UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea
in Ingegneria Informatica

Relazione finale

Sviluppo di Algoritmi Quantistici: Dal Modello Circuitale a Qiskit

Relatore
Chiar.mo Prof. Luca Giuzzi

Laureando
Gabriele Rossini
Matricola n. 712391

Anno Accademico 2024/2025

Indice

Introduzione

L'informatica quantistica non rappresenta una mera evoluzione della capacità di calcolo classica, bensì un fondamentale cambio di paradigma nel trattamento dell'informazione. Mentre l'industria dei semiconduttori si avvicina ai limiti fisici della Legge di Moore, il Quantum Computing offre un modello computazionale alternativo che sfrutta i principi della meccanica quantistica — sovrapposizione ed entanglement — per risolvere classi di problemi intrattabili per le architetture di von Neumann tradizionali.

Per un ingegnere del software, l'adozione di questo paradigma impone però un ripensamento radicale delle primitive di programmazione. Se il bit classico è deterministico e duplicabile, il qubit è al contrario un'entità probabilistica descritta da vettori in spazi di Hilbert complessi, soggetta a vincoli fisici stringenti come il Teorema di No-Cloning e il collasso della funzione d'onda all'atto della misura. Questi vincoli trasformano la progettazione di algoritmi, il debugging e la gestione della memoria in sfide ingegneristiche inedite, dove la reversibilità logica e la gestione del rumore (nell'attuale era NISQ - *Noisy Intermediate-Scale Quantum*) diventano requisiti funzionali primari.

Il presente lavoro di tesi si propone di analizzare il calcolo quantistico attraverso la lente dell'ingegneria del software, astraendo dalla fisica sottostante per concentrarsi sul modello circuitale come standard per la definizione degli algoritmi. L'obiettivo è duplice: fornire una formalizzazione matematica rigorosa delle porte logiche e degli stati quantistici necessaria per la comprensione del sistema, e implementare tali concetti utilizzando moderni framework di sviluppo come Qiskit.

Capitolo 1

Fondamenti del Quantum Computing

Il paradigma del Quantum Computing impone un cambiamento radicale nel modo di modellare e manipolare l'informazione. Mentre l'informatica classica si fonda sull'algebra booleana e su stati discreti deterministici, l'informatica quantistica opera in spazi vettoriali complessi, governati dalle leggi dell'algebra lineare e della meccanica quantistica [6].

Questo passaggio implica l'abbandono delle certezze binarie in favore di un modello probabilistico e continuo. Prima di poter progettare algoritmi o utilizzare framework di sviluppo ad alto livello come Qiskit, è necessario padroneggiare il formalismo matematico che descrive lo stato del sistema e le sue trasformazioni, poiché ogni istruzione in un linguaggio quantistico corrisponde a una precisa operazione algebrica su vettori e matrici [2].

In questo capitolo definiremo le primitive fondamentali del calcolo quantistico. Inizieremo definendo gli Spazi di Hilbert e la notazione di Dirac per descrivere gli stati, per poi analizzare le proprietà del qubit e la sua rappresentazione sulla Sfera di Bloch. Il capitolo si conclude con la discussione degli operatori unitari che costituiscono le porte logiche del nostro modello computazionale.

1.1 Preliminari Matematici

Il calcolo quantistico è descritto rigorosamente nel linguaggio dell'algebra lineare e dell'analisi funzionale. A differenza del calcolo classico, che opera in spazi discreti booleani $\{0,1\}^n$, il calcolo quantistico opera in spazi vettoriali continui sui numeri complessi $\mathbb{C}$. Comprendere questa astrazione è prerequisito

necessario per modellare lo stato del sistema, le trasformazioni (porte logiche) e le misurazioni.

1.1.1 Numeri Complessi e Spazi di Hilbert

Il modello matematico formale della meccanica quantistica si basa sugli spazi di Hilbert [6]. Prima di definirli, richiamiamo alcune proprietà essenziali dei numeri complessi. Un numero complesso $z \in \mathbb{C}$ è definito come $z = x + iy$, dove $x, y \in \mathbb{R}$ e $i = \sqrt{-1}$ è l'unità immaginaria.

Definiamo il **coniugio complesso** di z, denotato come z^*, l'operazione che inverte il segno della parte immaginaria:

$$z^* = x - iy \tag{1.1}$$

Questa operazione è fondamentale nel calcolo delle probabilità quantistiche. La norma (o modulo) al quadrato di un numero complesso è definita come il prodotto del numero per il suo coniugato:

$$|z|^2 = zz^* = (x + iy)(x - iy) = x^2 + y^2 \tag{1.2}$$

Tale quantità è sempre un numero reale non negativo [2].

Uno **Spazio di Hilbert** $\mathcal{H}$ è definito come uno spazio vettoriale completo sui numeri complessi $\mathbb{C}$, dotato di un prodotto interno. La proprietà di **completezza** è cruciale per l'analisi matematica: essa garantisce che ogni successione di Cauchy di vettori in $\mathcal{H}$ converga a un limite che appartiene ancora allo spazio $\mathcal{H}$ [10]. Formalmente, una successione $\{v_n\}$ è di Cauchy se la distanza tra i vettori $||v_n - v_m||$ tende a zero al crescere di n e m. La completezza assicura che i limiti delle sequenze iterative (spesso utilizzate negli algoritmi variazionali o nelle simulazioni numeriche) siano stati quantistici ben definiti all'interno dello spazio di lavoro [2].

Nel contesto del quantum computing a dimensione finita, ci limitiamo a spazi di dimensione 2^n (dove n è il numero di qubit), isomorfi a $\mathbb{C}^{2^n}$ [5].

1.1.2 Notazione di Dirac e Prodotto Interno

La notazione standard utilizzata nella letteratura e nelle librerie software come Qiskit è la notazione Bra-Ket di Dirac. Essa semplifica la manipolazione di vettori e operatori lineari.

- **Ket** ($|\psi\rangle$): Rappresenta un vettore colonna nello spazio di Hilbert. È l'analogo quantistico di un vettore di stato $\boldsymbol{v}$.

$$|\psi\rangle = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \tag{1.3}$$

- **Bra** ($\langle\psi|$): Rappresenta il vettore duale del Ket nello spazio duale $\mathcal{H}^*$. Esso corrisponde al trasposto coniugato (o aggiunto hermitiano) del vettore colonna:

$$\langle\psi| = (|\psi\rangle)^\dagger = \begin{pmatrix} c_1^* & c_2^* & \dots & c_n^* \end{pmatrix} \tag{1.4}$$

  dove il simbolo † (dagger) denota l'operazione di trasposizione coniugata [11].

L'operazione fondamentale che dota lo spazio di Hilbert della sua struttura geometrica è il **prodotto interno** (o scalare), denotato come $\langle\phi|\psi\rangle$. In meccanica quantistica, questo è definito come il **prodotto Hermitiano standard**. Esso è una forma *sesquilineare*, ovvero:

- Lineare nel secondo argomento (Ket): $\langle\phi|a\psi_1 + b\psi_2\rangle = a\langle\phi|\psi_1\rangle + b\langle\phi|\psi_2\rangle$
- Antilineare nel primo argomento (Bra): $\langle a\phi_1 + b\phi_2|\psi\rangle = a^*\langle\phi_1|\psi\rangle + b^*\langle\phi_2|\psi\rangle$

dove $a, b \in \mathbb{C}$ sono scalari complessi [6].

Il prodotto interno è una forma **definita positiva**: il prodotto scalare di un vettore con se stesso, $\langle\psi|\psi\rangle$, è sempre un numero reale non negativo e si annulla se e solo se $|\psi\rangle$ è il vettore nullo [10]. Questa proprietà permette di definire la **norma** euclidea del vettore di stato come:

$$||\psi|| = \sqrt{\langle\psi|\psi\rangle} \tag{1.5}$$

Nel software quantistico, la norma assume un significato fisico immediato: i vettori di stato che rappresentano stati fisici validi devono essere normalizzati a 1 ($||\psi|| = 1$), condizione necessaria affinché la somma delle probabilità di tutti i possibili esiti di misura sia pari al 100% (interpretazione probabilistica o Regola di Born) [2].

Due vettori sono detti *ortogonali* se il loro prodotto interno è nullo: $\langle\phi|\psi\rangle = 0$. I vettori della base computazionale, $|0\rangle$ e $|1\rangle$, formano una base ortonormale per lo spazio $\mathbb{C}^2$.

1.1.3 Operatori Lineari e Matrici

L'evoluzione di un sistema quantistico chiuso è descritta da operatori lineari. Un operatore lineare A mappa vettori in vettori: $A : \mathcal{H} \to \mathcal{H}$. In uno spazio a dimensione finita, questi operatori sono rappresentati da matrici $N \times N$ [11].

Due classi di operatori sono di particolare interesse ingegneristico:

- **Operatori Unitari (U)**: Rappresentano le porte logiche quantistiche reversibili. Soddisfano la condizione $U^\dagger U = I$. Preservano la norma del vettore di stato [1].
- **Operatori Hermitiani (H)**: Rappresentano le osservabili fisiche. Soddisfano la condizione $H = H^\dagger$. I loro autovalori sono reali e corrispondono ai possibili risultati di una misurazione [4].

1.1.4 Prodotto Tensoriale e Sistemi Composti

La capacità di descrivere sistemi composti da più sottosistemi (es. un registro di n qubit) è garantita dall'operazione di **prodotto tensoriale**. Siano $\mathcal{H}_A$ e $\mathcal{H}_B$ due spazi di Hilbert di dimensione n e m rispettivamente. Lo spazio degli stati del sistema composto è definito come il prodotto tensoriale degli spazi componenti, denotato come $\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$ [6].

Il prodotto tensoriale è un'operazione astratta che combina due vettori $|\psi\rangle \in \mathcal{H}_A$ e $|\phi\rangle \in \mathcal{H}_B$ in un nuovo vettore $|\psi\rangle \otimes |\phi\rangle \in \mathcal{H}_{AB}$. Tale operazione deve soddisfare la proprietà di **bilinearità**. Per ogni scalare $z \in \mathbb{C}$ e vettori $|v_1\rangle, |v_2\rangle \in \mathcal{H}_A$, $|w\rangle \in \mathcal{H}_B$, valgono le seguenti relazioni [10]:

1. $z(|v_1\rangle \otimes |w\rangle) = (z\,|v_1\rangle) \otimes |w\rangle = |v_1\rangle \otimes (z\,|w\rangle)$
2. $(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle$
3. $|v_1\rangle \otimes (|w_1\rangle + |w_2\rangle) = |v_1\rangle \otimes |w_1\rangle + |v_1\rangle \otimes |w_2\rangle$

La dimensione dello spazio risultante è il prodotto delle dimensioni degli spazi originali: $\dim(\mathcal{H}_{AB}) = n \times m$. Una base per $\mathcal{H}_{AB}$ è formata dal prodotto tensoriale dei vettori delle basi dei sottosistemi. Se $\{|e_i\rangle\}$ è una base per $\mathcal{H}_A$ e $\{|f_j\rangle\}$ per $\mathcal{H}_B$, allora $\{|e_i\rangle \otimes |f_j\rangle\}$ è una base per lo spazio composto [10]. Spesso si utilizza la notazione abbreviata $|\psi\rangle\,|\phi\rangle$ o $|\psi\phi\rangle$ per indicare $|\psi\rangle \otimes |\phi\rangle$.

Rappresentazione Matriciale: Il Prodotto di Kronecker

Dal punto di vista implementativo (ad esempio nella simulazione di circuiti quantistici su hardware classico), è necessario adottare una rappresentazione concreta del prodotto tensoriale. Quando gli operatori e i vettori sono espressi rispetto a una base ordinata specifica (come la base computazionale), il prodotto tensoriale corrisponde al **prodotto di Kronecker** matriciale [2, 1].

Dati due vettori colonna che rappresentano gli stati $|\psi\rangle = [\psi_0, \psi_1]^T$ e $|\phi\rangle = [\phi_0, \phi_1]^T$, la rappresentazione del loro prodotto tensoriale è:

$$|\psi\rangle \otimes |\phi\rangle \equiv \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix} \otimes \begin{pmatrix} \phi_0 \\ \phi_1 \end{pmatrix} = \begin{pmatrix} \psi_0 \begin{pmatrix} \phi_0 \\ \phi_1 \end{pmatrix} \\ \psi_1 \begin{pmatrix} \phi_0 \\ \phi_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \psi_0\phi_0 \\ \psi_0\phi_1 \\ \psi_1\phi_0 \\ \psi_1\phi_1 \end{pmatrix} \tag{1.6}$$

È cruciale mantenere la distinzione concettuale: mentre il prodotto tensoriale è l'operazione algebrica che definisce la struttura dello spazio di Hilbert composto, il prodotto di Kronecker è l'algoritmo utilizzato per calcolare le componenti del vettore risultante in una base fissata. Questa operazione evidenzia la crescita esponenziale della memoria richiesta per simulare sistemi quantistici: l'aggiunta di ogni qubit raddoppia la dimensione del vettore di stato [6].

1.2 Il Qubit e l'Informazione Quantistica

Il *bit* classico è l'unità fondamentale dell'informazione nell'informatica tradizionale; esso può assumere uno dei due stati mutuamente esclusivi, rappresentati dai valori logici 0 e 1. In termini fisici, questi stati corrispondono a livelli di tensione distinti in un transistor o direzioni di magnetizzazione su un disco rigido.

Il **qubit** (quantum bit) è l'analogo quantistico del bit classico, ma con una differenza sostanziale: mentre il bit classico è limitato a uno spazio discreto $\{0, 1\}$, il qubit è descritto da un vettore di stato unitario in uno spazio di Hilbert complesso bidimensionale $\mathbb{C}^2$ [6].

Si definisce **base computazionale** l'insieme dei vettori ortonormali standard $\{|0\rangle, |1\rangle\}$, rappresentati vettorialmente come:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1.7}$$

Questa scelta è convenzionale e associa il vettore $|0\rangle$ al primo elemento della base standard e $|1\rangle$ al secondo [2, 5]. Utilizzando questa base, lo stato di un qubit $|\psi\rangle$ è descritto dalla sovrapposizione lineare:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{1.8}$$

dove $\alpha, \beta \in \mathbb{C}$ sono le *ampiezze di probabilità*.

1.2.1 Differenze col Modello Classico e Normalizzazione

La differenza fondamentale tra bit e qubit risiede nella natura dei coefficienti che ne descrivono lo stato. Mentre lo stato di un bit classico è descritto da un singolo valore binario, la descrizione esatta di un qubit richiede due numeri complessi continui α e β. Tuttavia, il sistema è vincolato fisicamente dalla **condizione di normalizzazione**:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{1.9}$$

Questo vincolo non è arbitrario, ma deriva dall'interpretazione probabilistica della meccanica quantistica (Regola di Born). Sebbene il concetto di misura verrà trattato formalmente nelle sezioni successive, è necessario anticiparne qui l'intuizione fisica per comprendere il significato di α e β.

Quando un qubit viene misurato, il risultato non è mai una sovrapposizione, ma è *sempre* uno stato unico e determinato della base computazionale: o si osserva $|0\rangle$ o si osserva $|1\rangle$. Le ampiezze determinano la probabilità di questo esito:

- La probabilità di misurare lo stato $|0\rangle$ è pari a $|\alpha|^2$.
- La probabilità di misurare lo stato $|1\rangle$ è pari a $|\beta|^2$.

Di conseguenza, l'equazione esprime la certezza che, effettuando una misurazione, si otterrà uno dei due possibili risultati con probabilità totale pari al 100% [2, 6].

È necessario operare una distinzione fondamentale tra la rappresentazione dello stato e l'informazione accessibile. La descrizione matematica esatta di un qubit, basata sulle ampiezze α e β, richiede due numeri complessi continui e potrebbe richiedere una precisione arbitrariamente elevata per essere specificata completamente [5]. Tuttavia, ciò non implica una capacità di memorizzazione

infinita accessibile: i teoremi fondamentali dell'informazione quantistica stabiliscono che la quantità di informazione classica estraibile da un singolo qubit non può superare un bit. La complessità del sistema risiede dunque nella continuità dello spazio degli stati, non nella quantità di dati recuperabili con una singola misurazione [11, 6].

1.2.2 Rappresentazione Geometrica: La Sfera di Bloch

Un generico vettore di stato normalizzato in $\mathbb{C}^2$ è descritto da quattro parametri reali. Tuttavia, l'equazione può essere riscritta raccogliendo un termine di fase comune:

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right) \tag{1.10}$$

dove $\gamma, \theta, \phi \in \mathbb{R}$. Il termine $e^{i\gamma}$ è denominato **fattore di fase globale**. Dal punto di vista della misurazione di un singolo qubit isolato, questo fattore è fisicamente irrilevante. Infatti, per il Postulato della Misura, la probabilità di osservare uno stato x è data dal modulo quadro dell'ampiezza: $|e^{i\gamma}c_x|^2 = |e^{i\gamma}|^2|c_x|^2 = 1 \cdot |c_x|^2$. Poiché il modulo di un esponenziale immaginario è unitario, la fase globale scompare e non influenza le statistiche di misura [6].

Tuttavia, è fondamentale operare una distinzione netta tra fase globale e **fase relativa** ϕ:

- La **fase globale** ($e^{i\gamma}$) è un artefatto della rappresentazione vettoriale di un sistema isolato e non incide sulle osservabili.
- La **fase relativa** ($e^{i\phi}$), invece, è parte integrante dello stato di sovrapposizione. Essa determina come le ampiezze si sommano (interferenza costruttiva o distruttiva) quando al qubit vengono applicate porte logiche successive (come la porta Hadamard) [2].

Inoltre, in sistemi a più qubit, quella che appare come una fase globale su un sottosistema può trasformarsi in una fase relativa (osservabile) sul sistema complessivo attraverso meccanismi di controllo come il *Phase Kickback*, sfruttato in algoritmi come la stima della fase o l'algoritmo di Deutsch-Jozsa [3].

Per la rappresentazione geometrica di un singolo qubit isolato, possiamo fissare arbitrariamente $\gamma = 0$, ottenendo la forma canonica:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{1.11}$$

Questa parametrizzazione permette di mappare biunivocamente lo stato su una sfera unitaria in $\mathbb{R}^3$, nota come **Sfera di Bloch**.

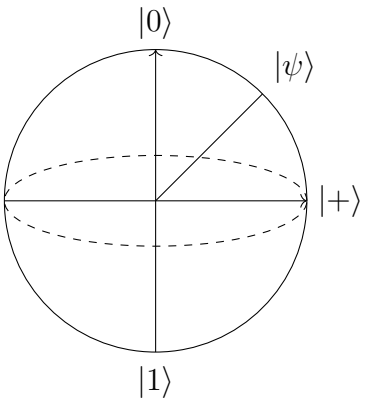

Figura 1.1: Rappresentazione bidimensionale semplificata della Sfera di Bloch. Il polo nord rappresenta lo stato $|0\rangle$, il polo sud lo stato $|1\rangle$.

In questa rappresentazione geometrica:

- Il polo nord corrisponde allo stato $|0\rangle$.
- Il polo sud corrisponde allo stato $|1\rangle$.
- L'equatore rappresenta gli stati di sovrapposizione equamente pesata (es. $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$).

Le operazioni logiche a singolo qubit (single-qubit gates) possono essere visualizzate come rotazioni del vettore di stato attorno agli assi di questa sfera [4]. Nel formalismo del quantum computing, le operazioni che trasformano lo stato di un qubit e le operazioni di misurazione appartengono a due classi distinte di operatori lineari:

1. **Operatori Unitari (Gate):** L'evoluzione temporale di un qubit chiuso (non soggetto a misurazione) è descritta da operatori unitari U tali che $U^\dagger U = I$. Questi operatori preservano la norma del vettore di stato, garantendo che la somma delle probabilità rimanga 1. Essi corrispondono alle porte logiche reversibili del circuito quantistico [10].

2. **Operatori Hermitiani (Osservabili):** Le quantità fisiche misurabili sono associate a operatori Hermitiani H tali che $H = H^\dagger$. Gli autovalori di questi operatori sono reali e rappresentano i possibili risultati della misurazione. Per un qubit, l'osservabile standard è l'operatore Z (matrice di Pauli σ_z), i cui autovalori $+1$ e -1 sono associati agli stati di base $|0\rangle$ e $|1\rangle$ [5].

1.2.3 Il Principio di Superposizione

La superposizione è il principio secondo cui un sistema quantistico può esistere in una combinazione lineare di più stati base contemporaneamente. Per un registro di n qubit, la base computazionale contiene 2^n stati. Grazie alla superposizione, un computer quantistico può elaborare simultaneamente un vettore di stato che rappresenta una combinazione di tutti i 2^n input possibili. Mermin nota come sia errato pensare che il qubit possieda *sia* il valore 0 *sia* il valore 1 nello stesso modo in cui un oggetto classico possiede proprietà. La superposizione è una descrizione intrinseca e irriducibile dello stato prima della misurazione [5].

1.3 Porte Logiche Quantistiche

Nel modello circuitale del quantum computing, un algoritmo è espresso come una sequenza di porte logiche elementari che manipolano lo stato dei qubit. Tuttavia, a differenza della logica booleana classica dove la scelta delle porte è flessibile, la natura fisica dei sistemi quantistici impone vincoli rigidi sulla classe di operatori ammissibili.

Poiché l'evoluzione temporale di un sistema quantistico chiuso deve conservare la probabilità totale degli stati (la norma del vettore di stato deve rimanere unitaria, $\langle\psi|\psi\rangle = 1$ in ogni istante), gli operatori lineari U che descrivono le porte logiche devono necessariamente essere **unitari** [6]. La condizione di unitarietà non è dunque una scelta architetturale, ma una necessità fisica definita dalla relazione:

$$U^\dagger U = I \tag{1.12}$$

dove $U^\dagger$ è l'aggiunto hermitiano (trasposta coniugata) e I è la matrice identità.

Questa necessità comporta una conseguenza fondamentale: ogni porta logica quantistica deve essere **logicamente reversibile**. Infatti, dalla definizione di matrice unitaria segue che l'inversa esiste sempre ed è pari all'aggiunto ($U^{-1} = U^\dagger$). Questo marca una profonda differenza rispetto all'elettronica digitale classica, che si basa su porte dissipative e irreversibili come la NAND (dato l'output di una NAND, non è possibile ricostruire univocamente gli input). In un circuito quantistico, invece, non è possibile cancellare informazione a livello di gate unitari senza violare le leggi della meccanica quantistica; l'informazione deve essere sempre preservata e recuperabile invertendo il circuito [5, 2].

1.3.1 Porte a singolo Qubit

Le porte a singolo qubit sono rappresentate da matrici unitarie 2×2. Esse corrispondono a rotazioni del vettore di stato sulla sfera di Bloch.

Porte di Pauli (X, Y, Z)

Le porte di Pauli sono fondamentali per la manipolazione dei singoli bit quantistici e corrispondono alle matrici di Pauli introdotte in fisica [2]:

- **Porta X (Bit-flip):** È l'equivalente quantistico della porta NOT classica. Scambia le ampiezze degli stati $|0\rangle$ e $|1\rangle$.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle \tag{1.13}$$

- **Porta Z (Phase-flip):** Lascia inalterato lo stato $|0\rangle$ ma inverte il segno (fase relativa) dello stato $|1\rangle$. Non ha un analogo classico diretto.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle \tag{1.14}$$

- **Porta Y:** Combina l'effetto di bit-flip e phase-flip con un fattore di fase immaginaria i.

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{1.15}$$

Porta di Hadamard (H)

La porta di Hadamard è una delle operazioni più utilizzate negli algoritmi quantistici poiché permette di creare stati di sovrapposizione bilanciata a partire da stati della base computazionale [6]. La sua matrice è:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.16}$$

La sua applicazione trasforma gli stati di base in:

$$H|0\rangle = |+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad H|1\rangle = |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{1.17}$$

In termini geometrici, H rappresenta una rotazione di 90° attorno all'asse Y seguita da una rotazione di 180° attorno all'asse X della sfera di Bloch.

Porte di Fase (S, T)

Le porte di fase introducono una rotazione relativa attorno all'asse Z della sfera di Bloch, modificando la fase dello stato $|1\rangle$ senza alterarne le probabilità di misurazione (poiché $|e^{i\theta}|^2 = 1$). Queste porte sono essenziali per definire la granularità delle rotazioni possibili in un hardware discreto.

- **Porta S** (Phase Gate): Introduce una fase di $\pi/2$ (90°). La sua matrice è:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{1.18}$$

  Si noti che $S = T^2$ e $S^2 = Z$. Questo gate fa parte del cosiddetto *Gruppo di Clifford*, un insieme di operazioni che, sebbene fondamentale, non è sufficiente per l'universalità (teorema di Gottesman-Knill) [2].

- **Porta T** ($\pi/8$ Gate): Introduce una fase di $\pi/4$ (45°).

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \tag{1.19}$$

  Storicamente denominata porta $\pi/8$ (a meno di un fattore di fase globale irrilevante), la porta T riveste un ruolo cruciale nell'architettura dei computer quantistici fault-tolerant.

  È necessario operare una distinzione tra universalità esatta e approssimata. Un set di porte contenente rotazioni continue arbitrarie $R_z(\theta)$ sarebbe esattamente universale, ma impossibile da realizzare in un hardware digitale soggetto a rumore. Al contrario, l'aggiunta della porta discreta T al set di Clifford $\{H, S, CNOT\}$ rende l'insieme *universalmente approssimante*: il Teorema di Solovay-Kitaev garantisce che qualsiasi operazione unitaria arbitraria (quindi qualsiasi fase arbitraria) possa essere approssimata con un errore ϵ arbitrariamente piccolo utilizzando una sequenza finita di porte H e T di lunghezza polilogaritmica $O(\log^c(1/\epsilon))$ [6, 5].

  Questo implica che il compilatore quantistico deve decomporre le rotazioni continue definite dall'algoritmo in lunghe sequenze di porte discrete T e H, con un impatto diretto sulla profondità del circuito e sulla gestione degli errori [3].

1.3.2 Porte a più Qubit

Le porte a singolo qubit non sono sufficienti per sfruttare la piena potenza del calcolo quantistico. Per eseguire algoritmi complessi è necessario far interagire i qubit tra loro attraverso porte a più input.

La porta CNOT (Controlled-NOT)

La porta CNOT opera su due qubit: un qubit di controllo ($|c\rangle$) e un qubit target ($|t\rangle$). La sua importanza nell'architettura dei computer quantistici è centrale: è stato dimostrato che l'insieme costituito dalla porta CNOT e dalle rotazioni a singolo qubit costituisce un **set universale**. Ciò significa che qualsiasi operazione unitaria su n qubit può essere decomposta in una sequenza finita di porte CNOT e porte a singolo qubit [6, 1].

Nella base computazionale $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, la matrice della CNOT è:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.20}$$

L'azione della porta sugli stati di base è spesso descritta come una generalizzazione dell'operazione classica XOR ($\oplus$):

$$|c\rangle|t\rangle \xrightarrow{CNOT} |c\rangle|t \oplus c\rangle \tag{1.21}$$

Se il controllo è $|1\rangle$, il target viene invertito (NOT); altrimenti resta invariato.

Tuttavia, è fondamentale comprendere che la CNOT **non è semplicemente una porta XOR**. L'analogia con lo XOR vale esclusivamente quando i qubit si trovano negli stati puri della base computazionale ($|0\rangle$ o $|1\rangle$). Quando il qubit di controllo si trova in uno stato di sovrapposizione, il comportamento diverge drasticamente dal modello classico. Consideriamo il caso in cui il controllo è in sovrapposizione uniforme $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ e il target è $|0\rangle$. Per la linearità dell'operatore quantistico:

$$CNOT\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}(CNOT|00\rangle + CNOT|10\rangle) = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{1.22}$$

Il risultato non è un valore logico derivato da un'operazione bit a bit, ma uno **stato entangled** (nello specifico, uno stato di Bell) [2, 3]. Mentre una porta

XOR classica propaga valori di bit (0 o 1), la porta CNOT quantistica propaga e correla le **ampiezze di probabilità** complesse. Questa capacità di creare correlazioni non locali (entanglement) è ciò che distingue la logica quantistica da una semplice logica classica probabilistica.

La porta di Toffoli (CCNOT)

La porta di Toffoli è un'operazione a tre qubit: due controlli e un target.

$$|x, y, z\rangle \xrightarrow{Toffoli} |x, y, z \oplus (x \cdot y)\rangle \tag{1.23}$$

dove $(\cdot)$ rappresenta l'AND logico. Classicamente, questa porta è universale per la computazione reversibile: può implementare le porte NAND e il FANOUT.

Anche in questo caso, è necessario insistere sulla distinzione fisica. Sebbene la tabella di verità sugli stati di base coincida con l'operazione classica "se control1 AND control2 allora NOT target", in un computer quantistico la Toffoli manipola ampiezze di probabilità in uno spazio di Hilbert di dimensione $2^3 = 8$. Come evidenziato da *Johnston et al.*, una porta Toffoli classica non potrebbe gestire input in cui le condizioni di controllo sono "parzialmente vere e parzialmente false" (sovrapposizione) [3]. Nel calcolo quantistico, la Toffoli agisce coerentemente su tutte le componenti della sovrapposizione simultaneamente, permettendo l'implementazione di oracoli complessi per algoritmi come quello di Grover o Shor, dove l'interferenza tra le ampiezze è il meccanismo che guida la computazione verso la soluzione corretta [5].

1.3.3 Il Teorema di No-Cloning

Nel design di circuiti classici, l'operazione di *fan-out* (copiare un bit in ingresso su più linee di uscita) è un'operazione banale e fondamentale. In informatica quantistica esiste una restrizione fisica rigorosa nota come Teorema di No-Cloning. Questo teorema afferma che è impossibile costruire un dispositivo unitario (e quindi un circuito quantistico) in grado di creare una copia identica di uno stato quantistico arbitrario sconosciuto [6, 5].

Dimostrazione per Linearità

Supponiamo per assurdo che esista un operatore unitario U (una macchina di clonazione) che, dato uno stato arbitrario $|\psi\rangle$ e un qubit ancillare inizializzato

a $|0\rangle$, produca in uscita due copie dello stato $|\psi\rangle$:

$$U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle \tag{1.24}$$

Consideriamo due stati ortogonali $|u\rangle$ e $|v\rangle$ (ad esempio $|0\rangle$ e $|1\rangle$). L'operatore U dovrebbe funzionare per entrambi:

$$U(|u\rangle \otimes |0\rangle) = |u\rangle \otimes |u\rangle \tag{1.25}$$

$$U(|v\rangle \otimes |0\rangle) = |v\rangle \otimes |v\rangle \tag{1.26}$$

Consideriamo ora uno stato $|\psi\rangle$ che sia una sovrapposizione generica $|\psi\rangle = \alpha|u\rangle + \beta|v\rangle$. A causa della linearità della meccanica quantistica, l'applicazione di U a questo stato produce:

$$U(|\psi\rangle \otimes |0\rangle) = \alpha U(|u\rangle \otimes |0\rangle) + \beta U(|v\rangle \otimes |0\rangle) = \alpha(|u\rangle \otimes |u\rangle) + \beta(|v\rangle \otimes |v\rangle) \tag{1.27}$$

Tuttavia, se U fosse un vero clonatore, dovrebbe produrre il tensore dello stato $|\psi\rangle$ con se stesso:

$$|\psi\rangle \otimes |\psi\rangle = (\alpha|u\rangle + \beta|v\rangle) \otimes (\alpha|u\rangle + \beta|v\rangle) = \alpha^2|uu\rangle + \beta^2|vv\rangle + \alpha\beta|uv\rangle + \beta\alpha|vu\rangle \tag{1.28}$$

Confrontando le due equazioni, notiamo che differiscono per i termini incrociati ($|uv\rangle$ e $|vu\rangle$). L'uguaglianza regge solo se $\alpha\beta = 0$, ovvero se lo stato è già uno degli stati di base. Non è quindi possibile clonare una sovrapposizione arbitraria sconosciuta [7].

Implicazioni

Il teorema di No-Cloning impone vincoli architetturali severi nello sviluppo di software quantistico:

- **Assenza di Backup:** Non è possibile salvare lo stato di un registro quantistico in una variabile temporanea per riutilizzarlo in seguito. Una volta che un qubit viene processato o misurato, il suo stato precedente è perso se non è possibile invertire l'intero circuito [11].
- **Correzione degli Errori:** Nei sistemi classici, la protezione contro i bit-flip si ottiene tramite ridondanza (es. codifica a ripetizione: $0 \to 000$). Poiché non possiamo clonare lo stato $|\psi\rangle$ in $|\psi\rangle|\psi\rangle|\psi\rangle$, i codici di correzione degli errori quantistici (QECC) devono utilizzare l'entan-

glement per proteggere l'informazione senza mai duplicarla o misurarla direttamente [4].

- **Fan-out limitato:** In un diagramma circuitale, una linea che si divide in due non rappresenta una copia fisica del segnale (come in un circuito elettrico), ma è vietata o rappresenta un gate CNOT che crea entanglement, non copie indipendenti [6].

1.4 Il Problema della Misura e il Collasso della Funzione d'Onda

Fino a questo punto, abbiamo descritto l'evoluzione di un sistema quantistico come un processo unitario e reversibile. Tuttavia, per estrarre informazioni dal sistema, è necessario effettuare una *misurazione*.

1.4.1 Osservabili e Operatori Hermitiani

In meccanica quantistica, una quantità fisica misurabile (osservabile) non è rappresentata da una semplice funzione, ma da un operatore lineare M che agisce sullo spazio degli stati. Un operatore M si dice **Hermitiano** (o autoaggiunto) se coincide con il suo trasposto coniugato:

$$M = M^\dagger \tag{1.29}$$

La proprietà distintiva degli operatori Hermitiani è che i loro autovalori sono garantiti essere reali. Poiché il risultato di una misurazione fisica (o la lettura di un registro in un computer) deve essere un numero reale, le osservabili devono necessariamente appartenere a questa classe [2].

Tuttavia, la sola Hermiticità non basta per descrivere completamente il processo di misura. È necessario invocare il **Teorema Spettrale**, il quale garantisce che ogni operatore Hermitiano su uno spazio vettoriale di dimensione finita sia **unitariamente diagonalizzabile** [6, 10]. Questo teorema assicura che esiste sempre una base ortonormale costituita dagli autovettori di M. Ciò è fondamentale perché implica che una misurazione proiettiva è sempre definita rispetto a una base valida dello spazio di Hilbert, permettendo al sistema di "collassare" in uno stato ben definito (un autostato) conservando la struttura probabilistica [7].

1.4.2 Formalismo delle Misure Proiettive

Grazie alla garanzia fornita dal Teorema Spettrale, un qualsiasi operatore Hermitiano M può essere scritto nella sua **decomposizione spettrale** (o rappresentazione diagonale):

$$M = \sum_m \lambda_m P_m \tag{1.30}$$

dove:

- λ_m sono gli autovalori reali dell'operatore (i valori che verranno letti dallo strumento di misura o dal registro classico).
- P_m sono i proiettori sui rispettivi autospazi degli autovettori. Se lo stato $|m\rangle$ è l'autovettore associato a λ_m, allora $P_m = |m\rangle\langle m|$ [6].

Secondo il postulato della misura (regola di Born), la probabilità di ottenere il risultato m misurando uno stato $|\psi\rangle$ è data da:

$$p(m) = \langle\psi| P_m |\psi\rangle = ||P_m |\psi\rangle||^2 \tag{1.31}$$

Per un singolo qubit misurato nella base computazionale $\{|0\rangle, |1\rangle\}$, l'osservabile è tipicamente l'operatore Z (matrice di Pauli σ_z). I proiettori associati sono $P_0 = |0\rangle\langle 0|$ e $P_1 = |1\rangle\langle 1|$. Dato uno stato generico $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, le probabilità sono [2]:

$$p(0) = |\alpha|^2, \quad p(1) = |\beta|^2 \tag{1.32}$$

1.4.3 Il Collasso della Funzione d'Onda

L'aspetto più critico è il cambiamento di stato indotto dalla misurazione. Se la misurazione produce il risultato m, lo stato del sistema *immediatamente dopo* la misurazione non è più il vettore originale $|\psi\rangle$, ma collassa nello stato proiettato normalizzato [6]:

$$|\psi'\rangle = \frac{P_m |\psi\rangle}{\sqrt{p(m)}} \tag{1.33}$$

Questo fenomeno è noto come **collasso della funzione d'onda**. Se misuriamo il qubit $\alpha|0\rangle + \beta|1\rangle$ e otteniamo il risultato classico "0", lo stato quantistico diventa istantaneamente $|0\rangle$, e le informazioni contenute precedentemente nelle ampiezze α e β vengono irrimediabilmente perse (o "cancellate") [3].

1.4.4 Irreversibilità e Debugging

Il collasso della funzione d'onda introduce vincoli severi per lo sviluppo e il debugging di algoritmi quantistici, differenziando drasticamente il paradigma quantistico da quello classico:

- **Impossibilità di lettura non distruttiva:** Non è possibile "ispezionare" il valore delle ampiezze di probabilità (α, β) durante l'esecuzione di un algoritmo reale. Qualsiasi tentativo di leggere lo stato forza il sistema in uno stato base, distruggendo la sovrapposizione e l'interferenza necessarie per la computazione [4].

- **Debugging Probabilistico:** In un debugger classico, è possibile sospendere l'esecuzione e leggere le variabili. In un computer quantistico reale, leggere lo stato altera il flusso del programma. Di conseguenza, il debugging si basa spesso sull'esecuzione ripetuta del circuito (migliaia di "shots") per ricostruire la distribuzione di probabilità dell'output, piuttosto che sulla verifica deterministica di una singola esecuzione [3].

- **Irreversibilità Logica:** Mentre le porte unitarie descritte nella Sezione 1.3 sono reversibili (non dissipano informazione), l'operazione di misura è intrinsecamente irreversibile. In un diagramma circuitale, la misura rappresenta il confine tra il dominio quantistico e il dominio classico, dove l'informazione quantistica viene convertita in bit classici per l'elaborazione successiva [10].

Capitolo 2

Il Modello Circuitale

Il modello circuitale rappresenta l'astrazione standard per la computazione quantistica, analogo ai circuiti logici booleani nell'informatica classica. In questo modello, l'informazione è codificata in un registro di qubit e l'elaborazione avviene attraverso l'applicazione sequenziale di porte logiche (quantum gates). A differenza dei circuiti classici, dove i fan-out e i loop di feedback sono comuni, un circuito quantistico è descritto matematicamente come una sequenza di operatori unitari che evolvono lo stato del sistema in modo reversibile, terminando generalmente con una misurazione [6].

Qui esamineremo i requisiti minimi affinché un insieme di porte possa eseguire qualsiasi computazione quantistica (universalità) e le implicazioni ingegneristiche della composizione di tali porte.

2.1 Universalità dei gate quantistici

Nel calcolo classico, è noto che la porta NAND costituisce un insieme universale: qualsiasi funzione booleana $f : \{0,1\}^n \to \{0,1\}^m$ può essere calcolata utilizzando una rete composta esclusivamente da porte NAND. Nel calcolo quantistico, il problema è più complesso poiché lo spazio degli stati è continuo. Cerchiamo un insieme finito di porte elementari che permetta di approssimare qualsiasi operazione unitaria U su n qubit con una precisione arbitraria.

2.1.1 Definizione di Universalità

Un insieme di porte quantistiche si dice *universale* se, per ogni operazione unitaria U agente su n qubit e per ogni errore $\epsilon > 0$, esiste una sequenza finita di porte appartenenti all'insieme che implementa un operatore V tale che la

distanza tra U e V sia inferiore a ϵ [1, 6]. Formalmente, si definisce l'errore $E(U, V)$ come:

$$E(U, V) \equiv \max_{|\psi\rangle} \|(U - V)\,|\psi\rangle\| \tag{2.1}$$

dove il massimo è preso su tutti gli stati quantistici normalizzati $|\psi\rangle$ [6].

Da una prospettiva tradizionale, è necessario fare un'osservazione preliminare sulla natura del risultato computazionale. Poiché l'estrazione dell'informazione avviene tramite misurazioni proiettive, i risultati degli algoritmi quantistici sono intrinsecamente probabilistici (ad eccezione di casi specifici deterministici come l'algoritmo di Deutsch-Jozsa). Anche disponendo di un hardware ideale, il collasso della funzione d'onda introduce una varianza statistica intrinseca nell'output [5]. Di conseguenza, il concetto classico di "algoritmo esatto" lascia il posto a quello di approssimazione con probabilità di successo limitata (bounded-error), tipica della classe di complessità BQP. Questo fatto fisico giustifica l'approccio di cercare una computazione approssimata anche a livello di porte logiche: se la lettura finale è soggetta a statistica, è accettabile che anche l'evoluzione unitaria U sia implementata da un operatore V che approssima U entro una tolleranza ϵ, purché tale errore sia statisticamente indistinguibile dal risultato ideale rispetto alla precisione di misura richiesta [6].

2.1.2 Scomposizione in Gate a Singolo e Doppio Qubit

Un risultato fondamentale per l'architettura dei processori quantistici stabilisce che qualsiasi matrice unitaria $U \in U(2^n)$ può essere decomposta esattamente in una sequenza di porte che operano su un singolo qubit e porte CNOT (Controlled-NOT) [1, 11]. Matematicamente, questo significa che è possibile costruire qualsiasi evoluzione complessa dell'intero registro manipolando individualmente i singoli qubit e introducendo l'entanglement esclusivamente tramite la porta CNOT [11].

Williams sottolinea che sebbene l'insieme di tutti i gate a un qubit più il CNOT sia universale, esistono altre famiglie di gate universali che potrebbero essere più efficienti a seconda dell'implementazione hardware fisica (es. gate iSWAP o $\sqrt{\text{SWAP}}$) [11].

2.1.3 Approssimazione Discreta e Teorema di Solovay-Kitaev

Riprendendo la distinzione tra universalità esatta e approssimata introdotta nella sezione precedente (cfr. Porte di Fase), osserviamo che non è fisicamente realizzabile un hardware che supporti un continuo di rotazioni a singolo qubit con precisione infinita e priva di errori. È dunque necessario definire un set di istruzioni discreto e finito, spesso noto come *fault-tolerant gate set*.

Un insieme discreto comunemente utilizzato è l'insieme di Clifford+T:

$$\mathcal{G} = \{H, S, T, CNOT\} \tag{2.2}$$

dove, come definito nel capitolo precedente, H è la porta di Hadamard, S è la porta di fase ($Z^{1/2}$) e T è la porta $\pi/8$ ($Z^{1/4}$), la cui aggiunta è essenziale per rendere l'insieme denso nello spazio degli operatori unitari [6].

Il **Teorema di Solovay-Kitaev** formalizza matematicamente la garanzia di efficienza discussa preliminarmente. Esso garantisce che è possibile approssimare qualsiasi gate unitario a singolo qubit con un errore ϵ utilizzando una sequenza di gate dall'insieme discreto $\mathcal{G}$ di lunghezza polilogaritmica in $1/\epsilon$, specificamente $O(\log^c(1/\epsilon))$ con $c \approx 2$ [2, 6]. Questo risultato teorico è fondamentale per l'ingegneria del software quantistico, poiché assicura che il costo computazionale (in termini di profondità del circuito) necessario per aumentare la precisione non cresca esponenzialmente, rendendo fattibile la computazione quantistica approssimata anche con un set di istruzioni limitato.

2.1.4 Il ruolo del Transpiler e i Basis Gates

Nello stack software moderno, come evidenziato da Kasirajan, il concetto teorico di universalità viene gestito operativamente dal componente noto come *Transpiler* [4]. Il programmatore definisce l'algoritmo usando porte logiche astratte di alto livello. Il transpiler ha il compito di tradurre questo circuito astratto nel set di *basis gates* supportati nativamente dalla topologia specifica del backend hardware.

Ad esempio, un processore reale potrebbe non supportare direttamente una porta CNOT tra due qubit non adiacenti fisicamente. Il transpiler deve quindi iniettare porte SWAP aggiuntive per spostare gli stati logici sui qubit fisici connessi, aumentando la profondità del circuito e, conseguentemente, l'accumulo di rumore [4]. Un tipico set di basis gates per processori superconduttori

(come quelli IBM) include:

$$\text{Basis Gates} = \{\text{ID}, \text{RZ}, \text{SX}, \text{X}, \text{CNOT}\} \tag{2.3}$$

La decomposizione efficiente del circuito in questi gate base è un problema di ottimizzazione complesso che impatta direttamente le performance dell'algoritmo [4].

2.2 Circuiti Quantistici

Un circuito quantistico è un modello computazionale in cui l'elaborazione dell'informazione è rappresentata come una sequenza di porte logiche che agiscono su un registro di qubit, terminando tipicamente con operazioni di misurazione.

2.2.1 Rappresentazione Grafica e Flusso Temporale

La notazione standard per i circuiti quantistici utilizza diagrammi in cui:

- **Linee Orizzontali (Wires):** Ogni linea rappresenta un qubit e il tempo scorre da sinistra verso destra. A differenza dei circuiti elettrici classici, queste linee non denotano necessariamente un cavo fisico attraverso cui viaggia un segnale. In molte architetture (come i qubit superconduttori o gli ioni intrappolati), il qubit è un sistema fisico stazionario e la linea rappresenta la sua evoluzione temporale [11].
- **Box (Gates):** I blocchi posti sulle linee rappresentano le operazioni unitarie.
- **Controlli:** I punti pieni (o vuoti) collegati da linee verticali indicano che l'applicazione di un gate è condizionata dallo stato di un altro qubit (controllo) [1].
- **Misurazioni:** Il simbolo del misuratore indica la proiezione dello stato quantistico su una base classica (solitamente la base Z). Dopo questo punto, la linea diventa doppia per indicare che trasporta un bit classico [4].

2.2.2 Composizione Matematica dei Gate

Per un ingegnere del software, è cruciale comprendere come la rappresentazione grafica si traduca in operazioni algebriche lineari. Esistono due modalità di composizione:

1. **Composizione Seriale (Prodotto Righe per Colonne):** Se un gate A viene applicato al tempo t_1 e un gate B al tempo t_2 (con $t_2 > t_1$) sullo stesso insieme di qubit, l'operatore risultante è il prodotto matriciale $B \cdot A$.

2. **Composizione Parallela (Prodotto Tensoriale):** Se un gate A agisce su un sottosistema di qubit e, simultaneamente, un gate B agisce su un altro sottosistema disgiunto, l'operazione complessiva è descritta dal prodotto tensoriale $A \otimes B$ [11, 10].

   Nota Bene: L'ordine dei gate nel diagramma circuitale (sinistra $\rightarrow$ destra) è *inverso* rispetto all'ordine di moltiplicazione delle matrici nell'algebra lineare (destra $\rightarrow$ sinistra). Se nel circuito appare prima A e poi B, matematicamente si scrive $BA\,|\psi\rangle$, poiché l'operatore più a destra è il primo ad essere applicato al vettore di stato [11].

2.2.3 Esempio Fondamentale: Creazione di uno Stato di Bell

Uno degli esempi più illustrativi è il circuito per generare uno stato di Bell, ovvero uno stato massimamente entangled. Questo circuito è la primitiva base per protocolli come il teletrasporto quantistico [6, 4].

Il circuito è composto da una porta Hadamard (H) sul primo qubit, seguita da una porta CNOT tra i due qubit.

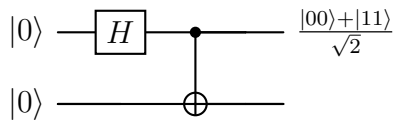

Figura 2.1: Circuito per la generazione dello stato di Bell $|\Phi^+\rangle$.

Analizziamo l'evoluzione dello stato passo dopo passo, prestando attenzione alla struttura tensoriale del sistema [2, 1]:

1. **Stato Iniziale**: Il registro è inizializzato nello stato fondamentale.

$$|\psi_0\rangle = |0\rangle \otimes |0\rangle = |00\rangle \tag{2.4}$$

2. **Applicazione di** $H \otimes I$: Si applica l'operatore tensoriale $H \otimes I$ al registro globale. Sebbene fisicamente il gate agisca sul primo qubit, matematicamente l'operazione è definita sull'intero spazio di Hilbert $\mathbb{C}^4$:

$$|\psi_1\rangle = (H \otimes I)|00\rangle = (H|0\rangle) \otimes (I|0\rangle) = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}} \tag{2.5}$$

   Il sistema si trova ora in una sovrapposizione prodotto, dove il primo qubit è in sovrapposizione e il secondo è determinato.

3. **Applicazione di CNOT**: La porta CNOT è intrinsecamente un operatore a due qubit che agisce sull'intero registro. Poiché il qubit di controllo (il primo) è in sovrapposizione, l'azione della CNOT si applica coerentemente a ciascun componente della sovrapposizione (linearità):

$$|\psi_2\rangle = CNOT|\psi_1\rangle = \frac{1}{\sqrt{2}}(CNOT|00\rangle + CNOT|10\rangle) = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.6}$$

Il risultato $|\Phi^+\rangle$ è uno stato in cui i valori dei due qubit sono perfettamente correlati, nonostante nessuno dei due abbia un valore definito singolarmente [6].

2.3 Misurazione e Output Classico

Al termine della computazione, l'informazione quantistica deve essere estratta per renderla accessibile al mondo macroscopico. L'operazione di misurazione proietta lo stato quantistico su uno degli autostati della base computazionale. In Qiskit e in altri framework, questa operazione converte i bit quantistici (q) in bit classici (c), permettendo all'utente di accedere ai risultati sotto forma di stringhe binarie [2].

Consideriamo come esempio pratico la misurazione dello Stato di Bell $|\Phi^+\rangle$ generato nella sezione precedente:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.7}$$

Matematicamente, la probabilità di ottenere una specifica stringa binaria x è data dal modulo quadro dell'ampiezza corrispondente (Regola di Born) [6]:

- Probabilità di leggere "00": $P(00) = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} = 50\%$
- Probabilità di leggere "11": $P(11) = |\frac{1}{\sqrt{2}}|^2 = \frac{1}{2} = 50\%$
- Probabilità di leggere "01" o "10": $P = 0$ (poiché le ampiezze sono nulle).

Poiché il risultato della singola esecuzione è probabilistico, non è sufficiente eseguire il circuito una sola volta. È necessario eseguire il circuito ripetutamente (un numero di volte definito *shots*, tipicamente 1024 o 4096) per ricostruire la distribuzione di probabilità empirica (istogramma) che approssima le probabilità teoriche calcolate sopra [4]. Questo processo evidenzia il cambio di paradigma nel testing software: la validazione di un algoritmo quantistico non avviene verificando una singola uscita deterministica, ma analizzando la statistica dei risultati su grandi campioni.

Capitolo 3

Architettura Hardware

Nei capitoli precedenti abbiamo trattato i qubit e le porte logiche come entità matematiche ideali: vettori di stato che evolvono in modo perfettamente unitario e deterministico. La transizione alla realizzazione fisica introduce però sfide ingegneristiche sostanziali che hanno un impatto diretto sullo sviluppo del software.

L'elettronica digitale moderna gode di un'astrazione robusta; l'hardware quantistico attuale opera invece nel regime *NISQ* (Noisy Intermediate-Scale Quantum) [4]. In questo contesto, i qubit fisici sono intrinsecamente rumorosi e soggetti a vincoli topologici che l'astrazione del circuito logico tende a nascondere. Queste limitazioni non sono dettagli trascurabili, ma vincoli di progetto: la decoerenza limita la profondità massima degli algoritmi, mentre la topologia del chip impone l'uso di compilatori intelligenti per il routing dei segnali.

Nelle pagine seguenti approfondiremo i requisiti fondamentali che un sistema fisico deve soddisfare per supportare la computazione quantistica e forniremo una panoramica delle tecnologie realizzative attuali, evidenziando le implicazioni per lo stack software.

3.1 I Criteri di DiVincenzo: Requisiti del Modello Circuitale

L'implementazione fisica di un calcolatore quantistico non può prescindere da una serie di requisiti stringenti che garantiscano la fedeltà e la scalabilità della computazione. Nel 2000, David DiVincenzo ha formalizzato tali requisiti in quello che è oggi noto come lo standard di riferimento per la validazione di qual-

siasi architettura hardware quantistica [11]. Questi criteri non rappresentano semplici vincoli fisici, ma definiscono le specifiche funzionali della *Macchina Astratta* su cui il software deve operare.

In sostanza questi criteri possono essere letti come le fasi operative necessarie a sostenere il ciclo di vita dell'informazione quantistica: la definizione del registro, l'inizializzazione, l'evoluzione controllata e la lettura, il tutto vincolato dalla persistenza della coerenza.

3.1.1 Il Substrato Fisico e l'Inizializzazione

Il punto di partenza per qualsiasi computazione è l'esistenza di un sistema fisico scalabile composto da qubit ben caratterizzati (Criterio 1) [4]. Questo implica la capacità di indirizzare singolarmente i qubit all'interno di uno spazio di Hilbert di dimensione 2^n, dove n è il numero di qubit fisici disponibili. La scalabilità non riguarda solo l'aumento del numero di qubit, ma la capacità di mantenere le interazioni parassite (crosstalk) entro limiti accettabili all'aumentare delle dimensioni del registro.

Una volta definito il registro, il sistema deve garantire la capacità di inizializzare lo stato dei qubit in un valore fiduciale noto (Criterio 2), tipicamente lo stato fondamentale $|00\ldots0\rangle$ [2]. Questa operazione corrisponde, in termini ingegneristici, al *reset* o al *boot* del sistema: senza la capacità di preparare uno stato deterministico puro a bassa entropia, non è possibile avviare alcun algoritmo in modo affidabile [4].

3.1.2 Evoluzione Unitaria e Universalità

Il cuore della computazione risiede nella manipolazione dello stato inizializzato. DiVincenzo richiede la disponibilità di un set universale di porte quantistiche (Criterio 4) [6]. Un set di gate si definisce universale se, attraverso una sequenza finita di istruzioni, è possibile approssimare qualsiasi operazione unitaria $U \in U(2^n)$ con precisione arbitraria [10].

Questo criterio definisce l'*Instruction Set Architecture* (ISA) della macchina. Sebbene teoricamente sia possibile decomporre qualsiasi circuito in rotazioni a singolo qubit e una porta entanglante a due qubit (come la CNOT), la topologia hardware specifica potrebbe imporre vincoli su quali qubit possono interagire direttamente, costringendo il compilatore (o *transpiler*) a inserire operazioni di SWAP aggiuntive [3, 4].

3.1.3 Il Vincolo Temporale: Coerenza vs Operazioni

Il limite fondamentale alla profondità di un circuito quantistico è dettato dai tempi di decoerenza. Questi devono essere significativamente più lunghi del tempo di esecuzione dei gate (Criterio 3) [11]. Sia T_{gate} il tempo medio per eseguire un'operazione logica elementare e T_{coh} il tempo caratteristico di decoerenza (spesso approssimato come $\min(T_1, T_2)$, dove T_1 è il tempo di rilassamento e T_2 il tempo di dephasing [4]). Il numero massimo di operazioni N_{op} eseguibili prima che l'informazione sia compromessa è approssimabile come:

$$N_{op} \approx \frac{T_{coh}}{T_{gate}} \tag{3.1}$$

Questo rapporto definisce il *budget temporale* a disposizione del programmatore. In architetture NISQ (*Noisy Intermediate-Scale Quantum*), dove la correzione d'errore non è pienamente implementata, superare questa profondità comporta risultati dominati dal rumore [4].

3.1.4 Misurazione e Comunicazione

Il ciclo computazionale si conclude con la necessità di estrarre il risultato tramite una misurazione specifica per qubit (Criterio 5) [6]. Questa capacità deve permettere la proiezione dello stato quantistico su una base classica (tipicamente la base Z) senza disturbare i qubit non misurati, abilitando così protocolli avanzati come la correzione d'errore attiva e la computazione basata sulla misura [4, 5].

Infine, per sistemi distribuiti o architetture modulari, DiVincenzo identifica due ulteriori criteri legati alla *comunicazione quantistica*: la capacità di interconvertire qubit stazionari (memoria) in qubit volanti (fotoni per la trasmissione) e la capacità di trasmettere fedelmente questi ultimi tra locazioni distinte [4]. Questi ultimi requisiti, sebbene meno critici per singoli processori isolati, sono fondamentali per la scalabilità a lungo termine verso l'Internet Quantistico.

3.1.5 Qubit Logici vs Fisici e la Barriera della Correzione d'Errore

L'applicazione rigorosa dei criteri di DiVincenzo si scontra, nella pratica, con la fragilità degli stati quantistici. Per realizzare un'architettura robusta (Fault-Tolerant Quantum Computing) è necessario introdurre un livello di astrazio-

ne che disaccoppi l'algoritmo dalle imperfezioni hardware. Questo porta alla distinzione fondamentale tra:

- **Qubit Fisici**: Sono i dispositivi hardware effettivi (es. ioni, circuiti superconduttori) descritti nella sezione successiva. Sono soggetti a decoerenza e rumore operativo.
- **Qubit Logici**: Sono unità di informazione astratte, composte da un insieme di qubit fisici entangled tra loro attraverso codici di correzione d'errore quantistico (QEC), come il codice di superficie (Surface Code) [4].

In un'architettura robusta, l'algoritmo viene definito e compilato (dal transpiler) sui qubit logici, che garantiscono tempi di coerenza arbitrariamente lunghi a patto che il tasso di errore fisico sia inferiore a una soglia critica (Threshold Theorem) [6]. Tuttavia, il costo di questa astrazione è elevatissimo in termini di risorse: la ridondanza necessaria per correggere gli errori implica che un singolo qubit logico possa richiedere da decine a migliaia di qubit fisici, a seconda del tasso di errore dell'hardware sottostante e del codice utilizzato. Stime attuali suggeriscono un rapporto che può variare da 1:100 fino a 1:1000 per implementazioni pratiche di algoritmi complessi come quello di Shor [2, 4]. Di conseguenza, la capacità di storage nominale di un chip (livello fisico) non corrisponde alla capacità computazionale effettiva (livello logico), un fattore da considerare nella valutazione della scalabilità degli algoritmi.

3.2 Realizzazioni Fisiche e Implicazioni Architetturali

Esistono molteplici approcci fisici per implementare un qubit. Attualmente, le due tecnologie dominanti nell'industria e accessibili tramite servizi cloud sono i *Qubit Superconduttori* (adottati da IBM e Google) e gli *Ioni Intrappolati* (adottati da IonQ e Honeywell). La scelta della tecnologia sottostante determina il set di istruzioni native, la topologia e il profilo di rumore che il software deve gestire.

3.2.1 Qubit Superconduttori

I qubit superconduttori sono circuiti macroscopici LC (induttore-condensatore) resi anarmonici tramite un elemento non lineare, la *Giunzione Josephson*, che

permette di isolare i due livelli energetici $|0\rangle$ e $|1\rangle$. Questi dispositivi operano a temperature criogeniche (15 mK) [4].

Controllo Software

A livello fisico, l'evoluzione dello stato è pilotata tramite impulsi a microonde. Per l'utilizzatore, la frequenza, la durata e la fase di questi impulsi corrispondono direttamente agli operatori di rotazione sulla Sfera di Bloch (discussi nel Cap. 1). Ad esempio, un impulso calibrato può implementare fisicamente la porta logica $R_x(\pi)$.

Vincoli Topologici

Questi chip presentano tipicamente una topologia planare a griglia, dove ogni qubit è connesso solo ai vicini prossimi (*nearest-neighbor*). Facendo riferimento al grafo G introdotto nella Sezione 3.1.1, questo implica che il grafo è sparso. Di conseguenza, l'esecuzione di un gate a due qubit tra operandi non adiacenti richiede l'inserimento di catene di gate SWAP da parte del transpiler, aumentando drasticamente la profondità del circuito [2].

3.2.2 Ioni Intrappolati

Questa tecnologia utilizza singoli atomi ionizzati (es. Ytterbio o Calcio) confinati nello spazio tramite campi elettromagnetici e manipolati tramite laser. I qubit sono "naturali": ogni ione di una data specie è identico a qualsiasi altro, garantendo un'altissima uniformità [6].

Connettività e Trade-off

A differenza dei superconduttori, gli ioni in una singola trappola lineare offrono una connettività **all-to-all** all'interno della catena: è possibile realizzare operazioni di entanglement tra qualsiasi coppia di ioni sfruttando i modi vibrazionali collettivi. Questo riduce a zero (o quasi) il costo di routing del compilatore. Tuttavia, questo vantaggio è bilanciato da tempi di esecuzione dei gate generalmente più lenti (ordine dei microsecondi) rispetto ai nanosecondi dei superconduttori [4].

Caratteristica	**Superconduttori**	**Ioni Intrappolati**
Natura del Qubit	Artificiale (Circuito)	Naturale (Atomo)
Velocità Gate	Molto Alta (ns)	Bassa (μs)
Coerenza	Bassa (μs)	Molto Alta (s)
Connettività	Nearest-Neighbor	All-to-All (locale)
Overhead SWAP	Alto (Critico)	Nullo o Basso

Tabella 3.1: Confronto tra le principali tecnologie hardware [4].

Capitolo 4

Complessità Computazionale e Supremazia

L'analisi degli algoritmi quantistici richiede un cambiamento di prospettiva rispetto ai paradigmi classici di valutazione delle risorse. Tradizionalmente, l'efficienza è misurata in termini di tempo di esecuzione e occupazione di memoria in funzione della dimensione dell'input. L'avvento del calcolo quantistico impone oggi l'integrazione di nuove metriche che tengano conto della natura probabilistica dell'output e delle risorse fisiche, come il numero di qubit e la profondità del circuito. Questo capitolo analizza come il modello quantistico ridefinisca i confini della trattabilità computazionale, delineando le nuove classi di complessità e introducendo il concetto di supremazia quantistica, intesa come il punto di demarcazione in cui l'hardware quantistico supera le capacità di simulazione delle più potenti architetture classiche esistenti.

4.1 Classi di Complessità e Trattabilità

La teoria della complessità classifica i problemi in base alla crescita delle risorse necessarie per risolverli al crescere dell'input n. Questa tassonomia è lo strumento fondamentale per distinguere ciò che è praticamente realizzabile da ciò che è teoricamente possibile ma intrattabile.

Nel dominio classico, la distinzione primaria è tra la classe **P** (Polynomial time), che racchiude i problemi risolvibili da una macchina di Turing deterministica in tempo polinomiale $O(n^k)$, e la classe **NP** (Nondeterministic Polynomial time), che include problemi per i quali è possibile verificare una soluzione in tempo polinomiale, ma non necessariamente trovarla con la stessa efficienza. La coincidenza tra le due classi rimane uno dei problemi aperti

più rilevanti della matematica, con profonde implicazioni per la crittografia moderna [2].

L'introduzione della meccanica quantistica richiede l'estensione di questo modello per accomodare il probabilismo intrinseco. Prima di definire la classe quantistica, è utile considerare la classe classica **BPP** (Bounded-error Probabilistic Polynomial time). Questa classe descrive problemi risolvibili da un computer classico dotato di un generatore di casualità, con una probabilità di errore limitata (tipicamente $\epsilon < 1/3$). Un aspetto ingegneristico cruciale di BPP è che l'errore può essere ridotto arbitrariamente ripetendo l'algoritmo e applicando un criterio di maggioranza [6].

La controparte quantistica di BPP è la classe **BQP** (Bounded-error Quantum Polynomial time). Essa rappresenta l'insieme dei problemi decisionali risolvibili da un computer quantistico in tempo polinomiale con una probabilità di errore limitata. Formalmente, un linguaggio appartiene a BQP se esiste una famiglia uniforme di circuiti quantistici di dimensione polinomiale tale che il circuito fornisca la risposta corretta con probabilità $\geq 2/3$. Un algoritmo quantistico efficiente non fornisce necessariamente la risposta corretta con certezza in una singola esecuzione; tuttavia, garantendo una separazione statistica sufficiente dalla pura casualità, è possibile amplificare la probabilità di successo con un overhead computazionale gestibile [11].

La relazione formale tra le classi di complessità classiche e quantistiche è descritta dalla catena di inclusioni:

$$P \subseteq BPP \subseteq BQP \subseteq PSPACE \tag{4.1}$$

Mentre l'inclusione $BPP \subseteq BQP$ è dimostrata (poiché un computer quantistico può simulare efficientemente qualsiasi macchina di Turing probabilistica), la questione se BQP sia strettamente più grande di BPP ($BQP \neq BPP$) rimane una congettura aperta. Tuttavia, l'esistenza di algoritmi come quello di Shor, che risolvono problemi in BQP ritenuti intrattabili per BPP, rende ragionevole supporre che tale separazione esista [6, 2].

La classe BQP rappresenta un limite teorico superiore, definito su hardware ideale scalabile. Gli algoritmi effettivamente implementabili su macchine reali potrebbero costituire un sottoinsieme più ristretto di BQP, limitato da vincoli fisici quali la connettività dei qubit, i tempi di coerenza e la profondità massima del circuito tollerabile prima che il rumore degradi il risultato [4].

4.2 Relazione tra Classico e Quantistico

Per comprendere l'effettivo potenziale dell'architettura quantistica, è necessario analizzare la relazione bidirezionale di simulazione: la capacità di una macchina quantistica di eseguire logica classica e, viceversa, i limiti di una macchina classica nel replicare l'evoluzione quantistica.

Il calcolo quantistico è, in prima approssimazione, una generalizzazione del calcolo classico reversibile. Le porte logiche classiche standard (come NAND o XOR) sono irreversibili poiché comportano una perdita di informazione. È comunque possibile mappare qualsiasi circuito classico in un circuito reversibile equivalente utilizzando porte come la porta di Toffoli, che è un operatore unitario e quindi una valida porta quantistica. Di conseguenza, un computer quantistico può emulare efficientemente qualsiasi computazione classica [6]. In questa modalità operativa, però, la macchina non sfrutta sovrapposizione o entanglement, comportandosi come un processore classico senza offrire vantaggi computazionali.

La relazione inversa presenta invece ostacoli insormontabili. La descrizione matematica di uno stato quantistico di n qubit richiede la memorizzazione di 2^n ampiezze complesse. Se un registro classico di n bit si trova in un unico stato determinato, quello quantistico può esistere in una sovrapposizione di tutti i 2^n stati base. Simulare l'evoluzione di tale sistema su una macchina di Turing classica richiede risorse di memoria e tempo che crescono esponenzialmente con il numero di qubit. Come osservato da Feynman, la simulazione esatta diventa intrattabile all'aumentare delle dimensioni del sistema, poiché ogni operazione di gate corrisponde a un'algebra lineare su matrici di dimensioni $2^n \times 2^n$ [11].

Questa asimmetria ha portato alla revisione della **Tesi di Church-Turing Estesa**, la quale asseriva che qualsiasi processo fisico potesse essere simulato efficientemente (con overhead polinomiale) da una macchina di Turing probabilistica. L'esistenza di algoritmi quantistici che offrono speedup esponenziali suggerisce che questa tesi sia falsa, portando alla formulazione della "Tesi di Church-Turing Quantistica", che pone il computer quantistico come modello universale per la simulazione efficiente dei processi fisici [2].

4.3 Supremazia Quantistica e Validazione

Il termine *Supremazia Quantistica* (Quantum Supremacy), introdotto da John Preskill nel 2012, definisce il momento in cui un dispositivo quantistico esegue un compito computazionale che risulta praticamente impossibile per qualsiasi

computer classico esistente [2]. Tuttavia questa è una nozione delicata da definire e richiede una distinzione fondamentale tra due ambiti applicativi:

- **Task Specifici (NISQ):** Esistono risultati sperimentali, come quelli ottenuti tramite il *Random Circuit Sampling* (campionamento da circuiti casuali), in cui processori quantistici reali (es. Google Sycamore) hanno completato in pochi minuti operazioni che richiederebbero millenni ai migliori supercomputer classici [4, 2]. Sebbene questi risultati dimostrino una capacità computazionale superiore in ambiti ristretti, essi sono spesso contestati dalla comunità scientifica a causa del continuo miglioramento degli algoritmi di simulazione classica e della mancanza di utilità pratica immediata del task svolto.
- **Calcolo General Purpose:** Se si intende la supremazia nell'ambito di un *General Purpose Quantum Computer* (un computer universale, scalabile e fault-tolerant), la supremazia è ancora una congettura da dimostrare fisicamente. Non esiste ancora un hardware capace di eseguire algoritmi complessi come la fattorizzazione di Shor su numeri di interesse crittografico (che richiederebbe milioni di qubit fisici per la correzione d'errore) superando le controparti classiche [4, 6].

In questo contesto faremo riferimento principalmente alla prospettiva **General Purpose**. L'obiettivo è analizzare come il modello circuitale permetta, in linea di principio, di ottenere vantaggi computazionali su problemi di utilità pratica, assumendo la disponibilità futura di un hardware sufficientemente robusto. Questo scenario impone sfide uniche per la validazione del software: quando si opera in regime di supremazia, la verifica diretta dei risultati diventa problematica poiché, per definizione, nessun computer classico può calcolare la soluzione di controllo in tempi ragionevoli.

4.3.1 Il Problema della Verifica

Nel regime della supremazia quantistica, emerge una sfida fondamentale per l'architettura classica: la validazione dell'output. Se il computer quantistico risolve un problema che nessun supercomputer classico può simulare in tempi ragionevoli, non esiste un "oracolo" classico con cui confrontare il risultato per confermarne la correttezza (testing dell'oracolo).

Per problemi in NP come la fattorizzazione (Shor), la verifica è efficiente: data la soluzione (i fattori primi), un computer classico può verificare rapidamente se il loro prodotto corrisponde al numero originale. Tuttavia, per

problemi di campionamento di circuiti random (spesso usati per dimostrare la supremazia), la verifica della distribuzione di probabilità in uscita diventa esponenzialmente costosa per una macchina classica [6].

4.3.2 Implicazioni per lo Sviluppo

Questo scenario impone l'adozione di nuove metodologie di testing e debugging:

- **Verifica su piccola scala:** Gli algoritmi vengono validati su istanze ridotte del problema, dove la simulazione classica è ancora trattabile (tipicamente sotto i 50 qubit), per poi estrapolare la correttezza su scala maggiore.
- **Benchmarking incrociato:** Si utilizzano metriche statistiche, come la *Cross-Entropy Benchmarking*, per quantificare quanto la distribuzione degli stati misurati si avvicini a quella teorica ideale, pur senza poter calcolare esattamente ogni singola ampiezza.
- **Codici di Correzione d'Errore:** Poiché l'hardware attuale è rumoroso (NISQ), diventa essenziale distinguere tra errori dovuti al rumore fisico e errori logici nell'algoritmo. L'uso di tecniche di mitigazione dell'errore diventa parte integrante del ciclo di sviluppo software.

In conclusione, la supremazia quantistica non è un punto di arrivo, ma l'ingresso in un regime computazionale dove le tradizionali garanzie di correttezza deterministica lasciano il posto a validazioni di natura statistica e inferenziale.

Capitolo 5

Algoritmi Quantistici

Nei capitoli precedenti è stato formalizzato il modello circuitale come astrazione standard per la descrizione dei processi quantistici, definendo i qubit come vettori in spazi di Hilbert complessi e le porte logiche come operatori unitari. Tuttavia, la mera esistenza di uno spazio degli stati che cresce esponenzialmente con il numero di qubit non è condizione sufficiente per garantire un vantaggio computazionale rispetto alle architetture classiche.

La progettazione di algoritmi quantistici richiede un approccio radicalmente diverso rispetto alla programmazione tradizionale. Mentre un algoritmo classico deterministico produce una sequenza definita di stati, e un algoritmo probabilistico (classe BPP) sfrutta la casualità per esplorare lo spazio delle soluzioni, un algoritmo quantistico (classe BQP) deve essere ingegnerizzato per sfruttare specifici fenomeni fisici: la sovrapposizione, l'entanglement e, soprattutto, l'interferenza quantistica [6].

L'obiettivo di un algoritmo quantistico non è semplicemente eseguire calcoli in parallelo, bensì manipolare le ampiezze di probabilità dei vettori di stato in modo tale che, attraverso un processo di interferenza costruttiva, la probabilità di misurare lo stato corrispondente alla soluzione corretta venga amplificata, mentre le probabilità associate alle risposte errate vengano soppresse tramite interferenza distruttiva [5].

Nei prossimi paragrafi analizzeremo le primitive algoritmiche fondamentali che costituiscono la base del vantaggio quantistico ("quantum advantage"). Inizieremo con il concetto di parallelismo quantistico, per poi esaminare l'algoritmo di Deutsch-Jozsa come primo esempio di separazione deterministica tra complessità classica e quantistica, e concluderemo con l'algoritmo di ricerca di Grover, di particolare interesse per le sue applicazioni in problemi di ricerca non strutturata.

5.1 Parallelismo Quantistico

Il termine "parallelismo quantistico" descrive la capacità nativa di un computer quantistico di valutare una funzione $f(x)$ per molteplici valori di input x simultaneamente, impiegando un singolo passaggio attraverso il circuito che implementa la funzione [6]. A differenza del calcolo parallelo classico, dove l'esecuzione simultanea richiede la duplicazione delle risorse hardware (più processori), nel dominio quantistico questo fenomeno emerge dalla linearità degli operatori applicati a stati di sovrapposizione.

Questa potenza computazionale presenta un vincolo architetturale critico: l'informazione calcolata in parallelo non è direttamente accessibile tramite le operazioni di I/O standard.

5.1.1 Definizione dell'Oracolo Quantistico

Consideriamo una funzione classica $f : \{0,1\}^n \to \{0,1\}$ che mappa una stringa di n bit in un singolo bit. In un contesto classico, valutare questa funzione per tutti i possibili 2^n input richiederebbe 2^n esecuzioni sequenziali o l'utilizzo di 2^n processori in parallelo.

Per implementare questa funzione in un ambiente quantistico, dobbiamo rispettare il vincolo di reversibilità imposto dalla natura unitaria degli operatori (Cap. 2). Poiché il calcolo di $f(x)$ potrebbe non essere invertibile (ad esempio se f non è biunivoca), non possiamo semplicemente trasformare $|x\rangle$ in $|f(x)\rangle$. È necessario utilizzare due registri: un data register di n qubit per l'input e un target register (o ancilla) di 1 qubit per l'output [5].

Definiamo l'operatore unitario U_f, spesso denominato "oracolo quantistico", che agisce sui due registri nel seguente modo:

$$U_f \, |x\rangle \, |y\rangle = |x\rangle \, |y \oplus f(x)\rangle \tag{5.1}$$

dove $|x\rangle$ rappresenta lo stato del registro di input, $|y\rangle$ lo stato del registro target e $\oplus$ denota l'addizione modulo 2 (corrispondente alla porta logica classica XOR) [6]. Questa trasformazione è unitaria e quindi fisicamente realizzabile, poiché U_f è l'inversa di se stessa ($U_f^\dagger U_f = I$).

5.1.2 Valutazione Simultanea

Il vantaggio computazionale si concretizza combinando l'oracolo U_f con la trasformata di Walsh-Hadamard, una routine standard per l'inizializzazione

dei registri in sovrapposizione equiprobabile. Il flusso di esecuzione algoritmico si sviluppa come segue:

1. **Inizializzazione dello Stato:** Si prepara il registro dati nello stato $|0\rangle^{\otimes n}$ e il target nello stato $|0\rangle$, ottenendo $|\psi_0\rangle = |0\rangle^{\otimes n} |0\rangle$.

2. **Generazione della Sovrapposizione:** Applicando la trasformata $H^{\otimes n}$ al registro di input, il sistema evolve in una sovrapposizione di tutti i 2^n possibili input. Dal punto di vista della gestione della memoria, il registro non contiene più un singolo valore, ma un vettore di stato che rappresenta simultaneamente tutti gli interi da 0 a $2^n - 1$:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |0\rangle \tag{5.2}$$

3. **Esecuzione Parallela:** L'applicazione dell'oracolo U_f allo stato $|\psi_1\rangle$ sfrutta la linearità della meccanica quantistica per processare tutti i termini della sommatoria in un unico step di clock del processore quantistico:

$$|\psi_2\rangle = U_f |\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle \tag{5.3}$$

   L'equazione (5.3) descrive uno stato che contiene tutte le coppie input-output $(x, f(x))$. Qui risiede il nodo cruciale per la progettazione del software quantistico: sebbene il sistema contenga l'informazione completa della tabella di verità della funzione f, tale informazione è codificata nelle ampiezze di probabilità e non è direttamente leggibile.

Un tentativo di lettura diretta dello stato $|\psi_2\rangle$ (misurazione nella base computazionale) causerebbe il collasso della funzione d'onda, restituendo una sola coppia $|x\rangle |f(x)\rangle$ casuale [6]. Questo comporta che il parallelismo quantistico non può essere utilizzato per sostituire il calcolo parallelo classico in compiti di tabulazione o calcolo distribuito massivo.

La sfida ingegneristica si sposta quindi dal calcolo dei valori all'estrazione di proprietà globali. Lo stato $|\psi_2\rangle$ non deve essere considerato come il risultato finale, ma come uno stato intermedio contenente correlazioni quantistiche (entanglement) tra il registro di input e quello di output. Per sfruttare questa risorsa, l'algoritmo deve proseguire con ulteriori operatori di interferenza, come l'operatore di diffusione di Grover, che permettano di estrarre proprietà relazionali della funzione $f(x)$ — come la sua periodicità o il bilanciamento — piuttosto che i singoli valori puntuali [5].

5.2 Algoritmo di Deutsch-Jozsa

L'algoritmo di Deutsch-Jozsa, proposto originariamente nel 1992 e successivamente migliorato, rappresenta uno dei primi esempi concreti di vantaggio computazionale quantistico deterministico rispetto al calcolo classico [6]. Sebbene il problema risolto sia di natura oracolare e di limitata utilità pratica diretta, esso costituisce un proof-of-concept fondamentale: dimostra come il parallelismo quantistico e l'interferenza possano essere orchestrati per estrarre una proprietà globale di una funzione con una singola valutazione, un compito che richiederebbe risorse esponenziali nel caso classico peggiore.

5.2.1 Definizione del Problema e Complessità

Il problema è definito nel contesto delle cosiddette Promise Problems. Sia data una funzione booleana $f : \{0,1\}^n \to \{0,1\}$ implementata come un oracolo (o black-box). Viene garantito a priori (la "promessa") che la funzione appartenga a una delle seguenti due classi [4]:

- **Costante:** La funzione restituisce lo stesso valore per ogni input (o sempre 0, o sempre 1).
- **Bilanciata:** La funzione restituisce 0 per esattamente la metà degli input e 1 per l'altra metà.

L'obiettivo è determinare a quale classe appartiene la funzione f minimizzando il numero di chiamate all'oracolo (query complexity).

Analisi Classica Per risolvere il problema con certezza assoluta nel caso peggiore, è necessario valutare la funzione su un numero di input sufficiente a escludere una delle due opzioni. Poiché ci sono 2^n possibili input, nel caso peggiore una funzione bilanciata potrebbe mimare una funzione costante per le prime 2^{n-1} valutazioni. Pertanto, sono necessarie $2^{n-1} + 1$ interrogazioni all'oracolo [6]. La complessità classica scala quindi esponenzialmente con il numero di bit n.

Analisi Quantistica Risolve il problema con certezza (determinismo) effettuando una singola chiamata all'oracolo quantistico U_f, indipendentemente dalla dimensione n dell'input [11]. Questo risultato evidenzia una separazione esponenziale tra le classi di complessità nel contesto della query complexity.

5.2.2 L'Architettura del Circuito

L'algoritmo utilizza due registri: un registro di query (o data register) composto da n qubit inizializzati a $|0\rangle$, e un registro ausiliario (ancilla) di un singolo qubit inizializzato a $|1\rangle$ [4]. Il circuito si sviluppa in tre fasi logiche: inizializzazione in sovrapposizione, interrogazione dell'oracolo tramite phase kickback, e interferenza finale.

Fase 1: Creazione della Sovrapposizione Si applica una porta di Hadamard a tutti i qubit. Il registro dati evolve in una sovrapposizione equiprobabile di tutti gli stati della base computazionale, mentre l'ancilla viene portata nello stato $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$:

$$|\psi_1\rangle = (H^{\otimes n} |0\rangle^{\otimes n}) \otimes (H |1\rangle) = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{5.4}$$

Questa configurazione è essenziale per sfruttare il parallelismo quantistico descritto nella sezione precedente [1].

Fase 2: Valutazione dell'Oracolo e Phase Kickback L'oracolo quantistico U_f applica la trasformazione $|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$. Tuttavia, grazie allo stato specifico dell'ancilla $|-\rangle$, l'effetto dell'oracolo non è quello di modificare il valore del qubit target (bit-flip), ma di iniettare una fase relativa nel registro di controllo [4]. Poiché $f(x)$ può essere solo 0 o 1, l'azione su $|-\rangle$ è:

$$U_f \left(|x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = (-1)^{f(x)} |x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{5.5}$$

Questo meccanismo, noto come phase kickback, permette di codificare l'informazione della funzione $f(x)$ direttamente nelle ampiezze di probabilità del registro dati, lasciando l'ancilla inalterata. Lo stato del sistema diventa [7]:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \otimes |-\rangle \tag{5.6}$$

Fase 3: Interferenza e Misura A questo punto, l'informazione è presente ma non accessibile direttamente tramite misura nella base computazionale. È necessario applicare nuovamente una trasformata di Hadamard $H^{\otimes n}$ al registro dati per causare interferenza tra le ampiezze. L'azione di $H^{\otimes n}$ su uno stato della base $|x\rangle$ è definita come $\sum_z (-1)^{x \cdot z} |z\rangle$ (dove $x \cdot z$ è il prodotto scalare bit a bit modulo 2) [7]. Lo stato finale del registro dati è:

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{z\in\{0,1\}^n} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot z + f(x)} |z\rangle \tag{5.7}$$

L'operazione finale consiste nella misurazione del registro dati. L'analisi si concentra sull'ampiezza di probabilità associata allo stato $|0\rangle^{\otimes n}$ (ossia il termine dove $z = 00\ldots0$). Poiché $x \cdot 0 = 0$, l'ampiezza per lo stato $|0\rangle^{\otimes n}$ è:

$$\alpha_0 = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} \tag{5.8}$$

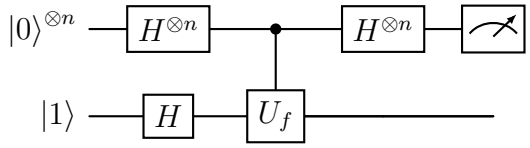

Figura 5.1: Schema circuitale dell'algoritmo di Deutsch-Jozsa. Il registro superiore (query) viene inizializzato in una sovrapposizione uniforme. L'oracolo U_f introduce il *phase kickback* grazie allo stato $|-\rangle$ del registro inferiore (ancilla).

5.2.3 Analisi dei Risultati

Analizziamo i due casi possibili garantiti dalla promessa iniziale:

- **Caso f Costante:** Se $f(x) = c$ (costante), allora $(-1)^{f(x)}$ è sempre 1 o sempre -1. La sommatoria diventa $\pm\sum_x 1 = \pm 2^n$. Di conseguenza, l'ampiezza $\alpha_0 = \pm 1$. La probabilità di misurare lo stato $|0\rangle^{\otimes n}$ è $|\pm 1|^2 = 1$. Tutta la probabilità collassa sullo stato zero (interferenza costruttiva) [5].

- **Caso f Bilanciata:** Se la funzione è bilanciata, $(-1)^{f(x)}$ sarà $+1$ per metà degli input e -1 per l'altra metà. La sommatoria $\sum_x (-1)^{f(x)}$ sarà quindi esattamente 0. L'ampiezza $\alpha_0 = 0$. La probabilità di misurare lo stato $|0\rangle^{\otimes n}$ è 0 (interferenza distruttiva totale) [7].

Conclusione Operativa L'algoritmo restituisce una risposta deterministica basata su una singola osservazione del registro dati:

- Se il risultato della misura è $00\ldots0 \implies f$ è Costante.

- Se il risultato della misura è un qualsiasi altro valore $\implies f$ è Bilanciata.

L'algoritmo di Deutsch-Jozsa introduce un pattern architetturale ricorrente: preparazione della sovrapposizione → codifica dell'informazione nella fase (oracolo) → interferenza per estrarre l'informazione nella base di misura [11]. Inoltre, evidenzia come il vantaggio quantistico risieda non nel calcolare più velocemente i singoli valori della funzione, ma nel determinare proprietà globali della funzione stessa (in questo caso, la costanza o il bilanciamento) sfruttando la struttura dello spazio di Hilbert.

5.3 Algoritmo di Grover

Mentre l'algoritmo di Deutsch-Jozsa offre una separazione esponenziale tra le complessità classica e quantistica in un contesto oracolare specifico, l'algoritmo di ricerca di Grover, introdotto nel 1996, fornisce un'accelerazione quadratica per una classe di problemi molto più ampia e applicabile: la ricerca in un database non strutturato [2]. Questo algoritmo è fondamentale perché modella problemi di inversione di funzione, ottimizzazione e soddisfacibilità di vincoli (come il problema SAT) [3].

5.3.1 Definizione del Problema

Il problema della ricerca non strutturata può essere formalizzato come segue. Sia dato uno spazio di ricerca di $N = 2^n$ elementi, indicizzati da $x \in \{0,1\}^n$. Si supponga di avere accesso a una funzione booleana $f(x)$ che restituisce 1 se x è l'elemento cercato (la soluzione w, winner) e 0 altrimenti:

$$f(x) = \begin{cases} 1 & \text{se } x = w \\ 0 & \text{se } x \neq w \end{cases} \tag{5.9}$$

In un contesto classico, trovare w richiede nel caso peggiore N valutazioni di f e in media $N/2$ valutazioni, portando a una complessità $O(N)$ [6]. L'algoritmo di Grover permette di trovare w con alta probabilità in $O(\sqrt{N})$ passi, dimostrando un vantaggio polinomiale. È stato dimostrato che questo limite è ottimale: nessun algoritmo quantistico può risolvere questo problema in meno di $\Omega(\sqrt{N})$ chiamate all'oracolo [1].

5.3.2 Algoritmo

L'esecuzione dell'algoritmo di Grover può essere schematizzata in tre fasi sequenziali distinte, che trasformano lo stato del registro quantistico dall'inizia-

lizzazione alla lettura finale, sfruttando l'interferenza costruttiva per amplificare la soluzione.

1. **Inizializzazione** Si predispone un registro quantistico Q composto da n qubit nello stato fondamentale $|0\rangle^{\otimes n}$. Successivamente, si applica una porta di Hadamard H a ciascun qubit del registro. Questa operazione genera una sovrapposizione uniforme di tutti i $N = 2^n$ stati della base computazionale [6]:

$$|\psi_0\rangle = H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \tag{5.10}$$

2. **Iterazione (Operatore di Grover)** Si applica per un determinato numero di volte t l'operatore unitario G al registro Q. L'operazione per una singola iterazione è definita come [1]:

$$G = H^{\otimes n} Z_{OR} H^{\otimes n} Z_f \tag{5.11}$$

3. **Misurazione** Al termine delle iterazioni, si effettua una misurazione dei qubit del registro Q rispetto alla base computazionale standard. L'output restituito è una stringa binaria che, con alta probabilità (dipendente dal numero di iterazioni t), corrisponde alla soluzione del problema di ricerca [2].

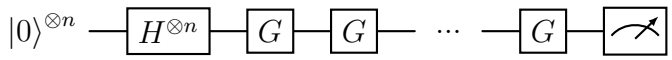

Figura 5.2: Rappresentazione ad alto livello dell'algoritmo di Grover. Dopo l'inizializzazione tramite Hadamard, l'Operatore di Grover G (composto da Oracolo e Diffusore) viene applicato $k \approx \frac{\pi}{4}\sqrt{N}$ volte per amplificare la probabilità della soluzione.

L'Oracolo Quantistico di Fase Z_f L'elemento centrale dell'algoritmo è l'operatore unitario Z_f, noto come oracolo di fase. L'oracolo rappresenta una subroutine "black-box" che astrae la complessità del criterio di ricerca, permettendo all'algoritmo di interagire con i dati senza dover conoscere a priori la struttura della soluzione [6].

Matematicamente, Z_f agisce sugli stati della base computazionale $|x\rangle$ applicando un cambiamento di fase condizionale basato sul valore di una funzione

booleana $f(x)$, che restituisce 1 se x è la soluzione cercata e 0 altrimenti. L'azione dell'operatore è definita dalla seguente relazione:

$$Z_f |x\rangle = (-1)^{f(x)} |x\rangle \tag{5.12}$$

Analizzando l'equazione, si distinguono due casi operativi che determinano l'evoluzione dello stato nel registro quantistico:

- **Caso Soluzione ($x \in A_1$):** Se x è la soluzione cercata (o una delle soluzioni), allora $f(x) = 1$. L'oracolo inverte la fase dello stato quantistico, poiché $(-1)^1 = -1$. Il coefficiente dell'ampiezza di probabilità cambia segno:
$$|x\rangle \xrightarrow{Z_f} -|x\rangle \tag{5.13}$$

- **Caso Non-Soluzione ($x \in A_0$):** Se x non è la soluzione, allora $f(x) = 0$. L'oracolo agisce come l'operatore identità, lasciando lo stato invariato, poiché $(-1)^0 = 1$:
$$|x\rangle \xrightarrow{Z_f} |x\rangle \tag{5.14}$$

L'Operatore di Riflessione Z_{OR} Il secondo componente fondamentale dell'iterazione di Grover è l'operatore Z_{OR}, spesso indicato in letteratura come operatore di spostamento di fase condizionale o riflessione rispetto allo stato fondamentale. Matematicamente, Z_{OR} è definito come una trasformazione lineare che agisce sull'intero registro di n qubit:

$$Z_{OR} = 2 |0\rangle^{\otimes n} \langle 0|^{\otimes n} - I \tag{5.15}$$

dove:

- $|0\rangle^{\otimes n} \langle 0|^{\otimes n}$ è il proiettore sullo stato fondamentale $|0\rangle^{\otimes n}$ (ossia lo stato in cui tutti i qubit sono a 0).
- I è l'operatore identità nello spazio di Hilbert di dimensione $N = 2^n$.

Questo operatore è costruito per implementare una specifica operazione geometrica: una riflessione del vettore di stato rispetto all'iperpiano ortogonale allo stato $|0\rangle^{\otimes n}$ [6].

Analisi Spettrale e Derivazione Per comprendere la dinamica indotta da Z_{OR}, analizziamo la sua azione sugli stati della base computazionale. Poiché

l'operatore è definito in termini di proiettori, possiamo distinguere due casi fondamentali basati sull'ortogonalità degli stati.

1. **Azione sullo stato fondamentale $|0\rangle^{\otimes n}$:** Applicando l'operatore allo stato zero, il termine di proiezione preserva il vettore:

$$\begin{aligned} Z_{OR}\,|0\rangle^{\otimes n} &= (2\,|0\rangle^{\otimes n}\,\langle 0|^{\otimes n} - I)\,|0\rangle^{\otimes n} \\ &= 2\,|0\rangle^{\otimes n}\,(\langle 0|0\rangle^{\otimes n}) - I\,|0\rangle^{\otimes n} \\ &= 2\,|0\rangle^{\otimes n}\,(1) - |0\rangle^{\otimes n} = |0\rangle^{\otimes n} \end{aligned} \tag{5.16}$$

Il risultato indica che lo stato $|0\rangle^{\otimes n}$ è un autostato dell'operatore con autovalore $+1$. In termini fisici, la sua fase rimane invariata [11].

2. **Azione sugli stati ortogonali $|x\rangle$ (con $x \neq 0$):** Consideriamo un qualsiasi stato della base $|x\rangle$ diverso dallo stato fondamentale. Per le proprietà della base computazionale, questi stati sono ortogonali a $|0\rangle^{\otimes n}$, ovvero il loro prodotto interno $\langle 0^{\otimes n}|x\rangle = 0$. L'azione dell'operatore diventa:

$$\begin{aligned} Z_{OR}\,|x\rangle &= (2\,|0\rangle^{\otimes n}\,\langle 0|^{\otimes n} - I)\,|x\rangle \\ &= 2\,|0\rangle^{\otimes n}\,(\langle 0^{\otimes n}|x\rangle) - I\,|x\rangle \\ &= 2\,|0\rangle^{\otimes n}\,(0) - |x\rangle = -\,|x\rangle \end{aligned} \tag{5.17}$$

Il risultato mostra che tutti gli stati del sottospazio ortogonale a $|0\rangle^{\otimes n}$ subiscono un'inversione di fase (autovalore -1) [1].

Conclusione e Interpretazione Geometrica L'operatore Z_{OR} definisce quindi una riflessione nello spazio di Hilbert che discrimina selettivamente lo stato fondamentale:

- Mantiene intatta la componente del vettore di stato parallela a $|0\rangle^{\otimes n}$.
- Inverte il segno di tutte le componenti che giacciono nel sottospazio ortogonale.

Questa operazione è cruciale perché, quando combinata (sandwich) tra due trasformate di Hadamard $H^{\otimes n}$, dà luogo all'operatore di diffusione $D = H^{\otimes n} Z_{OR} H^{\otimes n}$, che esegue l'inversione rispetto alla media delle ampiezze, il meccanismo fisico responsabile dell'amplificazione della probabilità della soluzione [6].

Definizione degli Insiemi e dei Vettori di Base Per analizzare la dinamica dell'algoritmo di Grover, è necessario partizionare lo spazio di ricerca

$\Sigma^n = \{0,1\}^n$ in due sottoinsiemi disgiunti, basati sulla risposta della funzione oracolo $f(x)$. Definiamo formalmente gli insiemi delle non-soluzioni (A_0) e delle soluzioni (A_1) come segue:

$$A_0 = \{x \in \Sigma^n : f(x) = 0\}, \quad A_1 = \{x \in \Sigma^n : f(x) = 1\} \tag{5.18}$$

A_1 rappresenta l'insieme degli stati che soddisfano la condizione di ricerca (i target o winners), mentre A_0 rappresenta l'insieme degli stati da scartare. Questi insiemi costituiscono una bipartizione completa dello spazio di ricerca computazionale, soddisfacendo le seguenti proprietà insiemistiche:

$$A_0 \cap A_1 = \emptyset \quad \text{e} \quad A_0 \cup A_1 = \Sigma^n \tag{5.19}$$

Sfruttando questa partizione, possiamo definire due stati quantistici normalizzati che rappresentano la sovrapposizione uniforme (equiprobabile) rispettivamente su tutte le non-soluzioni e su tutte le soluzioni. Questi vettori sono definiti come:

$$|A_0\rangle = \frac{1}{\sqrt{|A_0|}} \sum_{x \in A_0} |x\rangle, \quad |A_1\rangle = \frac{1}{\sqrt{|A_1|}} \sum_{x \in A_1} |x\rangle \tag{5.20}$$

dove $|A_0|$ e $|A_1|$ indicano la cardinalità degli insiemi (spesso indicata in letteratura rispettivamente come $N - M$ e M, dove $N = 2^n$) [6, 1].

Decomposizione dello Stato Uniforme L'obiettivo è esprimere $|u\rangle$ come combinazione lineare dei vettori normalizzati $|A_0\rangle$ e $|A_1\rangle$.

1. **Definizione dello stato iniziale.** Partiamo dalla definizione dello stato di sovrapposizione uniforme ottenuto applicando la trasformata di Hadamard allo stato fondamentale. Esso rappresenta la somma di tutte le possibili stringhe x nello spazio di ricerca Σ^n, normalizzata dal fattore $\sqrt{N}$:

$$|u\rangle = \frac{1}{\sqrt{N}} \sum_{x \in \Sigma^n} |x\rangle \tag{5.21}$$

2. **Partizione della sommatoria.** Poiché l'insieme di tutte le stringhe Σ^n è partizionato in due insiemi disgiunti A_0 (l'insieme delle non-soluzioni) e A_1 (l'insieme delle soluzioni), possiamo spezzare la sommatoria in due componenti distinte [2]:

$$|u\rangle = \frac{1}{\sqrt{N}} \left(\sum_{x \in A_0} |x\rangle + \sum_{x \in A_1} |x\rangle \right) \tag{5.22}$$

Espandendo il fattore di normalizzazione, otteniamo:

$$|u\rangle = \frac{1}{\sqrt{N}} \sum_{x \in A_0} |x\rangle + \frac{1}{\sqrt{N}} \sum_{x \in A_1} |x\rangle \tag{5.23}$$

3. **Utilizzo dei vettori normalizzati di base.** Ricordiamo le definizioni dei vettori normalizzati $|A_0\rangle$ e $|A_1\rangle$ introdotte precedentemente [6]:

$$|A_0\rangle = \frac{1}{\sqrt{|A_0|}} \sum_{x \in A_0} |x\rangle \implies \sum_{x \in A_0} |x\rangle = \sqrt{|A_0|}\,|A_0\rangle \tag{5.24}$$

$$|A_1\rangle = \frac{1}{\sqrt{|A_1|}} \sum_{x \in A_1} |x\rangle \implies \sum_{x \in A_1} |x\rangle = \sqrt{|A_1|}\,|A_1\rangle \tag{5.25}$$

4. **Sostituzione.** Sostituendo le sommatorie nell'equazione di $|u\rangle$ con le espressioni equivalenti in termini di $|A_0\rangle$ e $|A_1\rangle$, otteniamo:

$$|u\rangle = \frac{1}{\sqrt{N}}(\sqrt{|A_0|}\,|A_0\rangle) + \frac{1}{\sqrt{N}}(\sqrt{|A_1|}\,|A_1\rangle) \tag{5.26}$$

5. **Risultato Finale.** Raggruppando i termini, giungiamo alla scomposizione desiderata:

$$|u\rangle = \sqrt{\frac{|A_0|}{N}}\,|A_0\rangle + \sqrt{\frac{|A_1|}{N}}\,|A_1\rangle \tag{5.27}$$

Questa equazione è fondamentale perché dimostra che lo stato iniziale $|u\rangle$, pur essendo definito in uno spazio di Hilbert di dimensione 2^n, giace interamente nel piano bidimensionale (sottospazio) generato dai vettori ortonormali $|A_0\rangle$ e $|A_1\rangle$ [1].

Derivazione dell'Operatore di Diffusione L'operatore di diffusione D (o U_s), responsabile dell'amplificazione delle ampiezze, non è una primitiva fisica diretta nella maggior parte delle architetture hardware. Esso viene invece sintetizzato algoritmicamente applicando l'operatore di riflessione Z_{OR} in una base ruotata. La derivazione formale segue applicando le porte di Hadamard $H^{\otimes n}$ prima e dopo l'operatore Z_{OR}. Partendo dalla definizione $Z_{OR} = 2\,|0\rangle^{\otimes n}\,\langle 0|^{\otimes n} - I$, otteniamo la seguente trasformazione unitaria:

$$D = H^{\otimes n} Z_{OR} H^{\otimes n} = H^{\otimes n}(2\,|0\rangle^{\otimes n}\,\langle 0|^{\otimes n} - I)H^{\otimes n} \tag{5.28}$$

Sfruttando la linearità dell'algebra quantistica, possiamo distribuire gli

operatori $H^{\otimes n}$ all'interno della parentesi:

$$D = 2(H^{\otimes n} |0\rangle^{\otimes n})(\langle 0|^{\otimes n} H^{\otimes n}) - H^{\otimes n} I H^{\otimes n} \tag{5.29}$$

Analizziamo i termini risultanti passaggio per passaggio: 1. **Trasformazione del Proiettore:** Sappiamo che l'operatore di Hadamard applicato allo stato fondamentale genera la sovrapposizione uniforme $|u\rangle$. Poiché H è un operatore hermitiano ($H = H^\dagger$), la sua azione sul bra è simmetrica a quella sul ket:

$$H^{\otimes n} |0\rangle^{\otimes n} = |u\rangle \quad \text{e} \quad \langle 0|^{\otimes n} H^{\otimes n} = (H^{\otimes n} |0\rangle^{\otimes n})^\dagger = \langle u| \tag{5.30}$$

Di conseguenza, il termine $2H^{\otimes n} |0\rangle \langle 0| H^{\otimes n}$ diventa $2 |u\rangle \langle u|$.

2. **Trasformazione dell'Identità:** L'operatore di Hadamard è unitario e involutorio (è l'inverso di se stesso), ovvero $H^2 = I$. Pertanto, coniugare la matrice identità con H restituisce l'identità:

$$H^{\otimes n} I H^{\otimes n} = (H^{\otimes n})(H^{\otimes n}) = I \tag{5.31}$$

Sostituendo questi risultati nell'equazione originale, otteniamo la forma finale dell'operatore di diffusione:

$$D = 2 |u\rangle \langle u| - I \tag{5.32}$$

Calcolo dell'Azione di G su $|A_0\rangle$ Iniziamo analizzando l'evoluzione del vettore $|A_0\rangle$, che rappresenta la sovrapposizione uniforme di tutte le non-soluzioni. Ricordiamo le definizioni chiave stabilite nelle sezioni precedenti:

- Operatore di Grover: $G = (2 |u\rangle \langle u| - I)Z_f$.
- Azione dell'Oracolo (Z_f): Poiché per ogni stato $x \in A_0$ il valore della funzione $f(x)$ è 0, l'oracolo non introduce alcun cambiamento di fase relativa. Di conseguenza, l'azione globale sul vettore di sovrapposizione è l'identità:

$$Z_f |A_0\rangle = |A_0\rangle \tag{5.33}$$

- Proiezione dello stato uniforme: Dalla decomposizione dello stato $|u\rangle$, sappiamo che il prodotto scalare tra lo stato uniforme e le non-soluzioni è dato da:

$$\langle u|A_0\rangle = \sqrt{\frac{|A_0|}{N}} \tag{5.34}$$

Derivazione Passo-Passo. Calcoliamo l'applicazione dell'operatore G allo stato $|A_0\rangle$:

$$G\,|A_0\rangle = (2\,|u\rangle\,\langle u| - I)Z_f\,|A_0\rangle \tag{5.35}$$

1. Applicazione dell'Oracolo: Sfruttando la proprietà $Z_f\,|A_0\rangle = |A_0\rangle$, l'espressione si semplifica in una riflessione rispetto alla media dello stato non marcato [2]:

$$G\,|A_0\rangle = (2\,|u\rangle\,\langle u| - I)\,|A_0\rangle = 2\,|u\rangle\,\langle u|A_0\rangle - |A_0\rangle \tag{5.36}$$

2. Sostituzione del Prodotto Scalare: Sostituiamo il valore del prodotto interno $\langle u|A_0\rangle = \sqrt{|A_0|/N}$:

$$G\,|A_0\rangle = 2\,|u\rangle\left(\sqrt{\frac{|A_0|}{N}}\right) - |A_0\rangle \tag{5.37}$$

3. Espansione rispetto alla base $\{|A_0\rangle\,, |A_1\rangle\}$: Per ottenere il risultato finale in termini della base ortonormale del sottospazio, sostituiamo $|u\rangle$ con la sua decomposizione $|u\rangle = \sqrt{\frac{|A_0|}{N}}\,|A_0\rangle + \sqrt{\frac{|A_1|}{N}}\,|A_1\rangle$:

$$G\,|A_0\rangle = 2\left(\sqrt{\frac{|A_0|}{N}}\,|A_0\rangle + \sqrt{\frac{|A_1|}{N}}\,|A_1\rangle\right)\frac{\sqrt{|A_0|}}{\sqrt{N}} - |A_0\rangle \tag{5.38}$$

4. Semplificazione Algebrica: Eseguiamo le moltiplicazioni distribuendo i termini:

$$\begin{aligned} G\,|A_0\rangle &= 2\left(\frac{|A_0|}{N}\,|A_0\rangle + \frac{\sqrt{|A_0||A_1|}}{N}\,|A_1\rangle\right) - |A_0\rangle \\ &= \left(\frac{2|A_0|}{N} - 1\right)|A_0\rangle + \frac{2\sqrt{|A_0||A_1|}}{N}\,|A_1\rangle \end{aligned} \tag{5.39}$$

Risultato e Interpretazione. L'equazione finale descrive come lo stato delle non-soluzioni evolve dopo una singola iterazione:

$$G\,|A_0\rangle = \left(1 - \frac{2|A_1|}{N}\right)|A_0\rangle + \left(\frac{2\sqrt{|A_0||A_1|}}{N}\right)|A_1\rangle \tag{5.40}$$

dove abbiamo usato l'identità $|A_0| = N - |A_1|$ per riscrivere il coefficiente di $|A_0\rangle$. Da un punto di vista ingegneristico, questo risultato dimostra che l'operatore di Grover introduce una componente non nulla lungo la direzione

$|A_1\rangle$ (le soluzioni). Anche partendo da uno stato di pure non-soluzioni, l'iterazione "ruota" il vettore verso la soluzione desiderata, trasferendo ampiezza di probabilità nel componente $|A_1\rangle$ [6].

Risultato dell'Azione di G su $|A_1\rangle$ Analogamente a quanto mostrato per lo stato delle non-soluzioni, è possibile determinare l'evoluzione dello stato $|A_1\rangle$ (la sovrapposizione uniforme delle soluzioni) sotto l'azione dell'operatore di Grover. Tenendo conto che l'oracolo Z_f inverte la fase di $|A_1\rangle$ (poiché $f(x) = 1$ per $x \in A_1$), e applicando successivamente l'operatore di diffusione, si ottiene il seguente risultato diretto [6]:

$$G\,|A_1\rangle = -\frac{2\sqrt{|A_0||A_1|}}{N}\,|A_0\rangle + \left(1 - \frac{2|A_0|}{N}\right)|A_1\rangle \tag{5.41}$$

Interpretazione Geometrica Complessiva. Combinando questo risultato con quello ottenuto nella sezione precedente, emerge chiaramente la natura geometrica dell'algoritmo. L'operatore G agisce come una matrice di rotazione nello spazio bidimensionale generato dalla base $\{|A_0\rangle, |A_1\rangle\}$ [1]. Definendo l'angolo θ tale che $\sin(\theta) = \sqrt{|A_1|/N}$, le equazioni di trasformazione possono essere riscritte in una forma compatta che evidenzia la rotazione del vettore di stato di un angolo 2θ ad ogni iterazione:

$$G\begin{pmatrix}|A_0\rangle\\|A_1\rangle\end{pmatrix} = \begin{pmatrix}\cos(2\theta) & -\sin(2\theta)\\ \sin(2\theta) & \cos(2\theta)\end{pmatrix}\begin{pmatrix}|A_0\rangle\\|A_1\rangle\end{pmatrix} \tag{5.42}$$

Questo, come descritto in precedenza, garantisce che l'informazione quantistica non si disperda in altri sottospazi dell'enorme spazio di Hilbert (dimensione 2^n), ma rimanga confinata nel piano definito dalle condizioni iniziali e dall'oracolo, permettendo un controllo preciso dell'amplificazione dell'ampiezza desiderata [2].

Interpretazione Geometrica e Derivazione del Numero di Iterazioni Sulla base dei risultati ottenuti per l'azione di G sui vettori di base $|A_0\rangle$ e $|A_1\rangle$, possiamo ora riscrivere l'operatore di Grover come una trasformazione unitaria dipendente dall'angolo θ. Ricordando che lo stato iniziale è $|u\rangle = \cos(\theta)\,|A_0\rangle + \sin(\theta)\,|A_1\rangle$ con $\sin(\theta) = \sqrt{|A_1|/N}$, l'azione dell'operatore di Grover G può essere espressa come una rotazione rigida nel piano bidimensionale generato da $\{|A_0\rangle, |A_1\rangle\}$.

Rappresentazione Matriciale di G. L'operatore G agisce sul sottospazio bidimensionale come una matrice di rotazione $R(2\theta)$. Nella base ortonormale $\{|A_0\rangle, |A_1\rangle\}$, la matrice associata a G è [1]:

$$G = \begin{pmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{pmatrix} \tag{5.43}$$

L'applicazione di G allo stato del sistema provoca una rotazione del vettore di stato in senso antiorario di un angolo 2θ verso l'asse $|A_1\rangle$ (la soluzione).

Evoluzione dello Stato dopo t Iterazioni. Poiché ogni applicazione di G corrisponde a una rotazione di 2θ, l'applicazione di t iterazioni successive corrisponde a una rotazione totale di $2t\theta$. Partendo dallo stato iniziale $|u\rangle$, che forma già un angolo θ con l'asse $|A_0\rangle$, lo stato del sistema dopo t passi, denotato come $|\psi_t\rangle = G^t |u\rangle$, sarà caratterizzato da un angolo totale $(2t+1)\theta$:

$$|\psi_t\rangle = \cos((2t+1)\theta)\,|A_0\rangle + \sin((2t+1)\theta)\,|A_1\rangle \tag{5.44}$$

Questa espressione chiusa descrive l'intera dinamica dell'algoritmo [6].

Calcolo del Numero Ottimale di Iterazioni. L'obiettivo dell'algoritmo è massimizzare la probabilità di misurare uno stato appartenente all'insieme delle soluzioni A_1. La probabilità di successo $P(t)$ dopo t iterazioni è data dal quadrato dell'ampiezza della componente lungo $|A_1\rangle$:

$$P(t) = |\langle A_1|\psi_t\rangle|^2 = \sin^2((2t+1)\theta) \tag{5.45}$$

Per massimizzare questa probabilità, l'argomento del seno deve essere il più vicino possibile a $\pi/2$ (ovvero 90 gradi), portando il vettore di stato a sovrapporsi quasi perfettamente con la soluzione. Imponiamo la condizione:

$$(2t+1)\theta \approx \frac{\pi}{2} \tag{5.46}$$

Risolvendo rispetto a t, otteniamo il numero ottimale di iterazioni t_{opt}:

$$t_{opt} \approx \frac{\pi}{4\theta} - \frac{1}{2} \tag{5.47}$$

Assumendo che il numero di soluzioni sia molto piccolo rispetto alla dimensione del database ($|A_1| \ll N$), l'angolo θ è piccolo, e possiamo approssimare $\theta \approx \sin(\theta) = \sqrt{|A_1|/N}$. Sostituendo questa approssimazione, otteniamo la formula

standard per la complessità di Grover:

$$t_{opt} \approx \frac{\pi}{4}\sqrt{\frac{N}{|A_1|}} \tag{5.48}$$

Il risultato deve essere arrotondato all'intero più vicino [1]. Il loop di controllo non deve basarsi su una condizione di convergenza asintotica (come in molti algoritmi classici iterativi), ma deve arrestarsi esattamente a t_{opt}. Se si prosegue l'esecuzione oltre t_{opt} (fenomeno di over-rotation), il vettore di stato continuerà a ruotare oltre l'asse $|A_1\rangle$, riducendo la probabilità di successo secondo la legge periodica $\sin^2((2t+1)\theta)$ [2]. Pertanto, la conoscenza (o la stima) della cardinalità $|A_1|$ è un prerequisito fondamentale per configurare correttamente il numero di cicli nel software di controllo quantistico.

5.4 Vantaggio Quadratico e Limiti del Quantum Advantage

L'analisi degli algoritmi di Deutsch-Jozsa e di Grover permette di delineare con precisione la natura del vantaggio computazionale offerto dal modello quantistico. Mentre il primo offre una separazione esponenziale tra le classi di complessità in un contesto oracolare ristretto, l'algoritmo di Grover fornisce uno speedup polinomiale (specificamente quadratico) per un problema di portata generale.

5.4.1 Ottimalità dell'Algoritmo: Il Limite Inferiore

Una domanda fondamentale per l'ingegneria degli algoritmi è se sia possibile progettare un algoritmo di ricerca quantistica ancora più veloce di quello di Grover, magari raggiungendo un tempo logaritmico $O(\log N)$ o costante. La risposta è negativa. È stato dimostrato formalmente (Bennett, Bernstein, Brassard e Vazirani, 1997) che l'algoritmo di Grover è ottimale.

Teorema del Lower Bound Nessun algoritmo quantistico può risolvere il problema della ricerca non strutturata in un database di dimensione N utilizzando meno di $\Omega(\sqrt{N})$ chiamate all'oracolo. La dimostrazione si basa sulla divergenza tra lo stato iniziale e lo stato finale: si osserva che una singola chiamata all'oracolo può ruotare il vettore di stato nello spazio di Hilbert di

un angolo massimo di $O(1/\sqrt{N})$. Poiché lo stato iniziale e lo stato soluzione sono ortogonali (distanza angolare $\approx \pi/2$), sono necessarie almeno $\Omega(\sqrt{N})$ iterazioni per colmare tale distanza [6].

Questo risultato stabilisce un limite invalicabile per il software quantistico: per problemi di ricerca black-box puri, il vantaggio quantistico è limitato alla radice quadrata.

5.4.2 Il Costo dell'Oracolo

La complessità $O(\sqrt{N})$ si riferisce al numero di query all'oracolo, assumendo che l'oracolo stesso sia efficiente. Se il problema consiste nel cercare un elemento in un database fisico (es. un elenco telefonico non ordinato memorizzato su disco), il tempo necessario per costruire l'oracolo o per inizializzare la sovrapposizione dei dati potrebbe essere lineare, $O(N)$, annullando il vantaggio quantistico [3]. L'algoritmo di Grover è quindi vantaggioso principalmente quando:

1. L'oracolo è una funzione calcolabile (es. verificare se una chiave crittografica decifra un messaggio), non una lookup table.

2. I dati sono già caricati in una memoria quantistica ad accesso casuale (QRAM), che permette di accedere a N celle di memoria in sovrapposizione con un costo polilogaritmico. Tuttavia, la realizzazione fisica di una QRAM robusta e scalabile rimane una delle sfide aperte più ardue per l'hardware quantistico [2].

In conclusione, mentre l'algoritmo di Deutsch-Jozsa funge da dimostrazione teorica del parallelismo, l'algoritmo di Grover definisce il benchmark prestazionale per una vasta classe di problemi reali, imponendo però rigorosi vincoli sull'architettura della memoria e sulla definizione della funzione di costo (oracolo) per garantire un effettivo quantum advantage.

Capitolo 6

Implementazione e Simulazione con Qiskit

Il presente capitolo è dedicato alla traduzione dei modelli algoritmici discussi, in particolare l'algoritmo di Grover, in artefatti software eseguibili. Utilizzeremo **Qiskit**, un Software Development Kit (SDK) open-source che rappresenta lo standard industriale per la programmazione quantistica [8]. L'approccio adottato segue il ciclo di vita tipico del software quantistico:

1. **Definizione del Circuito:** Costruzione logica del registro e delle porte tramite primitive Python.
2. **Transpilazione:** Ottimizzazione e mappatura del circuito logico sulla topologia fisica del backend target (processo analogo alla compilazione in linguaggio macchina).
3. **Simulazione:** Verifica della correttezza logica in un ambiente privo di rumore o con rumore simulato.
4. **Esecuzione:** Lancio del job su Hardware Quantistico Reale (QPU) tramite servizi cloud.

L'analisi dei risultati, presentata nella parte conclusiva del capitolo, metterà a confronto le metriche ideali ottenute in simulazione con quelle reali, evidenziando le limitazioni imposte dal rumore nei dispositivi NISQ (*Noisy Intermediate-Scale Quantum*).

6.1 Il Framework Qiskit

Qiskit è un framework modulare scritto in Python che permette di operare a diversi livelli di astrazione, dai singoli impulsi a microonde per il controllo dei qubit fisici fino ai moduli applicativi di alto livello. Per gli scopi di questa tesi, ci concentreremo su tre componenti architetturali fondamentali: il motore di simulazione (*Aer*), il servizio di runtime per l'accesso all'hardware (*IBM Runtime*) e il processo di compilazione (*Transpiler*).

6.1.1 Architettura dell'SDK

L'interazione con il framework avviene attraverso la definizione di oggetti `QuantumCircuit`, i quali incapsulano la logica sequenziale delle porte quantistiche. Le librerie utilizzate nel nostro caso di studio riflettono la moderna architettura basata sulle *Primitive*, un modello di esecuzione che astrae la complessità della gestione dei job hardware.

Le dipendenze software principali sono:

- **Qiskit Core (Terra):** Fornisce le strutture dati fondamentali per la definizione dei circuiti, dei registri quantistici e classici, e le utility di visualizzazione. Include il modulo `qiskit.transpiler`, essenziale per adattare il circuito astratto ai vincoli fisici della macchina (connettività dei qubit e gate set nativo).

- **Qiskit Aer (`qiskit_aer`):** È il componente dedicato alla simulazione ad alte prestazioni su hardware classico. In particolare, la classe `AerSimulator` permette di emulare il comportamento di una QPU ideale, eseguendo l'algebra lineare sottostante l'evoluzione degli stati quantistici.

- **Qiskit IBM Runtime (`qiskit_ibm_runtime`):** Rappresenta il driver per l'interazione con i servizi cloud di IBM Quantum. La classe `QiskitRuntimeService` gestisce l'autenticazione e la coda dei job, mentre `SamplerV2` costituisce l'implementazione della primitiva di campionamento. A differenza dei modelli di esecuzione precedenti, le primitive come il *Sampler* sono ottimizzate per gestire sessioni di esecuzione e applicare automaticamente tecniche di mitigazione degli errori (*Error Mitigation*) sui risultati grezzi.

6.1.2 Il Ruolo del Transpiler

Uno degli aspetti più critici nello sviluppo software per macchine NISQ è il divario tra il modello logico e quello fisico. Un algoritmo può richiedere l'applicazione di una porta CNOT tra due qubit logici arbitrari, q_i e q_j. Tuttavia, nell'hardware superconduttivo reale, i qubit sono disposti secondo una specifica topologia di accoppiamento (coupling map); se q_i e q_j non sono fisicamente connessi, l'operazione non può essere eseguita direttamente.

La funzione `transpile` è responsabile della risoluzione di questi vincoli. Essa riscrive il circuito logico in un circuito fisico equivalente, introducendo operazioni di SWAP per spostare gli stati quantistici su qubit adiacenti e decomponendo le porte logiche complesse (come la porta di Toffoli o CCX usata nell'algoritmo di Grover) nel set di gate nativi supportati dal backend (solitamente rotazioni a singolo qubit e CNOT o CZ). Questo processo introduce un *overhead* computazionale, aumentando la profondità del circuito e, conseguentemente, l'esposizione al rumore e alla decoerenza.

6.2 Caso di Studio: Implementazione dell'Algoritmo di Grover

In questa sezione presentiamo un'implementazione modulare dell'algoritmo di Grover per la ricerca non strutturata in uno spazio di dimensione $N = 2^n$.

Il codice sorgente completo, inclusivo degli script di configurazione dell'ambiente, è disponibile nel repository pubblico dedicato [9].

6.2.1 Definizione della Classe e Inizializzazione

La classe `GroverAlgorithm` (Listato 6.1) incapsula la logica di costruzione del circuito, parametrizzando la dimensione del registro e lo stato target $|w\rangle$.

```
class GroverAlgorithm:
    def __init__(self, n_qubits, target_state):
        self.n = n_qubits
        self.target = target_state

        # Validazione input
        if len(target_state) != n_qubits:
            raise ValueError("Lunghezza target non valida.")
```

Listing 6.1: Inizializzazione della classe GroverAlgorithm

6.2.2 Costruzione degli Operatori (Oracolo e Diffusore)

Prima di procedere all'analisi del codice, è necessario introdurre un concetto architetturale critico per la corretta implementazione degli oracoli: l'**Uncomputation** (o decomputazione). Nel design di circuiti quantistici, è frequente l'utilizzo di qubit ausiliari (spesso denominati *ancilla* o *scratch qubits*) per memorizzare risultati intermedi di operazioni aritmetiche o logiche reversibili necessarie a calcolare la funzione $f(x)$ [3]. Tuttavia, questi qubit ausiliari rimangono *entangled* con il registro principale dei dati. Se questi venissero lasciati nel loro stato modificato o venissero misurati/scartati, l'entanglement residuo agirebbe sull'intero sistema come una misurazione parziale, introducendo decoerenza e distruggendo l'interferenza costruttiva necessaria per il funzionamento di algoritmi come quello di Grover [6].

Per evitare questo fenomeno, si applica la tecnica dell'Uncomputation: subito dopo aver utilizzato il risultato intermedio per influenzare il target (es. tramite un Phase Kickback), si applicano le operazioni inverse a quelle utilizzate per il calcolo. Questo processo riporta i qubit di ancilla allo stato fondamentale $|0\rangle$ e, cosa più importante, li disaccoppia (disentangle) dal registro dati, preservando la coerenza del sistema [3, 10].

Nel codice seguente, questa tecnica è visibile nel blocco dell'oracolo, dove le porte logiche vengono ri-applicate per ripristinare lo stato.

L'oracolo e il diffusore sono implementati come metodi che restituiscono istruzioni quantistiche opache. Si noti l'uso di `to_instruction()` nel Listato 6.2: questa chiamata migliora la leggibilità del diagramma e permette al *Transpiler* di ottimizzare il blocco come unità logica.

```
1     def create_oracle(self):
2         qc = QuantumCircuit(self.n)
3         # 1. 'Avvolge' lo stato target per attivare il
    controllo su |11..1>
4         for i, bit in enumerate(reversed(self.target)):
5             if bit == '0':
6                 qc.x(i)
7         # 2. Phase Kickback tramite Multi-Controlled Toffoli (
    MCX)
8         qc.h(self.n - 1)
9         qc.mcx(list(range(self.n - 1)), self.n - 1)
10        qc.h(self.n - 1)
11        # 3. Uncomputation (ripristino dello stato)
12        for i, bit in enumerate(reversed(self.target)):
13            if bit == '0':
```

```
                qc.x(i)
        return qc.to_instruction(label="Oracle")

    def create_diffuser(self):
        qc = QuantumCircuit(self.n)
        # Applicazione trasformata H e inversione X
        qc.h(range(self.n))
        qc.x(range(self.n))
        # Multi-Controlled Z simulato
        qc.h(self.n - 1)
        qc.mcx(list(range(self.n - 1)), self.n - 1)
        qc.h(self.n - 1)
        # Uncomputation
        qc.x(range(self.n))
        qc.h(range(self.n))
        return qc.to_instruction(label="Diffuser")
```

Listing 6.2: Metodi per la creazione degli operatori quantistici

6.2.3 Assemblaggio del Circuito

Il metodo `build_circuit` (Listato 6.3) assembla le componenti seguendo la teoria: inizializzazione in sovrapposizione uniforme ($H^{\otimes n}$), ripetizione dell'iterazione di Grover per k volte, e misura finale.

```
    def build_circuit(self, iterations):
        qc = QuantumCircuit(self.n, self.n)

        # Fase 1: Inizializzazione (Superposition)
        qc.h(range(self.n))

        # Creazione degli operatori
        oracle = self.create_oracle()
        diffuser = self.create_diffuser()

        # Fase 2: Loop di Grover
        for _ in range(iterations):
            qc.append(oracle, range(self.n))
            qc.append(diffuser, range(self.n))

        # Fase 3: Misura
        qc.measure(range(self.n), range(self.n))
        return qc
```

Listing 6.3: Assemblaggio del circuito finale

6.3 Risultati Sperimentali e Analisi delle Prestazioni

L'implementazione è stata eseguita su un simulatore ideale privo di rumore (`aer_simulator`) e su un processore quantistico reale basato su tecnologia a superconduttori.

6.3.1 Configurazione del test

Il test è stato configurato con le seguenti specifiche:

- **Spazio di Ricerca:** $N = 8$ elementi, codificati su un registro di $n = 3$ qubit (q_0, q_1, q_2).
- **Stato Target:** $|w\rangle = |101\rangle$ (corrispondente all'indice decimale 5).
- **Iterazioni di Grover:** Il numero ottimale di iterazioni k è stato calcolato come $k \approx \frac{\pi}{4}\sqrt{N} \approx 2$.
- **Campionamento:** Sono stati eseguiti 10.000 *shots* per ogni esecuzione al fine di ridurre l'errore statistico di campionamento ($\sigma \approx 1/\sqrt{N_{shots}} = 1\%$).

Il circuito logico implementato, comprendente l'inizializzazione, due iterazioni dell'operatore di Grover (Oracolo + Diffusore) e la misura finale, è rappresentato schematicamente nella Figura 6.1.

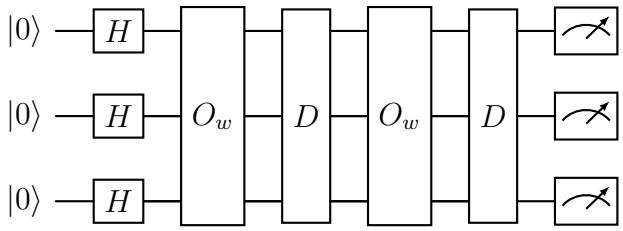

Figura 6.1: Schema logico del circuito di Grover a 3 qubit per due iterazioni. O_w rappresenta l'Oracolo per lo stato $|101\rangle$, D rappresenta l'operatore di Diffusione.

Per garantire la rigorosa riproducibilità dell'esperimento e disaccoppiare i parametri di esecuzione dalla logica dell'algoritmo, è stato adottato un approccio *configuration-driven*. I parametri discussi sopra sono stati definiti nel file `config.json`, utilizzato dal driver di esecuzione per istanziare la classe `GroverAlgorithm` e configurare il servizio IBM Runtime.

```
{
    "n_qubits": 3,
    "target_state": "101",
    "shots": 10000
}
```

Listing 6.4: File di configurazione `config.json` utilizzato per l'esecuzione sperimentale.

L'utilizzo di un file di configurazione strutturato permette di automatizzare la validazione dei vincoli in ingresso (ad esempio, verificando che la lunghezza di `target_state` corrisponda a `n_qubits`) prima di sottomettere il job alla QPU.

6.3.2 Specifiche dell'Hardware: IBM Heron r2

L'esecuzione fisica è stata effettuata sul backend `ibm_fez`, un processore quantistico della famiglia IBM Heron r2 situato nel data center di Washington DC. Le specifiche tecniche del dispositivo al momento dell'esecuzione sono riportate nella Tabella 6.1.

Parametro	**Valore**
Nome Backend	`ibm_fez`
Tipo Processore	Heron r2 (Superconduttore)
Numero di Qubit	156
Basis Gates (ISA)	`cz`, `id`, `rx`, `rz`, `rzz`, `sx`, `x`
Tempo di Coerenza T_1 (mediano)	150.7 µs
Tempo di Coerenza T_2 (mediano)	113.91 µs
Errore gate a 2 Qubit (mediano)	2.55×10^{-3}
Errore di Readout (mediano)	9.52×10^{-3}

Tabella 6.1: Specifiche tecniche del processore `ibm_fez` utilizzato per l'esperimento.

6.3.3 Analisi Comparativa: Simulazione vs Hardware

I risultati ottenuti mostrano una chiara discrepanza tra il comportamento ideale teorico e quello fisico, evidenziando le sfide del calcolo in regime NISQ (*Noisy Intermediate-Scale Quantum*).

Il Costo della Transpilation

Il circuito logico definito in Qiskit utilizza porte di alto livello come la porta di Toffoli (CCX), necessaria per l'oracolo e il diffusore. Tuttavia, l'hardware supporta nativamente solo un set ridotto di istruzioni (Basis Gates: `cz`, `sx`, `rz`, ecc.). Come evidenziato dai log di esecuzione (Tabella 6.2), il compilatore ha dovuto decomporre i 4 gate Toffoli logici in una sequenza complessa di gate fisici.

Metrica	**Simulatore (Logico)**	**Hardware (Fisico)**
Profondità (Depth)	10	139
Gate Count Totale	21	186
Gate a 2 Qubit (Entanglement)	0 (Astratti)	39 (CZ)
Gate a 1 Qubit	21	147 (SX, RZ, X)
Fattore di Espansione Profondità	-	**+1290%**

Tabella 6.2: Confronto delle metriche circuitali prima e dopo la transpilation sul backend. Si nota l'esplosione della complessità dovuta alla decomposizione dei gate Toffoli.

La decomposizione dei gate Toffoli ha introdotto 39 gate CZ fisici. Poiché ogni gate CZ ha un errore mediano di 2.55×10^{-3}, la probabilità di successo del circuito diminuisce esponenzialmente con il numero di gate applicati. Inoltre, l'aumento della profondità a 139 step implica una durata complessiva dell'esecuzione che si avvicina pericolosamente ai tempi di decoerenza T_2 (113.91 µs), causando perdita di informazione quantistica.

Fedeltà dei Risultati

La distribuzione dei risultati di misura per i 10.000 shots è riassunta nella Tabella 6.3.

Stato Misurato	Simulatore (%)	Hardware Reale (%)
$\|\mathbf{101}\rangle$ **(Target)**	**94.06%**	**74.99%**
$\|000\rangle$	0.85%	2.95%
$\|001\rangle$	0.77%	3.52%
$\|010\rangle$	0.92%	4.10%
$\|011\rangle$	0.82%	3.20%
$\|100\rangle$	0.91%	4.25%
$\|110\rangle$	0.84%	3.80%
$\|111\rangle$	0.83%	3.19%

Tabella 6.3: Distribuzione delle probabilità di misura. Il simulatore riflette il limite teorico dell'algoritmo di Grover (che non converge al 100% per $N = 8$). L'hardware mostra un calo di fedeltà dovuto al rumore.

Analisi del Rumore:

- **Simulatore:** Il tasso di successo del 94.06% è coerente con la teoria. L'algoritmo di Grover su 3 qubit non raggiunge ampiezza 1 esatta sullo stato target dopo 2 iterazioni, lasciando un residuo matematico distribuito sugli altri stati.
- **Hardware:** La probabilità di successo è scesa al 74.99%, registrando una perdita di fedeltà del $\approx 19\%$ rispetto all'ideale. Tuttavia, il rapporto segnale-rumore (SNR) rimane elevato: lo stato target $|101\rangle$ è stato misurato circa 20 volte più frequentemente del rumore di fondo medio (circa 3-4% per gli stati non target).

Il test valida l'efficacia dell'algoritmo anche in presenza di rumore, ma dimostra inequivocabilmente come l'astrazione software (il codice Qiskit) debba necessariamente confrontarsi con i vincoli fisici dell'hardware.

Conclusioni

Il percorso di ricerca svolto in questa tesi ha attraversato le fondazioni teoriche e le sfide implementative del Quantum Computing, delineando il profilo di una disciplina che non rappresenta una mera evoluzione dell'informatica classica, ma un radicale cambio di paradigma. L'analisi si è mossa dalla formalizzazione matematica degli spazi di Hilbert e degli operatori unitari fino alla loro traduzione in artefatti software eseguibili, culminando nella validazione sperimentale su architetture reali.

L'implementazione dell'algoritmo di Grover, cuore di questo lavoro, dal modello teorico astratto a un'implementazione concreta tramite il framework Qiskit. L'esperimento condotto, confrontando l'esecuzione su un simulatore ideale con quella sul processore fisico IBM, ha offerto una visione chiara dello stato dell'arte e delle attuali limitazioni tecnologiche. I dati raccolti hanno evidenziato una dicotomia fondamentale tra la logica algoritmica e la realtà fisica. In ambiente simulato, l'algoritmo ha dimostrato una fedeltà del 94.06%, un risultato in perfetto accordo con le previsioni teoriche, mentre il passaggio all'hardware reale ha comportato un calo della fedeltà al 74.99%. Questo risultato conferma la capacità della macchina di distinguere nettamente la soluzione corretta dal rumore di fondo, la degradazione del segnale porta alla luce le criticità dell'era NISQ (*Noisy Intermediate-Scale Quantum*).

L'aspetto più rilevante emerso dall'analisi non è tanto l'errore in sé, quanto la sua genesi strutturale legata al processo di *transpilation*. Abbiamo osservato come la necessità di mappare porte logiche astratte, come il gate di Toffoli, sul set di istruzioni nativo e sulla topologia ristretta del chip superconduttore abbia causato un'esplosione della profondità del circuito superiore al 1290%. Questo fenomeno dimostra che, nel calcolo quantistico odierno, l'astrazione software non è "gratuita": ogni livello logico aggiuntivo introduce un costo fisico tangibile in termini di decoerenza e fedeltà dei gate. Di conseguenza, l'ingegnere del software non può limitarsi alla traduzione di formule matematiche in codice, ma deve necessariamente possedere una sensibilità "hardware-aware", capace

di ottimizzare gli algoritmi in funzione dei vincoli fisici della macchina target, come i tempi di coerenza T_1 e T_2 e la connettività dei qubit.

Nonostante queste sfide tecniche, le prospettive future appaiono estremamente promettenti. L'informatica classica, per quanto potente, si sta avvicinando asintoticamente ai limiti fisici imposti dalla termodinamica e dalla litografia, rendendo intrattabili intere classi di problemi, dalla simulazione molecolare per la farmaceutica all'ottimizzazione logistica complessa. I risultati di questa tesi confermano che, anche in presenza di rumore, la manipolazione dell'interferenza quantistica è una realtà ingegneristica funzionante e non più solo una speculazione teorica.

Guardando avanti, il superamento dell'era NISQ avverrà attraverso l'integrazione di tecniche di correzione d'errore quantistico (*Quantum Error Correction*) e lo sviluppo di architetture *Fault-Tolerant* scalabili. Nel breve termine, l'approccio ibrido, che vede la QPU (Quantum Processing Unit) agire come coprocessore specializzato per task specifici all'interno di un flusso di lavoro classico, rappresenta la strada più percorribile per ottenere un vantaggio computazionale concreto. Il Quantum Computing si conferma come l'unica via percorribile per superare le barriere computazionali attuali, aprendo un nuovo orizzonte dove evolversi per gestire la natura probabilistica e le straordinarie potenzialità della meccanica quantistica.

Bibliografia

[1] Franklin de Lima Marquezino, Renato Portugal, and Carlile Lavor. *A Primer on Quantum Computing*. SpringerBriefs in Computer Science. Springer Nature Switzerland AG, Cham, Switzerland, 2019.

[2] Jack D. Hidary. *Quantum Computing: An Applied Approach*. Springer International Publishing, Cham, Switzerland, 2019.

[3] Eric R. Johnston, Nic Harrigan, and Mercedes Gimeno-Segovia. *Programming Quantum Computers: Essential Algorithms and Code Samples*. O'Reilly Media, Inc., Sebastopol, CA, 2019. Focus pratico sulla programmazione e simulatori come QCEngine.

[4] Venkateswaran Kasirajan. *Fundamentals of Quantum Computing: Theory and Practice*. Springer Nature Switzerland AG, Cham, Switzerland, 2021.

[5] N. David Mermin. *Quantum Computer Science: An Introduction*. Cambridge University Press, Cambridge; New York, 2007. Un approccio mirato specificamente agli informatici.

[6] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge; New York, 10th anniversary edition edition, 2010. Il testo di riferimento standard per il campo.

[7] Arthur O. Pittenger. *An Introduction to Quantum Computing Algorithms*, volume 19 of *Progress in Computer Science and Applied Logic*. Birkhäuser, Boston, 2000.

[8] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2026. Accessed: 2026-01-26.

[9] Gabriele Rossini. Qiskit Implementation of Grover's Algorithm - Thesis Repository. https://github.com/gabrosys/unibs-bsc-thesis-quantum-computing, 2026. Accessed: 2026-01-26.

[10] Wolfgang Scherer. *Mathematics of Quantum Computing: An Introduction*. Springer Nature Switzerland AG, Cham, Switzerland, 2019. Focalizzato sulle strutture matematiche formali (Spazi di Hilbert, Operatori).

[11] Colin P. Williams. *Explorations in Quantum Computing*. Texts in Computer Science. Springer-Verlag, London, 2nd edition, 2011.

Ringraziamenti

"If I have seen further than others, it is by standing upon the shoulders of giants."
— Isaac Newton

Il primo ringraziamento va al Prof. Luca Giuzzi, per avermi accompagnato con competenza e disponibilità in questo lavoro di tesi.

Questo traguardo non sarebbe stato possibile senza i miei zii, che mi hanno fornito il sostegno fondamentale per avviare i miei studi: a voi va la mia più sincera gratitudine per aver creduto in me fin dal primo giorno.

Un grazie di cuore alla mia famiglia e ai miei suoceri, porti sicuri e fonti inesauribili di supporto morale. Grazie anche ai miei amici e colleghi: grazie per la pazienza e per aver trasformato ogni esame superato in un momento di festa condivisa.

Ad Alessandro, alias 'Lancillotto': non sei stato solo un compagno di corso, ma un alleato necessario. Questo risultato è anche merito tuo: preparati, perché il prossimo brindisi sarà tutto per te.

Infine, il posto d'onore spetta a Laura, la mia compagna. Grazie per essere stata il mio punto fermo e per aver compreso le lunghe ore di studio. Ti avevo promesso che ci saremmo sposati solo dopo aver raggiunto questo traguardo: oggi posso dire che ci siamo riusciti, promessa mantenuta. Questa laurea è per noi, il primo passo verso il nostro 2026.

Brescia, Febbraio 2026
Gabriele