($n \geq 2$): per $n = 3$ (8 ingressi), 5 query classiche contro 1 quantistica.

Il vantaggio riguarda le interrogazioni all'*oracle*, non automaticamente il tempo fisico totale.

Il confronto esponenziale è con algoritmi classici deterministici ed esatti; con errore limitato anche metodi classici probabilistici usano un numero costante di query.