Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

Algoritmi dell'informazione quantistica

Relatore:
Prof. Elisa Ercolessi

Correlatore:
Prof. Fabio Ortolani

Presentata da:
Stefano Ragni

Anno Accademico 2016/2017

Sommario

Lo scopo di questa tesi è presentare alcuni algoritmi di base che rivestono grande importanza nell'ambito della computazione quantistica. La trattazione teorica viene accompagnata da codici eseguibili in QISKit, uno strumento che IBM mette a libera disposizione di studenti e docenti universitari: scopo ulteriore della tesi è saggiare le possibilità offerte dal toolkit, la sua semplicità di utilizzo e di conseguenza la sua validità come supporto didattico. Il capitolo centrale sugli algoritmi sarà preceduto da una breve presentazione dei fondamenti della meccanica quantistica necessari per la comprensione dei sistemi a due stati e da una piccola presentazione di QISKit. Saranno analizzati nel dettaglio la procedura di teletrasporto quantistico e gli algoritmi della trasformata di Fourier quantistica e della stima di fase.

Indice

1 Cenni introduttivi **2**
1.1 Il qubit come sistema quantistico a due stati . . . 2
1.1.1 Spazio degli stati . . . 2
1.1.2 Evoluzione temporale dello stato . . . 4
1.1.3 Misura dello stato . . . 5
1.2 Qubit multipli e prodotto tensoriale . . . 6
1.3 Porte quantistiche e circuiti . . . 7
1.4 Operazioni controllate . . . 10

2 IBM Quantum Experience **12**
2.1 QISKit . . . 13
2.1.1 Il modulo *qiskit* . . . 13
2.1.2 Esecuzione del codice su un processore quantistico reale . . . 15

3 Algoritmi quantistici **17**
3.1 Teletrasporto quantistico . . . 17
3.1.1 Realizzazione del circuito . . . 20
3.2 Trasformata di Fourier . . . 22
3.2.1 Cenni alla trasformata di Fourier discreta . . . 22
3.2.2 La trasformata di Fourier quantistica . . . 24
3.2.3 Descrizione dell'algoritmo e realizzazione del circuito . . . 29
3.3 Stima di fase . . . 36
3.3.1 Realizzazione del circuito . . . 38

Capitolo 1

Cenni introduttivi

L'informazione quantistica è lo studio delle tecniche che permettono di memorizzare ed elaborare le informazioni sfruttando un sistema quantistico. Il punto fondamentale da chiarire è che l'informazione è fisica, ovvero viene descritta da una qualche variabile di un sistema fisico: ad esempio nel mondo di oggi i computer immagazzinano i dati sotto forma di valori di tensione elettrica, che interpretata come alta o bassa fornisce un'informazione di tipo binario, e circuiti elettrici sono utilizzati per la sua manipolazione. Tutto questo è basato sostanzialmente sulla fisica dell'elettromagnetismo, una teoria classica, per cui si parla di informazione classica; ma questa non è l'unica possibilità, dato che in linea di principio qualunque sistema fisico può essere usato.
I sistemi quantistici sono interessanti da questo punto di vista perché descritti matematicamente da uno spazio di Hilbert: un *qubit*, ovvero un sistema che una volta misurato può fornire soltanto due possibili risultati, possiede dentro di sé una variabilità infinitamente maggiore. In questo capitolo si vedrà che gli stati occupabili da un qubit possono essere messi in corrispondenza con i punti di una superficie (un oggetto bidimensionale) di una sfera, anche se, come accennato, non sono direttamente accessibili alla misurazione. Il punto cruciale sta nello sfruttamento di questo contenuto informativo nascosto, che a volte rende possibile lo sviluppo di algoritmi migliori di quelli classici dal punto di vista computazionale. Nella sezione seguente verranno citati alcuni postulati di meccanica quantistica necessari per la descrizione matematica del qubit.

1.1 Il qubit come sistema quantistico a due stati

1.1.1 Spazio degli stati

Postulato 1 *A ciascun sistema fisico isolato è associato uno spazio vettoriale complesso dotato di prodotto scalare (ovvero uno spazio di Hilbert) noto come lo spazio degli stati del*

sistema. Il sistema è completamente descritto dal suo vettore di stato, che è un vettore unitario nello spazio degli stati del sistema.

Il sistema quantomeccanico che viene considerato nell'ambito dell'informazione quantistica è il *qubit* (quantum bit). Un qubit ha uno spazio degli stati bidimensionale. Indicando con $|0\rangle$ e $|1\rangle$ una base ortonormale per tale spazio, un arbitrario vettore di stato può essere scritto nel seguente modo:

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle \tag{1.1}$$

dove a e b sono numeri complessi. La condizione che $|\phi\rangle$ sia un vettore unitario, $\langle\psi|\psi\rangle = 1$, è quindi equivalente a $|a|^2 + |b|^2 = 1$. Tale condizione è nota come *condizione di normalizzazione* per i vettori di stato. Quanto detto in precedenza significa che, a differenza di un bit classico che può assumere unicamente i valori 0 e 1, un qubit si trova in generale in una combinazione lineare degli stati $|0\rangle$ e $|1\rangle$ a cui viene dato il nome di *sovrapposizione* di stati, e la base $\{|0\rangle, |1\rangle\}$ è denominata *base computazionale*. Una maniera utile per visualizzare lo stato di un qubit è data dalla seguente rappresentazione geometrica: dato che $|a|^2 + |b|^2 = 1$, è possibile scrivere:

$$|a| = \cos\frac{\theta}{2} \quad \text{e} \quad |b| = \sin\frac{\theta}{2} \quad \text{con} \quad \theta \in [0, \pi] \tag{1.2}$$

nella quale l'intervallo per θ è quello in cui le funzioni trigonometriche sono entrambe positive. Esprimendo i numeri complessi della 1.1 in forma polare e utilizzando la 1.2:

$$|\phi\rangle = |a|e^{i\phi_a}\,|0\rangle + |b|e^{i\phi_b}\,|1\rangle = \cos\frac{\theta}{2}e^{i\phi_a}\,|0\rangle + \sin\frac{\theta}{2}e^{i\phi_b}\,|1\rangle = \\ = e^{i\phi_a}\left(\cos\frac{\theta}{2}\,|0\rangle + \sin\frac{\theta}{2}e^{i(\phi_b-\phi_a)}\,|1\rangle\right)$$

Ponendo $\gamma = \phi_a$ e $\phi = \phi_b - \phi_a$ si ottiene:

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

nella quale θ, ϕ e γ sono numeri reali. Il fattore $e^{i\gamma}$ è a norma unitaria e moltiplica l'intero stato: per questo motivo viene chiamato *fase globale*. Nel paragrafo relativo alla misurazione di uno stato quantistico sarà evidenziato il fatto che una fase globale non produce effetti osservabili, e dunque è possibile ignorare tale fattore e scrivere:

$$|\phi\rangle = \cos\frac{\theta}{2}\,|0\rangle + \sin\frac{\theta}{2}e^{i\phi}\,|1\rangle \tag{1.3}$$

Visto che ora lo stato è scritto in una forma in cui compaiono i parametri reali $\theta \in [0, \pi]$ e $\phi \in [0, 2\pi[$, emerge una corrispondenza biunivoca tra i possibili vettori di stato di

un qubit e i punti sulla superficie di una sfera tridimensionale: in questo contesto tale sfera è chiamata *sfera di Bloch*. Facendo riferimento alla Figura 1.1, a θ è associato il significato di coordinata polare sferica definente l'angolo tra l'asse z e il vettore di stato (in nero) che individua un generico punto sulla superficie; ϕ indica invece la coordinata longitudinale, ovvero l'angolo tra la proiezione del vettore di stato sul piano xy e l'asse x.
I vettori della base computazionale corrispondono ai poli della sfera, ovvero alle due direzioni (positiva per $|0\rangle$ e negativa per $|1\rangle$) dell'asse z. Anche le operazioni sul singolo qubit che saranno descritte più avanti, di norma espresse tramite matrici complesse, avranno in questo modo una rappresentazione visiva in termini di trasformazioni agenti sulla sfera di Bloch. Si sottolinea però che tale intuizione è limitata perché non esiste una semplice generalizzazione della sfera di Bloch quando si dovranno considerare sistemi descritti da qubit multipli.

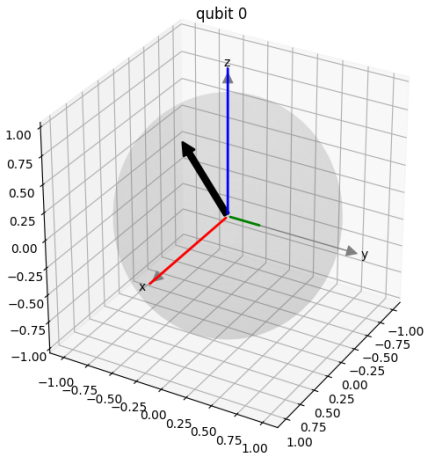


Figura 1.1: Rappresentazione di un vettore di stato nella sfera di Bloch

1.1.2 Evoluzione temporale dello stato

Postulato 2 *L'evoluzione di un sistema quantistico è descritto da una trasformazione unitaria: lo stato $|\psi_1\rangle$ del sistema al tempo t_1 è legato allo stato $|\psi_2\rangle$ del sistema al tempo t_2 da un operatore unitario U che dipende solamente dagli istanti t_1 e t_2,*

$$|\psi_2\rangle = U\,|\psi_1\rangle \tag{1.4}$$

Tale postulato implica in sostanza che la trasformazione deve mantenere la probabilità (unitaria) di trovare il sistema in una delle sue possibili configurazioni, visto che essa è

legata alla norma del vettore di stato.
Una trasformazione unitaria possiede autovalori unitari ed autovettori che formano una base ortonormale dello spazio; inoltre, essendo lineare, è definita univocamente dai valori assunti sulla base.

1.1.3 Misura dello stato

Postulato 3 *Le misurazioni quantistiche sono descritte da una collezione M_m di* operatori di misurazione*. Essi sono operatori che agiscono sullo spazio degli stati del sistema che sta venendo misurato. L'indice m si riferisce ai possibili risultati che possono scaturire dall'esperimento. Se lo stato del sistema quantistico è $|\phi\rangle$ immediatamente prima della misura, allora la probabilità che il risultato m occorra è data da:*

$$p(m) = \langle\phi| M_m^\dagger M_m |\phi\rangle \tag{1.5}$$

e lo stato del sistema dopo la misura è:

$$\frac{M_m |\psi\rangle}{\sqrt{\langle\phi| M_m^\dagger M_m |\phi\rangle}} \tag{1.6}$$

Gli operatori di misurazione soddisfano la relazione di completezza*,*

$$\sum_m M_m^\dagger M_m = I \tag{1.7}$$

La relazione di completezza esprime il fatto che le probabilità si sommano a uno:

$$1 = \sum_m p(m) = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle \tag{1.8}$$

Il più semplice esempio di misurazione è la misura di un qubit nella base computazionale. Introducendo i relativi operatori $M_0 = |0\rangle\langle 0|$ e $M_1 = |1\rangle\langle 1|$, si osserva che sono Hermitiani e che $M_0^2 = M_0$ e $M_1^2 = M_1$. Dunque $M_0^\dagger M_0 + M_1^\dagger M_1 = M_0 + M_1 = I$ e la relazione di completezza è soddisfatta. Allora la probabilità di ottenere il risultato 0 è:

$$p(0) = \langle\phi| M_0^\dagger M_0 |\phi\rangle = \langle\phi|0\rangle\langle 0|\phi\rangle = |a|^2 \tag{1.9}$$

Similmente, la probabilità di ottenere il risultato 1 è $p(1) = |b|^2$. Lo stato dopo la misura nei due casi diventa:

$$\frac{M_0 |\psi\rangle}{|a|} = \frac{a}{|a|} |0\rangle \quad \text{e} \quad \frac{M_1 |\psi\rangle}{|b|} = \frac{b}{|b|} |1\rangle \tag{1.10}$$

Nel prossimo paragrafo verrà visto che fattori globali di modulo unitario come $a/|a|$ possono essere ignorati, dunque gli stati ottenuti dopo la misurazione sono effettivamente $|0\rangle$ e $|1\rangle$. Questo processo viene spesso detto *collasso dello stato quantistico* in uno degli autostati dell'insieme di operatori di misurazione considerati.

Fattori di fase globali e relativi

Si consideri lo stato $e^{i\theta}|\psi\rangle$, dove $|\psi\rangle$ è un vettore di fase e θ è un numero reale: lo stato $e^{i\theta}|\psi\rangle$ viene detto uguale allo stato $|\psi\rangle$ a meno di un *fattore di fase globale* $e^{i\theta}$. Si nota che la statistica di misurazione prevista da questi due stati è la stessa. Per mostrare ciò si calcolino le probabilità di ottenere il risultato m associate all'azione di un operatore di misura M_m usando la 1.5:

$$|\psi\rangle \to \langle\psi| M_m^\dagger M_m |\psi\rangle$$
$$e^{i\theta}|\psi\rangle \to \langle\psi| e^{-i\theta} M_m^\dagger M_m e^{i\theta} |\psi\rangle = \langle\psi| M_m^\dagger M_m |\psi\rangle$$

Essendo irrilevanti nelle proprietà di un sistema fisico che possono essere osservate, i fattori di fase globali verranno ignorati. Un'altra categoria di fattori di fase comprende i *fattori di fase relativi*, che hanno caratteristiche piuttosto differenti. Quando due coefficienti a e b di uno stato sono tali che $a = e^{i\theta}b$ si dice che a e b differiscono per un fattore di fase relativo. Ad esempio:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{|0\rangle + e^{i0}|1\rangle}{\sqrt{2}} \qquad \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{|0\rangle + e^{i\pi}|1\rangle}{\sqrt{2}}$$

e quindi i due coefficienti differiscono di un angolo $\theta = \pi$. Si sottolinea che, a differenza dei fattori di fase globali, quelli relativi influiscono nelle misure e dipendono inoltre dalla base in cui si sceglie di esprimere un certo stato.

1.2 Qubit multipli e prodotto tensoriale

Il *prodotto tensoriale* è un'operazione che mette insieme spazi vettoriali per formare uno spazio vettoriale più grande: ciò permette di descrivere spazi di Hilbert di dimensione arbitraria tramite qubit multipli, analogamente al modo con cui un computer classico rappresenta i dati come insieme di bit.
Supponendo di partire dagli spazi di Hilbert V e W, di dimensione rispettivamente m e n, si indica il prodotto tensoriale di V con W con il simbolo $V \otimes W$: esso è uno spazio vettoriale di dimensione mn, e Per definizione il prodotto tensoriale soddisfa le seguenti proprietà:

- $\forall z \in \mathbb{C}, \forall |v\rangle \in V, \forall |w\rangle \in W \quad z(|v\rangle \otimes |w\rangle) = (z|v\rangle) \otimes |w\rangle = |v\rangle \otimes (z|w\rangle)$
- $\forall |v_1\rangle, |v_2\rangle \in V, \forall |w\rangle \in W (|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle$
- $\forall |v\rangle \in V, \forall |w_1\rangle, |w_2\rangle \in W (|v\rangle \otimes |w_1\rangle + |w_2\rangle) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle$

Un esempio di stato quantistico a 2 qubit è lo *stato di Bell*:

$$\psi_{AB} \equiv \frac{|0_A\rangle \otimes |0_B\rangle + |1_A\rangle \otimes |1_B\rangle}{\sqrt{2}} \equiv \frac{|0_A 0_B\rangle + |1_A 1_B\rangle}{\sqrt{2}}$$

Si noti che tale stato non può essere scritto come prodotto tensoriale di due stati ad 1 qubit e pertanto viene detto *entangled*.

1.3 Porte quantistiche e circuiti

Gli operatori unitari che determinano l'evoluzione di uno stato quantistico vengono chiamati *porte* o *gate* nel gergo della computazione quantistica. Si elencano i gate più rilevanti, scritti in forma matriciale rispetto alla base computazionale:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{1.11}$$

note come le *matrici di Pauli*, e:

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad S \equiv \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \qquad T \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \tag{1.12}$$

rispettivamente *gate di Hadamard*, *gate di fase* e *gate* $\pi/8$. Il nome del gate T deriva dal fatto che storicamente veniva espresso nella forma:

$$e^{i\pi/8} \begin{bmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{bmatrix}$$

equivalente alla precedente perché ottenuta dal semplice raccoglimento di un fattore di fase globale.
Per comprendere invece i nomi assegnati alle matrici di Pauli è necessario trovarne gli autovalori e autovettori (unitari):

$$X - \lambda I \equiv \begin{bmatrix} -\lambda & 1 \\ 1 & -\lambda \end{bmatrix} \qquad Y - \lambda I \equiv \begin{bmatrix} -\lambda & -i \\ i & -\lambda \end{bmatrix} \qquad Z - \lambda I \equiv \begin{bmatrix} 1-\lambda & 0 \\ 0 & -1-\lambda \end{bmatrix}$$

Risolvendo l'equazione caratteristica per ciascuna matrice:

- $\lambda^2 - 1 = 0 \to \lambda_{1/2} = \pm 1$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \to x = y \to v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = - \begin{bmatrix} x \\ y \end{bmatrix} \to -x = y \to v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

- $\lambda^2 - 1 = 0 \to \lambda_{1/2} = \pm 1$

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \to ix = y \to v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ i \end{bmatrix}$$
$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = - \begin{bmatrix} x \\ y \end{bmatrix} \to -ix = y \to v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -i \end{bmatrix}$$

- Z è già diagonalizzata, dunque i suoi autovettori unitari sono i vettori della base computazionale.

Se si interpretano gli autovettori trovati nel contesto della sfera di Bloch, si nota che gli autovettori di X rappresentano i due versi (positivo e negativo) della direzione indicata dall'asse x, e analogamente per Y e Z. Ad esempio per X si ha:

$$|v_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{i0}|1\rangle \rightarrow \theta = \frac{\pi}{2},\ \phi = 0$$

Quando le matrici di Pauli vengono esponenziate danno origine agli *operatori di rotazione* attorno agli assi x, y e z:

$$R_x(\theta) = e^{-i\theta X/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{1.13}$$

$$R_y(\theta) = e^{-i\theta Y/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{1.14}$$

$$R_z(\theta) = e^{-i\theta Z/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix} \tag{1.15}$$

Se $\hat{n} = (n_x, n_y, n_z)$ è un vettore reale unitario in tre dimensioni, si possono generalizzare le definizioni precedenti a rotazioni di angolo θ attorno all'asse $\hat{n}$ tramite la formula:

$$R_{\hat{n}}(\theta) \equiv e^{-i\theta\hat{n}\cdot\vec{\sigma}/2} = \cos\left(\frac{\theta}{2}\right)I - i\sin\left(\frac{\theta}{2}\right)(n_xX + n_yY + n_zZ) \tag{1.16}$$

dove $\vec{\sigma}$ denota un vettore operatoriale le cui tre componenti sono le matrici di Pauli. Tale definizione è importante perché si può dimostrare che ogni operatore unitario agente su un singolo qubit può essere scritto nella forma:

$$U = e^{i\alpha}R_{\hat{n}}(\theta) \tag{1.17}$$

ovvero una rotazione del tipo 1.16 moltiplicata per una fase globale. Inoltre, come anticipato alla fine della sezione 1.1.1, rotazioni definite in questo modo corrispondono proprio alle rotazioni della sfera di Bloch attorno all'asse definito dal vettore $\hat{n}$ di un angolo θ.

Rappresentazione del gate di Hadamard come rotazione

Per trovare il l'asse $\hat{n}$ e l'angolo θ corrispondenti al gate di Hadamard (definito in 1.12) si potrebbe procedere in questo modo: una rotazione sulla sfera deve lasciare invariato lo stato che rappresenta l'asse di rotazione, ma tale stato è per definizione autovettore

dell'operatore in questione.
Si risolve dunque il problema agli autovalori per H:

$$H - \lambda I \equiv \begin{bmatrix} \frac{1}{\sqrt{2}} - \lambda & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} - \lambda \end{bmatrix}$$

Risolvendo l'equazione caratteristica:

$$-\left(\frac{1}{2} - \lambda^2\right) - \frac{1}{2} = 0 \rightarrow \lambda^2 - 1 = 0 \rightarrow \lambda_{1/2} = \pm 1$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow y = (\sqrt{2} - 1)x \rightarrow v_1 = \frac{1}{2} \begin{bmatrix} \sqrt{2 + \sqrt{2}} \\ \sqrt{2 - \sqrt{2}} \end{bmatrix}$$
$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = - \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow y = (-\sqrt{2} - 1)x \rightarrow v_2 = \frac{1}{2} \begin{bmatrix} \sqrt{2 - \sqrt{2}} \\ -\sqrt{2 + \sqrt{2}} \end{bmatrix}$$

e passando alla descrizione nella sfera di Bloch:

$$|v_1\rangle = \frac{\sqrt{2 + \sqrt{2}}}{2} |0\rangle + \frac{\sqrt{2 - \sqrt{2}}}{2} e^{i0} |1\rangle \rightarrow \theta = \frac{\pi}{4},\ \phi = 0$$
$$|v_2\rangle = \frac{\sqrt{2 - \sqrt{2}}}{2} |0\rangle + \frac{\sqrt{2 + \sqrt{2}}}{2} e^{i\pi} |1\rangle \rightarrow \theta = \frac{3}{4}\pi,\ \phi = \pi$$

Questi due vettori di stato descrivono i due versi di uno stesso asse, ovvero l'asse di rotazione. Considerando v_1, passando dalle coordinate sferiche a quelle cartesiane si trova $\hat{n} = (\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2})$. Sostituendo l'asse appena trovato nella 1.16:

$$R_{\hat{n}}(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Ora è immediato vedere che il risultato cercato si ottiene per una rotazione di $\theta = \pi$, infatti:

$$iR_{\hat{n}}(\theta) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

che è la 1.17 per $U = H$, $\hat{n} = (\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2})$ e $\theta = \pi$.

1.4 Operazioni controllate

Una *operazione controllata* è una operazione del tipo 'Se A è vero, allora fai B'. Se U è un operatore unitario agente su un singolo qubit, una operazione *U-controllata* è una trasformazione a due qubit, dove uno dei due svolge la funzione di *controllo* (c) e l'altro di *bersaglio* t. Se il qubit di controllo è acceso allora U viene applicato al qubit di bersaglio, altrimenti il qubit di bersaglio resta invariato. La definizione generale è:

$$|0\rangle\langle 0| \otimes \mathcal{I} + |1\rangle\langle 1| \otimes \mathcal{U} \tag{1.18}$$

dove il primo qubit svolge il ruolo di c e l'altro t. Volendo ricavare la rappresentazione matriciale per un'operazione controllata a due qubit, si nota in primo luogo che lo spazio su cui agisce deve essere quadridimensionale (è il prodotto tensoriale degli spazi associati a ciascun qubit). Se c è spento t non deve variare: $|00\rangle \to |00\rangle$ e $|01\rangle \to |01\rangle$, condizioni che ci forniscono le prime due colonne della matrice. Il resto è determinato dall'applicazione di U su t se c è acceso e dal fatto che in ogni caso c non viene influenzato:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{11} & u_{12} \\ 0 & 0 & u_{21} & u_{22} \end{bmatrix}$$

Una operazione controllata di base è costituita dalla porta CNOT (indicato con $C\mathcal{X}$ come operatore e come $\oplus$ negli schemi dei circuiti), che per definizione inverte il qubit di bersaglio quando il qubit di controllo è acceso. L'azione sulla base è la seguente:

$$\begin{aligned} |00\rangle &\to |00\rangle \qquad |10\rangle \to |11\rangle \\ |01\rangle &\to |01\rangle \qquad |11\rangle \to |10\rangle \end{aligned} \tag{1.19}$$

Da cui si ricava la matrice:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Rappresentazione grafica delle porte quantistiche

Invece di descrivere a parole o tramite simboli matematici le porte quantistiche applicate in sequenza, un modo per rappresentare in maniera semplice ed efficace un circuito è tramite uno schema. Si consideri un circuito a due qubit definito dall'applicazione di una porta X a q_1, una porta H a q_0, una porta CZ controllata da q_1 e agente su q_0 e infine ancora una porta H a q_0: lo schema corrispondente è raffigurato nella 1.2.

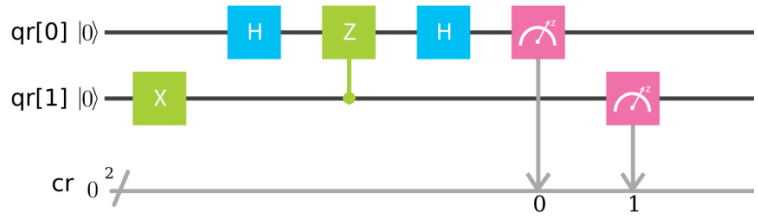


Figura 1.2: Un circuito di esempio

Esempio: semplificazione di un circuito

Il circuito in Figura 1.3 è equivalente ad un circuito formato dalla sola porta CNOT. Per verificarlo si applicano in successione gli operatori rappresentativi delle tre porte allo stato iniziale $|q_1 q_0\rangle$, iniziando da $\mathcal{I} \otimes \mathcal{H}$:

$$|q_1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{q_0} |1\rangle)$$

Applicando ora $\mathcal{CZ}$ controllato da q_1

$$\frac{1}{\sqrt{2}}(\mathcal{CZ} |q_1 0\rangle + (-1)^{q_0} \mathcal{CZ} |q_1 1\rangle) = |q_1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{q_0+q_1} |1\rangle)$$

Infine applicando nuovamente $\mathcal{I} \otimes \mathcal{H}$:

$$|q_1\rangle \otimes \frac{1}{\sqrt{2}}(\mathcal{H} |0\rangle + (-1)^{q_0+q_1} \mathcal{H} |1\rangle) = |q_1\rangle \otimes \frac{1}{2}(|0\rangle + |1\rangle + (-1)^{q_0+q_1}(|0\rangle - |1\rangle)) =$$
$$|q_1\rangle \otimes \left(\frac{1 + (-1)^{q_0+q_1}}{2} |0\rangle + \frac{1 - (-1)^{q_0+q_1}}{2} |1\rangle\right)$$

Al variare di q_0 e q_1 l'azione risultante sulla base è la stessa descritta dalle 1.19 e dunque equivalente ad una porta CNOT.

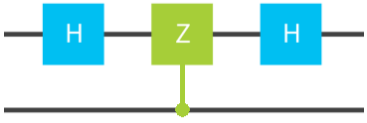


Figura 1.3: Circuito da semplificare

Capitolo 2

IBM Quantum Experience

> "We acknowledge use of the IBM Q experience for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q experience team."

La *IBM Quantum Experience* (spesso abbreviata in *IBM Q Experience* oppure *IBM QX*) è una piattaforma cloud mirata a studenti e ricercatori per interagire online con un vero computer quantistico, collocato in un laboratorio di ricerca IBM.
Al centro delle funzionalità fornite vi è l'accesso a diversi *backend*, ovvero a diverse classi di dispositivi capaci di eseguire programmi quantistici. I backend si suddividono in dispositivi reali, simulatori online e simulatori locali. Di seguito saranno indicati i backend disponibili al tempo della scrittura di questo documento, ma in futuro è prevista l'inclusione di altri dispositivi.
Hardware corrente:

- IBMQX2: un chip 'transmon bowtie' a 5 qubit
- IBMQX4: un chip 'transmon bowtie' a 5 qubit
- IBMQX5: un chip 'ladder' a 16 qubit

Simulatori correnti:

- Simulatore QASM: simula un circuito quantistico e prevede i risultati di un esperimento (la distribuzione degli stati finali su un dato numero di esecuzioni)
- Simulatore QASM con rumore: simula un circuito quantistico aggiungendo rumore in input
- Simulatore unitario: calcola la matrice unitaria di un circuito quantistico (ignorando misurazioni e operazioni condizionali)

La sigla QASM si riferisce al linguaggio 'di basso livello' usato per specificare i programmi sul computer quantistico: in un certo senso è l'analogo del linguaggio *Assembly* della computazione classica, e allo stesso modo lo si ottiene dalla traduzione di un linguaggio ad alto livello. Lo strumento utilizzato per generare codice QASM sarà descritto in breve nel resto del capitolo.

2.1 QISKit

QISKit (acronimo di *Quantum Information Software Kit*) è un Software Development Kit (SDK) per usufruire della IBM Q Experience e facilitare la produzione di codice OpenQASM. QISKit viene usato per creare programmi di computazione quantistica, compilarli ed eseguirli su uno dei possibili backend. A differenza del *Quantum Composer*, che permette la creazione di semplici circuiti attraverso un interfaccia grafica web, QISKit permette la realizzazione di algoritmi più complessi poiché fa uso del linguaggio di programmazione ad alto livello *Python*: tutte le caratteristiche di tale linguaggio (cicli, funzioni, oggetti...) possono dunque essere sfruttate per rendere il circuito quantistico generico e flessibile a piacere.

2.1.1 Il modulo *qiskit*

Come già accennato, gli scopi principali di QISKit sono la produzione di codice OpenQASM e la connessione a uno dei backend disponibili: entrambi i compiti vengono realizzati in Python tramite chiamate a funzioni su opportuni oggetti resi disponibili dall'importazione del modulo *qiskit*. Il metodo consigliato per l'installazione di tutto il necessario per poter lavorare con QISKit SDK consiste nell'utilizzare *pip*, il package manager di Python. In ambiente Linux, ad esempio, è sufficiente eseguire il seguente comando:

```
$ pip install --user qiskit
```

per installare il modulo qiskit e tutte le dipendenze necessarie.

Un circuito di esempio

In questa sezione si illustreranno i semplici passaggi da includere nella realizzazione di qualsiasi circuito in QISKit: la creazione del circuito e dei registri, la selezione del backend, la compilazione e esecuzione del programma e la stampa dei risultati. Il circuito di esempio si limita all'accensione del qubit meno significativo tramite l'applicazione di un gate opportuno:

```
1 # Importazione del modulo
2 from qiskit import QuantumProgram
3
4 # Creazione di QuantumProgram()
```

```
qp = QuantumProgram()

# Creazione di un registro quantistico e uno classico a n bit
n = 3
qr = qp.create_quantum_register("qr", n)
cr = qp.create_classical_register("cr", n)

# Creazione di un circuito che sfutta i registri appena creati
qc = qp.create_circuit("circuito_zero", [qr], [cr])
qc.x(qr[0])

# Misurazione del registro quantistico
qc.measure(qr, cr)

# Esescuzione del circuito sul simulatore locale
result = qp.execute(["circuito_zero"], backend="local_qasm_simulator",
    shots=1024)

# Stampa del codice QASM
print(qp.get_qasm("circuito_zero"))
# Stampa dei risultati su stdout
print(result)
print(result.get_data("circuito_zero"))
```

Tutti i passaggi sono spiegati direttamente nel codice sotto forma di commenti. Alcune precisazioni: per quanto riguarda l'ordine dei qubit, la convenzione seguita è la stessa utilizzata in questo documento, ovvero il LSB[1] è $q[0]$ e il MSB è $q[n-1]$. Al momento della creazione del registro quantistico, tutti i qubit sono implicitamente inizializzati nello stato di partenza $|0\rangle$: per sintetizzare uno stato differente occorre applicare opportuni gate a tale stato. Il registro classico è un oggetto utilizzato principalmente al momento della misurazione: dopo la misurazione, infatti, conterrà il risultato del collasso degli stati quantistici dei qubit sulla base computazionale[2]. Il parametro *shots* passato alla funzione execute determina il numero di volte il circuito descritto viene eseguito, allo scopo di analizzare statisticamente i risultati. L'output del programma contiene una prima parte in cui viene riportata la traduzione in linguaggio QASM, e una riga finale dove è possibile visualizzare il risultato di ciascuna delle 1024 iterazioni:

```
OPENQASM 2.0;
include "qelib1.inc";
qreg qr[3];
creg qc[3];
x qr[0];
measure qr[0] -> qc[0];
measure qr[1] -> qc[1];
```

[1]LSB significa bit meno significativo, MSB bit più significativo.

[2]Nell'atto della misurazione è possibile modificare l'ordine dei qubit misurando ciascun qubit singolarmente e immagazzinando il risultato nella posizione voluta del registro classico.

```
measure qr[2] -> qc[2];

COMPLETED
{'counts': {'001': 1024}}
```

2.1.2 Esecuzione del codice su un processore quantistico reale

Per poter eseguire i codici sui processori quantistici reali forniti da IBM occorre innanzitutto creare un account IBM Q experience[3]. La registrazione è molto rapida e può essere compiuta da ogni studente, visto che richiede esclusivamente l'inserimento dell'istituto di provenienza e del motivo per cui si è interessati ad usufruire di tale strumento. Successivamente è necessario fare il login e generare un *API Token*, ovvero una stringa di caratteri esadecimali che dovrà essere riportata in un file nella seguente maniera:

```
APItoken = "inserire qua il token"

config = {
    "url": "https://quantumexperience.ng.bluemix.net/api",
}
```

Tale file deve essere presente nella cartella in cui è presente il programma principale da eseguire, il quale dovrà importare Qconfig.py ed autenticarsi utilizzando un'apposita funzione. Si riporta un programma esplicativo, che realizza un circuito già presentato nel capitolo precedente (Figura 1.2):

```
from qiskit import QuantumProgram
from qiskit.tools.visualization import plot_histogram
import Qconfig

qp = QuantumProgram()
# Necessaria per l'accesso al processore reale
qp.set_api(Qconfig.APItoken, Qconfig.config["url"])

n = 2
qr = qp.create_quantum_register("qr", n)
cr = qp.create_classical_register("cr", n)

qc = qp.create_circuit("cnot", [qr], [cr])

# Preparo lo stato iniziale
qc.x(qr[1])
# L'insieme di questi gates equivale ad un CNOT
qc.h(qr[0])
qc.cz(qr[1], qr[0])
qc.h(qr[0])
```

[3] https://quantumexperience.ng.bluemix.net/

```
21
22 qc.measure(qr, cr)
23
24 # Il parametro passato alla seguente funzione seleziona il backend
25 result = qp.execute(["cnot"], backend="ibmqx5", shots=1024)
26
27 print(qp.get_qasm("cnot"))
28 print(result)
29 plot_histogram(data = result.get_counts("cnot"))
```

Eseguendo il sorgente si ottiene il seguente output:

```
OPENQASM 2.0;
include "qelib1.inc";
qreg qr[2];
creg cr[2];
x qr[1];
h qr[0];
cz qr[1],qr[0];
h qr[0];
measure qr[0] -> cr[0];
measure qr[1] -> cr[1];

COMPLETED
```

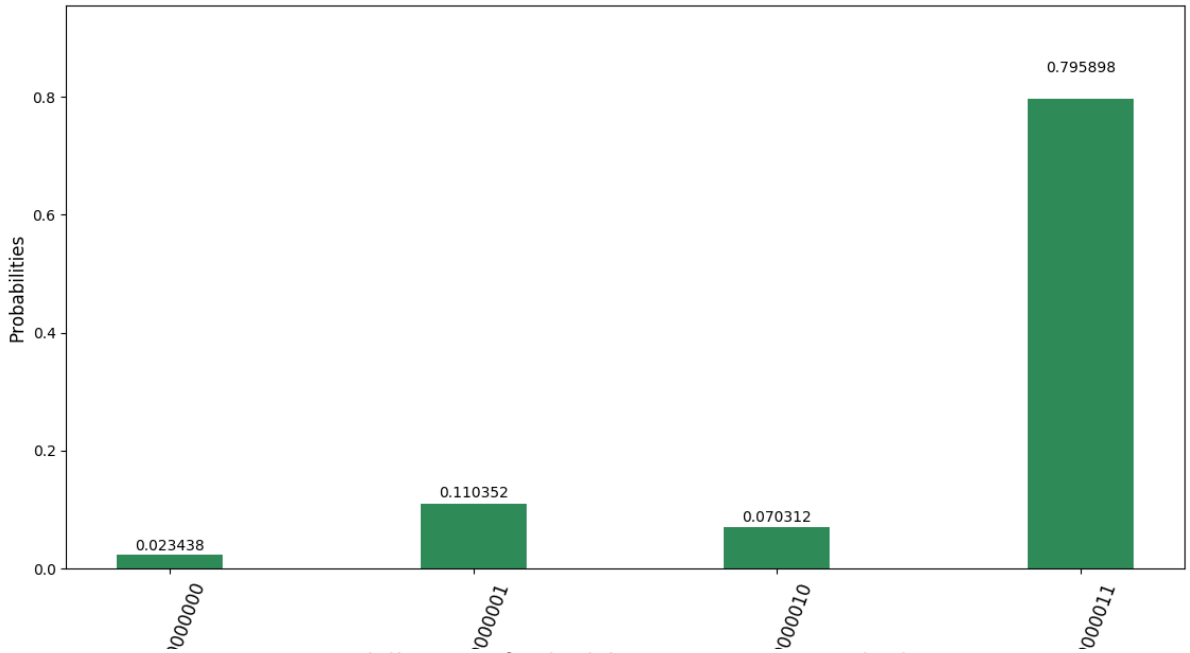


Figura 2.1: Frequenze dello stato finale del registro su un totale di 1024 esecuzioni

Risultato in Figura 2.1: si nota che il processore reale subisce effetti di disturbo, deviando dal comportamento ideale (tutti gli shots dovrebbero rendere $|11\rangle$).

Capitolo 3

Algoritmi quantistici

La promessa dei computer quantistici consiste nel permettere nuovi algoritmi che rendono possibili compiti che richiederebbero risorse enormi ad essere risolti su computer classici. Ad oggi esistono due categorie di algoritmi quantistici che realizzano questa promessa: la prima è basata sulla *trasformata di Fourier quantistica* elaborata da Shor ed include l'algoritmo di fattorizzazione, punto cruciale del sistema di crittografia RSA. La sua descrizione occuperà la maggior parte del capitolo, ma è preceduta da una discussione sul trasferimento di 'dati quantistici' detto *teletrasporto quantistico*. La seconda classe è basata sull'algoritmo di Grover della *ricerca quantistica*, che invece non viene trattato in questa tesi.

3.1 Teletrasporto quantistico

Per teletrasporto quantistico si intende una procedura capace di trasferire stati tra i costituenti di un sistema quantistico: la trattazione verrà limitata alla trasmissione di un singolo qubit tra due parti, che saranno indicate con $\hat{A}$ e $\hat{B}$. In questo caso l'algoritmo richiede tre qubit, denotati con Q, A e B, dove i primi due appartengono all'ente $\hat{A}$ e l'ultimo appartiene naturalmente a $\hat{B}$. La funzione d'onda dell'intero sistema verrà dunque indicata con $|\psi_{QAB}\rangle$: si raccomanda di fare attenzione all'ordine scelto per i qubit (B è il LSB e ha indice 0, A ha indice 1, ecc...)
Il qubit che l'ente $\hat{A}$ vuole trasmettere a $\hat{B}$ è Q, e pertanto si suppone che sia in uno stato qualsiasi $|\psi_Q\rangle$. I qubit A e B servono invece a realizzare la comunicazione e inizialmente si trovano allo stato fondamentale $|0_A\rangle$ e $|0_B\rangle$. Si elencano ora i passaggi di cui si compone l'algoritmo:

1. Preparare il sistema di qubit AB nello stato di Bell

2. Applicare i gate del 'teletrasporto'

3. Misurare i qubit Q ed A e comunicare il risultato a $\hat{B}$, che in base a ciò applicherà il corretto operatore al suo qubit

Preparazione dello stato di Bell

Il primo passo consiste nella preparazione dello stato di Bell a partire da due qubit nello stato fondamentale. Si ricorda l'espressione dello stato di Bell, che era già stato presentato nella sezione 1.2:

$$\psi_{AB} = \frac{|0_A 0_B\rangle + |1_A 1_B\rangle}{\sqrt{2}}$$

Si nota che tale stato assomiglia a quello ottenuto applicando un gate di Hadamard ad un singolo qubit nello stato fondamentale. Infatti provando ad effettuare tale operazione su uno dei due qubit (ad esempio B) lo stato risultante è:

$$(\mathcal{I} \otimes \mathcal{H}) |0_A 0_B\rangle = |0_A\rangle \otimes \frac{|0_B\rangle + |1_B\rangle}{\sqrt{2}} = \frac{|0_A 0_B\rangle + |0_A 1_B\rangle}{\sqrt{2}}$$

A questo punto per ottenere lo stato di Bell si dovrebbe applicare un'operazione che lasci invariato il primo termine della somma e 'spenga' il bit relativo ad A nel secondo termine. Ma tale operazione è realizzata proprio dalla porta CNOT controllata da q_B:

$$\frac{C\mathcal{X} |0_A 0_B\rangle + (C\mathcal{X} |0_A 1_B\rangle}{\sqrt{2}} = \frac{|0_A 0_B\rangle + |1_A 1_B\rangle}{\sqrt{2}}$$

Il nucleo dell'algoritmo

A questo punto il sistema complessivo si trova nello stato:

$$|\psi_Q\rangle \otimes \frac{|0_A 0_B\rangle + |1_A 1_B\rangle}{\sqrt{2}}$$

Ed espandendo $|\psi_Q\rangle$ nella base computazionale:

$$(\alpha |0_Q\rangle + \beta |1_Q\rangle) \otimes \frac{|0_A 0_B\rangle + |1_A 1_B\rangle}{\sqrt{2}} = \alpha \frac{|0_Q 0_A 0_B\rangle + |0_Q 1_A 1_B\rangle}{\sqrt{2}} + \beta \frac{|1_Q 0_A 0_B\rangle + |1_Q 1_A 1_B\rangle}{\sqrt{2}}$$

Il passo successivo consiste nell'applicare prima una porta CNOT controllata da Q su A:

$$\alpha \frac{|0_Q 0_A 0_B\rangle + |0_Q 1_A 1_B\rangle}{\sqrt{2}} + \beta \frac{|1_Q 1_A 0_B\rangle + |1_Q 0_A 1_B\rangle}{\sqrt{2}} = \alpha |0_Q\rangle \otimes \frac{|0_A 0_B\rangle + |1_A 1_B\rangle}{\sqrt{2}} + \beta |1_Q\rangle \otimes \frac{|1_A 0_B\rangle + |0_A 1_B\rangle}{\sqrt{2}}$$

e in seguito un gate di Hadamard sul qubit Q:

$$\alpha\frac{|0_Q\rangle + |1_Q\rangle}{\sqrt{2}} \otimes \frac{|0_A0_B\rangle + |1_A1_B\rangle}{\sqrt{2}} + \beta\frac{|0_Q\rangle - |1_Q\rangle}{\sqrt{2}} \otimes \frac{|1_A0_B\rangle + |0_A1_B\rangle}{\sqrt{2}}$$

Infine separando Q e A da B tramite raccoglimento:

$$\begin{aligned}&\frac{1}{2}|0_Q0_A\rangle(\alpha|0_B\rangle + \beta|1_B\rangle) + \frac{1}{2}|0_Q1_A\rangle(\beta|0_B\rangle + \alpha|1_B\rangle) + \\ &\frac{1}{2}|1_Q0_A\rangle(\alpha|0_B\rangle - \beta|1_B\rangle) + \frac{1}{2}|1_Q1_A\rangle(-\beta|0_B\rangle + \alpha|1_B\rangle)\end{aligned} \tag{3.1}$$

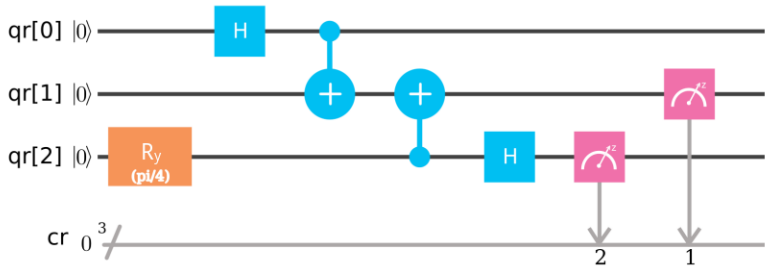


Figura 3.1: Passi 1 e 2 del teletrasporto quantistico

Misurazione della coppia QA e conclusione del teletrasporto

Osservando la 3.1 si nota la vicinanza con il risultato cercato: ciò che rimane da fare è una misura nella base computazionale della coppia di qubit QA, che avrà il risultato di far collassare lo stato dell'intero sistema in uno dei quattro termini possibili. Se l'ente $\hat{A}$, che detiene la misura q_Qq_A, la comunica a $\hat{B}$[1], $\hat{B}$ sa in quale stato si trova il suo sistema e può applicare l'opportuno operatore per raggiungere lo stato che aveva inizialmente Q. È semplice vedere che la corrispondenza tra misura q_Qq_A ottenuta e gate da applicare a B è la seguente:

$$0_Q0_A \to \mathcal{I} \qquad 0_Q1_A \to \mathcal{I} \qquad 1_Q0_A \to \mathcal{Z} \qquad 1_Q1_A \to \mathcal{Y}$$

dopodiché il qubit B si troverà nello stato $\alpha|0_Q\rangle + \beta|1_Q\rangle$, che era proprio lo stato posseduto da $\hat{A}$ e conservato inizialmente in Q che si voleva trasferire. Si osserva però

[1]Tale informazione è di tipo classico, quindi dovrà essere usato un usuale canale di comunicazione non quantistico.

che questa procedura ha modificato irrimediabilmente il qubit Q (a questo punto si trova in uno dei due possibili vettori della base computazionale), tutta l'informazione che conteneva è stata 'distrutta' perché spostata in B: una trasmissione del genere permette lo spostamento ma non la copia di uno stato quantistico.

3.1.1 Realizzazione del circuito

Lo stato di partenza scelto per il trasferimento è quello ottenuto applicando un gate $R_y(\frac{\pi}{4})$ (vedi 1.11) allo stato fondamentale, ovvero:

$$|q_Q\rangle = R_y\left(\frac{\pi}{4}\right)|0\rangle = \cos\left(\frac{\pi}{8}\right)|0\rangle + \sin\left(\frac{\pi}{8}\right)|1\rangle$$

per cui la statistica di misura nella base computazionale prevista è ~ 0.85355 per lo stato $|0\rangle$ e ~ 0.14645 per lo stato $|1\rangle$. Poi si procede con l'implementazione di tutte le operazioni descritte[2]:

```
import math
from qiskit import QuantumProgram
from qiskit.tools.visualization import plot_histogram

qp = QuantumProgram()

n = 3
qr = qp.create_quantum_register("qr", n)
cr = qp.create_classical_register("cr", n)

qc = qp.create_circuit("teletrasporto", [qr], [cr])

# 0) Creazione dello stato da "trasmettere" (Q)
qc.ry(math.pi/4, qr[2])

# 1) Creazione stato di Bell
qc.h(qr[0])
qc.cx(qr[0], qr[1])

# 2) "Cuore" del teletrasporto
qc.cx(qr[2], qr[1])
qc.h(qr[2])

# 3) Misura di QA e relativa azione su B
qc.measure(qr[2], cr[2])
qc.measure(qr[1], cr[1])

qc.x(qr[0]).c_if(cr, 0b010)
```

[2]Per questo circuito non è possibile usare i backend reali non essendo ancora state implementate le operazioni *controlled-if* richieste.

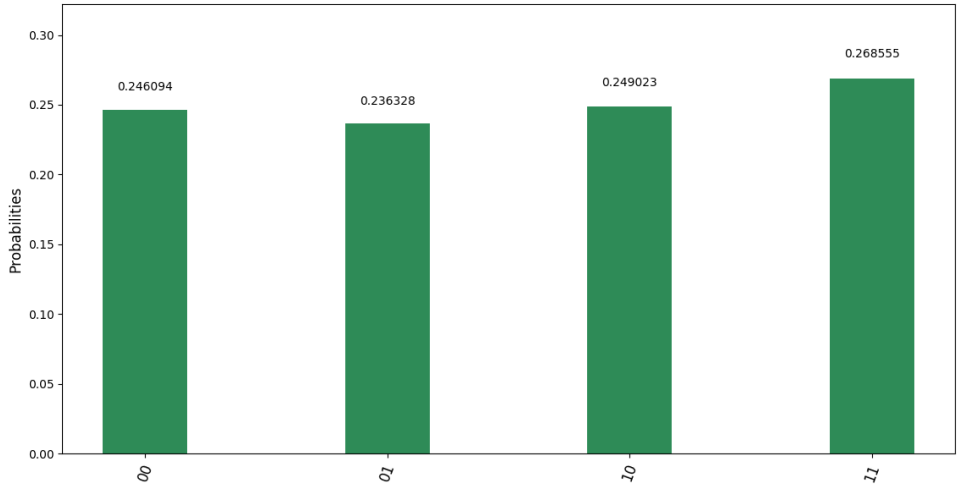
```
qc.z(qr[0]).c_if(cr, 0b100)
qc.y(qr[0]).c_if(cr, 0b110)

# L'algoritmo è concluso, si misura lo stato di B
qc.measure(qr[0], cr[0])

result = qp.execute(["teletrasporto"], backend="local_qasm_simulator",
    shots=1024)

print(qp.get_qasm("teletrasporto"))
print(result)
data = result.get_counts("teletrasporto")

# Si accorpano tutti gli stati in base al valore di QA
QA = {}
QA["00"] = data["000"] + data["001"]
QA["01"] = data["010"] + data["011"]
QA["10"] = data["100"] + data["101"]
QA["11"] = data["110"] + data["111"]
plot_histogram(QA)

# Si accorpano tutti gli stati in base al valore di B
B = {}
B["0"] = data["000"] + data["010"] + data["100"] + data["110"]
B["1"] = data["001"] + data["011"] + data["101"] + data["111"]
plot_histogram(B)
```

Output del codice:

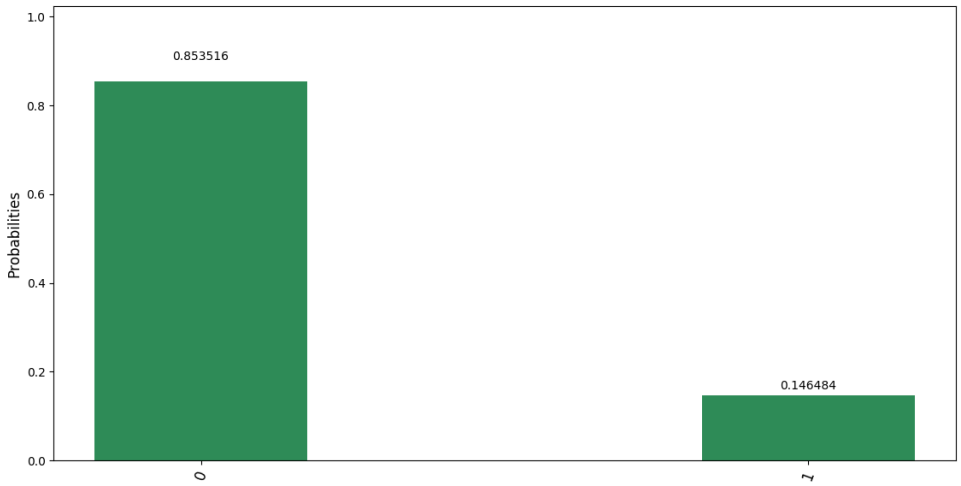
```
OPENQASM 2.0;
include "qelib1.inc";
qreg qr[3];
creg cr[3];
ry(0.785398163397448) qr[2];
h qr[0];
cx qr[0],qr[1];
cx qr[2],qr[1];
h qr[2];
measure qr[2] -> cr[2];
measure qr[1] -> cr[1];
if(cr==2) x qr[0];
if(cr==4) z qr[0];
if(cr==6) y qr[0];
measure qr[0] -> cr[0];

COMPLETED
```

La statistica ottenuta è raffigurata nella 3.2 e concorda con le previsioni teoriche.

Figura 3.2: Distribuzione statistica degli stati finali su un totale di 1024 esecuzioni



(a) Frequenze dello stato finale dei qubit Q e A



(b) Frequenze dello stato finale del qubit B

3.2 Trasformata di Fourier

3.2.1 Cenni alla trasformata di Fourier discreta

La *trasformata di Fourier discreta* (abbreviata in DFT dall'inglese *discrete Fourier transform*) è un'applicazione che trasforma la N-upla di numeri complessi $(x_0, ..., x_{N-1})$ nella N-upla di numeri complessi $(y_0, ..., y_{N-1})$ definita nel modo seguente:

$$y_k = \frac{1}{N} \sum_{j=0}^{N-1} x_j e^{-i\frac{2\pi}{N}kj} \tag{3.2}$$

Si riportano invece le espressioni della trasformata di Fourier diretta e inversa nel continuo, valide per un'ampia classe di funzioni (le funzioni a decrescenza rapida):

$$\tilde{f}(p) = \int_{-\infty}^{+\infty} f(x)e^{-ipx}dx \tag{3.3}$$

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \tilde{f}(p)e^{ipx}dp \tag{3.4}$$

Si cerca ora di investigare la relazione tra le due trasformate, ovvero il passaggio dal continuo al discreto che rende possibile computare una approssimazione della trasformata al calcolatore. Partendo dalla 3.3 si può scrivere:

$$\tilde{f}(p) = \lim_{L\to\infty}\int_{-L/2}^{L/2} f(x)e^{-ipx}dx = \lim_{L\to\infty}\tilde{f}_L(p) = \lim_{L\to\infty}\int_{-\infty}^{+\infty} f_L(x)e^{-ipx}dx \tag{3.5}$$

nella quale si è introdotta la funzione:

$$f_L(x) = \begin{cases} f(x) & \text{se } -\frac{L}{2} \le x < \frac{L}{2} \\ 0 & \text{altrove} \end{cases} \tag{3.6}$$

In questo modo $\tilde{f}(p) \approx \tilde{f}_L(p)$ per L molto grande. Ora si può pensare $f_L(x)$ come funzione appartenente a $\mathcal{L}^2\left(-\frac{L}{2}, \frac{L}{2}\right)$ e di conseguenza espanderla in serie di Fourier:

$$f_L(x) = \sum_{k=-\infty}^{+\infty} \tilde{f}_k e^{ip_k x} \qquad \text{con} \qquad p_k = \frac{2\pi}{L}k \tag{3.7}$$

che risulta nella discretizzazione della variabile p secondo multipli interi di $\frac{2\pi}{L}$. I relativi coefficienti di Fourier saranno dunque:

$$\tilde{f}_k = \frac{1}{L}\int_{-L/2}^{L/2} f(x)e^{-ip_k x}dx = \frac{1}{L}\tilde{f}_L(p_k) \tag{3.8}$$

Ora va compiuta un'ulteriore discretizzazione per valutare numericamente l'integrale in $\tilde{f}_L(p)$, che consiste nel suddividere l'intervallo $\left[-\frac{L}{2}, \frac{L}{2}\right]$ in sottointervalli di estensione a:

$$\tilde{f}_L(p) = \lim_{a\to 0}\sum_{-L/2\le ja<L/2} f(x_j)e^{-ipx_j}a \qquad \text{con} \qquad x_j = ja \tag{3.9}$$

In questo modo si ottiene una somma su un numero finito di intervalli $N = \left\lceil \frac{L}{a} \right\rceil$ (primo intero $\ge \frac{L}{a}$, cioè approssimazione intera per eccesso di $\frac{L}{a}$):

$$\tilde{f}_{L,a}(p) = \sum_{-N/2\le j<N/2} f(x_j)e^{-ipx_j}a \tag{3.10}$$

A questo punto si nota che non tutte le frequenze p sono indipendenti, infatti:

$$e^{-ipx_j} = e^{-ipja} = e^{-i\left(p+\frac{2\pi}{a}\right)ja} \tag{3.11}$$

e le frequenze indipendenti stanno in un intervallo di ampiezza $\frac{2\pi}{a}$. Per valori fuori da tale intervallo vale $\tilde{f}_{L,a}(p) = \tilde{f}_{L,a}(p + \frac{2\pi}{a})$. Una scelta possibile è $-\frac{\pi}{a} \leq p < \frac{\pi}{a}$, indicata con il termine 'prima zona di Brillouin'. Combinando tale restrizione con la condizione data dalla 3.7:

$$-\frac{\pi}{a} \leq p_k = \frac{2\pi}{L}k < \frac{\pi}{a}$$

Per cui anche il numero di p_k è finito e vale $N = \left\lceil \frac{L}{a} \right\rceil$. La 3.2 risulta ora giustificata: il fattore di normalizzazione N ha origine dalla combinazione di fattori $\frac{a}{L}$, mentre l'argomento dell'esponenziale complesso deriva da:

$$p_k x_j = \frac{2\pi}{L}akj = \frac{2\pi}{N}kj$$

3.2.2 La trasformata di Fourier quantistica

Per gli scopi di questa sezione si utilizza una definizione alternativa della DFT, che differisce dalla 3.2 per il fattore di normalizzazione impiegato e il segno nell'argomento dell'esponenziale:

$$y_l = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_k e^{2\pi ikl/N} \tag{3.12}$$

La (3.12) può essere vista come il risultato dell'azione di un operatore lineare $\mathcal{F}$ sulle coordinate di elementi di uno spazio vettoriale di dimensione N. Supponiamo inoltre che sia definito un prodotto scalare sullo spazio e che la base scelta sia ortonormale, in modo da poter indicare le coordinate del vettore generico $|x\rangle$ come $\langle k|x\rangle$. In questa notazione la (3.2) diventa:

$$\langle l| \mathcal{F} |x\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \langle k|x\rangle \, e^{2\pi ikl/N} \tag{3.13}$$

dalla quale è immediato ricavare gli elementi di matrice di $\mathcal{F}$:

$$\langle l| \mathcal{F} |j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \langle k|j\rangle \, e^{2\pi ikl/N} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \delta_{kj} e^{2\pi ikl/N} = \frac{1}{\sqrt{N}} e^{2\pi ijl/N} \tag{3.14}$$

Dunque l'azione di $\mathcal{F}$ sulla base $|j\rangle$ risulta:

$$\mathcal{F} |j\rangle = \sum_{k=0}^{N-1} |k\rangle \langle k| \mathcal{F} |j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle \, e^{2\pi ijk/N} \tag{3.15}$$

N = 1

Se $N = 1$, $\mathcal{F}$ è rappresentata da una matrice di un singolo elemento di valore 1, e perciò si ha banalmente $\mathcal{F}|0\rangle = |0\rangle$.

N = 2

Se $N = 2$, lo spazio su cui viene effettuata $\mathcal{F}$ è descritto da un singolo qubit. Riprendendo la definizione:

$$\mathcal{F}|j_0\rangle = \frac{1}{\sqrt{2}} \sum_{k_0=0}^{1} e^{i2\pi j_0 k/2}$$

Espandendo la formula si ottiene la seguente azione sulla base:

$$\mathcal{F}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad \mathcal{F}|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\pi}|1\rangle\right) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

La matrice corrispondente è:

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & e^{i\pi} \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

ovvero il gate di Hadamard.

N = 4

Se $N = 4$, il numero n di qubit necessari per descrivere lo spazio diventa 2. Si pone per convenienza $\omega = e^{i2\pi\frac{1}{N}}$ (in questo modo possiamo esprimere ciascun elemento di matrice come potenza di ω) e tenendo conto che $\forall p \in \mathbb{N}, \omega^p = \omega^{p \bmod N}$ per la periodicità dell'esponenziale complesso, la matrice risulta:

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & 1 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

A questo punto è possibile ricavare le coordinate delle immagini di ciascun vettore della base canonica dalle colonne della matrice. Ad esempio per lo stato $|00\rangle$:

$$\mathcal{F}|00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

In vista di ottenere un algoritmo quantistico in grado di realizzare la QFT è necessario operare una scomposizione della formula precedente in termini dei singoli qubit che

compongono lo stato. In altre parole, deve essere espressa in forma di prodotto tensoriale:

$$\mathcal{F}|00\rangle = \frac{1}{2}(|0\rangle\otimes|0\rangle + |0\rangle\otimes|1\rangle + |1\rangle\otimes|0\rangle + |1\rangle\otimes|1\rangle) =$$
$$\frac{1}{2}(|0\rangle\otimes(|0\rangle+|1\rangle) + |1\rangle\otimes(|0\rangle+|1\rangle)) = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle) =$$
$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi 0}|1\rangle\right)\otimes\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi 0}|1\rangle\right)$$

Applicando lo stesso procedimento su ciascun vettore di base:

$$\mathcal{F}|01\rangle = \frac{1}{2}\left(|00\rangle + e^{i2\pi\frac{1}{4}}|01\rangle + e^{i2\pi\frac{2}{4}}|10\rangle + e^{i2\pi\frac{3}{4}}|11\rangle\right) =$$
$$\frac{1}{2}\left(|00\rangle + e^{i2\pi\frac{1}{4}}|01\rangle + e^{i2\pi\frac{1}{2}}\left(|10\rangle + e^{i2\pi\frac{1}{4}}|11\rangle\right)\right) =$$
$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi\frac{1}{2}}|1\rangle\right)\otimes\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi\frac{1}{4}}|1\rangle\right)$$

$$\mathcal{F}|10\rangle = \frac{1}{2}(|0\rangle\otimes|0\rangle - |0\rangle\otimes|1\rangle + |1\rangle\otimes|0\rangle - |1\rangle\otimes|1\rangle) =$$
$$\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle) = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi 0}|1\rangle\right)\otimes\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi\frac{1}{2}}|1\rangle\right)$$

$$\mathcal{F}|11\rangle = \frac{1}{2}\left(|00\rangle + e^{i2\pi\frac{3}{4}}|01\rangle + e^{i2\pi\frac{6}{4}}|10\rangle + e^{i2\pi\frac{9}{4}}|11\rangle\right) =$$
$$\frac{1}{2}\left(|00\rangle + e^{i2\pi\frac{3}{4}}|01\rangle + e^{i2\pi\frac{1}{2}}\left(|10\rangle + e^{i2\pi\frac{3}{4}}|11\rangle\right)\right) =$$
$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi\frac{1}{2}}|1\rangle\right)\otimes\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi\frac{3}{4}}|1\rangle\right)$$

N = 8

Se $N = 8$, il numero n di qubit necessari per descrivere lo spazio diventa 3. In maniera analoga al caso precedente, si valuta la matrice associata:

$$\frac{1}{2\sqrt{2}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix} = \frac{1}{2\sqrt{2}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{i\frac{1}{4}} & i & e^{i\frac{3}{4}} & -1 & e^{i\frac{5}{4}} & -i & e^{i\frac{7}{4}} \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & e^{i\frac{3}{4}} & -i & e^{i\frac{1}{4}} & -1 & e^{i\frac{7}{4}} & i & e^{i\frac{5}{4}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & e^{i\frac{5}{4}} & i & e^{i\frac{7}{4}} & -1 & e^{i\frac{1}{4}} & -i & e^{i\frac{3}{4}} \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & e^{i\frac{7}{4}} & -i & e^{i\frac{5}{4}} & -1 & e^{i\frac{3}{4}} & i & e^{i\frac{1}{4}} \end{bmatrix}$$

$$\mathcal{F}\left|j\right\rangle = \frac{1}{2\sqrt{2}} \sum_{k=0}^{7} e^{i2\pi \frac{jk}{2^3}} \left|k\right\rangle$$

$$\begin{aligned}
\mathcal{F}\left|j\right\rangle &= \frac{1}{2\sqrt{2}} \sum_{k_2=0}^{1} \sum_{k_1=0}^{1} \sum_{k_0=0}^{1} e^{i2\pi j\left(2^{-1}k_2 + 2^{-2}k_1 + 2^{-3}k_0\right)} \left|k_2 k_1 k_0\right\rangle \\
&= \frac{1}{2\sqrt{2}} \sum_{k_2=0}^{1} \sum_{k_1=0}^{1} \sum_{k_0=0}^{1} e^{i2\pi j 2^{-1}k_2} \left|k_2\right\rangle \otimes e^{i2\pi j 2^{-2}k_1} \left|k_1\right\rangle \otimes e^{i2\pi j 2^{-3}k_0} \left|k_0\right\rangle \\
&= \frac{1}{\sqrt{2}} \sum_{k_2=0}^{1} e^{i2\pi j 2^{-1}k_2} \left|k_2\right\rangle \otimes \frac{1}{\sqrt{2}} \sum_{k_1=0}^{1} e^{i2\pi j 2^{-2}k_1} \left|k_1\right\rangle \otimes \frac{1}{\sqrt{2}} \sum_{k_0=0}^{1} e^{i2\pi j 2^{-3}k_0} \left|k_0\right\rangle
\end{aligned}$$

Espandendo le somme, lo stato finale dei singoli qubit che compongono il risultato è:

$$\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi 2^{-1}j}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi 2^{-2}j}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi 2^{-3}j}\left|1\right\rangle\right) \tag{3.16}$$

Ora si consideri il caso in cui tutti i qubit dello stato in input siano spenti tranne il bit più significativo, che viene lasciato variabile: naturalmente ciò equivale a porre $j = 2^2 j_2$, e sostituendo questa espressione nella formula 3.16 si ottiene:

$$\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi 2^{1}j_2}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi 2^{0}j_2}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi 2^{-1}j_2}\left|1\right\rangle\right)$$

Anche in questo caso si nota che, a causa della periodicità, il bit più significativo in input influenza soltanto il bit meno signficativo in output, e lo fa sotto forma di una trasformazione di Hadamard. Riconsiderando anche il comportamento della QFT nel caso $n = 2$, si potrebbe pensare che i bit in input influenzino progressivamente quelli in output, ma in ordine inverso. Questo fatto viene confermato dalla formula che sarà ricavata per un generico $j = 2^2 j_2 + 2^1 j_1 + 2^0 j_0$. Partendo dalla formula 3.16:

$$\begin{aligned}
&\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi(2j_2 + j_1 + j_0/2)}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi(j_2 + j_1/2 + j_0/4)}\left|1\right\rangle\right) \otimes \\
&\qquad \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi(j_2/2 + j_1/4 + j_0/8)}\left|1\right\rangle\right) = \\
&\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi(j_0/2)}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi(j_1/2 + j_0/4)}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi(j_2/2 + j_1/4 + j_0/8)}\left|1\right\rangle\right) \\
&\qquad = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi 0.j_0}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi 0.j_1 j_0}\left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{i2\pi 0.j_2 j_1 j_0}\left|1\right\rangle\right)
\end{aligned}$$

dove nell'ultimo passaggio si è introdotta una notazione molto comoda chiamata *frazione binaria* definita in questo modo: $0.j_p \ldots j_0 = \sum_{k=0}^{p} 2^{-(p-k+1)} j_k$.

Caso generale

Nel caso generale (in cui $N = 2^n$) lo stato N-dimensionale viene rappresentato da un array di n qubit. Analogamente alle situazioni precedenti, è necessario scomporre la trasformazione nella azione sui singoli qubit, ovvero esprimerla in termini del loro prodotto tensoriale. Partendo dalla 3.15, il procedimento è il seguente:

$$\mathcal{F}\,|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle\, e^{i2\pi jk/N} \tag{3.17}$$

Operando la sostituzione $N = 2^n$:

$$\frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} |k\rangle\, e^{i2\pi jk2^{-n}} \tag{3.18}$$

Esprimendo k in base 2 si ha $k = \sum_{l=0}^{n-1} k_l 2^l$ e la somma può essere scritta come n somme su ciascuna cifra binaria:

$$\frac{1}{2^{n/2}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=0}^{1} |k_{n-1} \ldots k_0\rangle\, e^{i2\pi j \sum_{l=0}^{n-1} k_l 2^{l-n}} \tag{3.19}$$

Sfruttando la proprietà dell'esponenziale di trasformare somme in prodotti ed esplicitando $|k_{n-1} \ldots k_0\rangle = |k_{n-1}\rangle \otimes \cdots \otimes |k_0\rangle$:

$$\begin{aligned}
&\frac{1}{2^{n/2}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=0}^{1} \bigotimes_{l=n-1}^{0} |k_l\rangle\, e^{i2\pi jk_l 2^{l-n}} = \\
&\qquad \frac{1}{2^{n/2}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=0}^{1} \bigotimes_{l=n-1}^{1} |k_l\rangle\, e^{i2\pi jk_l 2^{l-n}} \otimes |k_0\rangle\, e^{i2\pi jk_0 2^{-n}} = \\
&\qquad\qquad \frac{1}{2^{n/2}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=1}^{1} \bigotimes_{l=n-1}^{1} |k_l\rangle\, e^{i2\pi jk_l 2^{l-n}} \otimes \sum_{k_0=0}^{1} |k_0\rangle\, e^{i2\pi jk_0 2^{-n}}
\end{aligned} \tag{3.20}$$

Iterando i passaggi precedenti per ciascuna delle n cifre si ottiene:

$$\frac{1}{2^{n/2}} \bigotimes_{l=n-1}^{0} \sum_{k_l=0}^{1} |k_l\rangle\, e^{i2\pi jk_l 2^{l-n}} = \frac{1}{2^{n/2}} \bigotimes_{l=n-1}^{0} \left(|0\rangle + |1\rangle\, e^{i2\pi j 2^{l-n}}\right) \tag{3.21}$$

Esprimendo j in base 2 si scrive $2\pi \sum_{r=0}^{n-1} j_r 2^{r+l-n}$ il fattore di fase dell'esponenziale complesso, ma le potenze di due non negative forniscono contributo nullo a causa della periodicità. I termini della somma che sopravvivono sono quelli con $r < n-l$, ad esempio per $l = n-1$ si avrà $r = 0$, mentre per $l = 0$ si avrà $r = 0, \ldots, n-1$. Si nota che in questo modo ad ogni valore di l corrisponderà la frazione binaria $0.j_{n-l-1} \ldots j_0$:

$$\frac{1}{2^{n/2}} \left(|0\rangle + |1\rangle\, e^{i2\pi 0.j_0}\right) \otimes \left(|0\rangle + |1\rangle\, e^{i2\pi 0.j_1 j_0}\right) \otimes \cdots \otimes \left(|0\rangle + |1\rangle\, e^{i2\pi 0.j_{n-1} \ldots j_0}\right) \tag{3.22}$$

3.2.3 Descrizione dell'algoritmo e realizzazione del circuito

Una volta ottenuta una forma fattorizzata per la QFT, si devono individuare le giuste operazioni da applicare ai qubit per realizzare lo stato trasformato. Osservando la 3.22, si nota che lo stato trasformato di ciascun qubit dipende anche dagli stati degli altri: ciò significa che è necessario fare uso di gate controllati e fare attenzione all'ordine con cui si eseguono le operazioni.
Il primo qubit da modificare sarà il MSB (ovvero j_{n-1}), poiché il suo stato influenza un unico qubit trasformato. Applicandogli un gate di Hadamard si ottiene il seguente stato:

$$\mathcal{H}\left|q_{n-1}\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle + e^{i\pi q_{n-1}}\left|1\right\rangle) = \frac{1}{\sqrt{2}}(\left|0\right\rangle + e^{i2\pi q_{n-1}/2}\left|1\right\rangle) = \frac{1}{\sqrt{2}}(\left|0\right\rangle + e^{i2\pi 0.q_{n-1}}\left|1\right\rangle)$$

dove si è usata una notazione compatta per rappresentare i due possibili casi $q_{n-1} = 0, 1$ in un'unica espressione. A questo punto occorre impiegare una porta quantistica controllata per aggiungere i termini dipendenti dagli altri qubit, la quale rappresenta una generalizzazione dei gate S e T descritti nella sezione 1.3:

$$U1(\theta) \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

applicando una versione controllata dal qubit q_{n-2} con $\theta = \frac{\pi}{2}$ allo stato precedentemente ottenuto si ha:

$$\mathcal{CU}1(\pi/2)\mathcal{H}\left|q_{n-1}\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle + e^{i2\pi(0.q_{n-1}+q_{n-2}/4)}\left|1\right\rangle) = \frac{1}{\sqrt{2}}(\left|0\right\rangle + e^{i2\pi 0.q_{n-1}q_{n-2}}\left|1\right\rangle)$$

Per ottenere l'intera frazione binaria si continua ad applicare lo stesso operatore controllato da ognuno dei qubit che seguono q_{n-1}, con θ ottenuto dividendo π per potenze di due successive. Ripetendo l'intero procedimento (Hadamard + $CU1$ in serie) per tutti gli altri qubit si giunge a:

$$\frac{1}{2^{n/2}}\left(\left|0\right\rangle + \left|1\right\rangle e^{i2\pi 0.q_{n-1}\ldots q_0}\right) \otimes \left(\left|0\right\rangle + \left|1\right\rangle e^{i2\pi 0.q_{n-2}\ldots q_0}\right) \otimes \cdots \otimes \left(\left|0\right\rangle + \left|1\right\rangle e^{i2\pi 0.q_0}\right)$$

ma il risultato non è ancora lo stato 3.22. Per ottenere effettivamente la QFT occorre scambiare l'ordine dei qubit (nel senso di 'rovesciare' l'array di qubit) tramite $n/2$ operazioni[3] di swap. Nella 3.3 si raffigura il circuito completo, dove gli operatori di swap sono stati sostituiti dall'inversione dell'ordine di misurazione.
Si calcoli ora la complessità computazionale dell'algoritmo in termini di porte utilizzate: si inizia con un gate di Hadamard e $n-1$ rotazioni controllate sul MSB, per un totale di n gate. Poi un gate di Hadamard e $n-2$ rotazioni controllate sul qubit successivo, per un totale di $n + (n-1)$ gate. Continuando in questo modo, si mostra che sono richieste

[3] $(n-1)/2$ se n è dispari.

$n + (n-1) + \cdots + 1 = n(n+1)/2$ porte (più quelle da impiegare per gli swap). Ciò significa che questo algoritmo per la trasformata di Fourier è di ordine n^2.
Il miglior algoritmo classico che realizza lo stesso scopo su un array di 2^n elementi è la *Fast Fourier Transform* (FFT), di ordine $n2^n$: pertanto, è richiesto un numero di operazioni esponenzialmente maggiore in n per computare la trasformata di Fourier in un computer classico rispetto al caso quantistico.

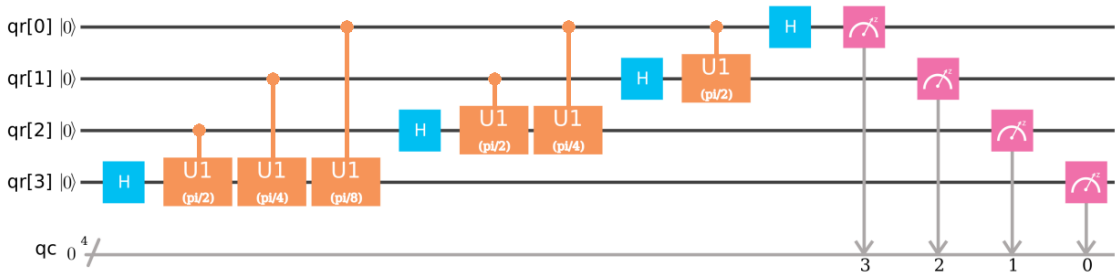


Figura 3.3: Circuito che realizza la QFT a 4 qubit

```
import math
from qiskit import QuantumProgram

qp = QuantumProgram()

n = 4
qr = qp.create_quantum_register("qr", n)
cr = qp.create_classical_register("qc", n)

qc = qp.create_circuit("qft", [qr], [cr])

for i in range(n-1, -1, -1):
  qc.h(qr[i])
  for j in range(i-1, -1, -1):
    qc.cu1(math.pi/(2**(i-j)), qr[j], qr[i])
for i in range(n // 2):
  qc.swap(qr[i], qr[n-1-i])

qc.measure(qr, cr)

print(qp.get_qasm("qft"))
result = qp.execute(["qft"], backend="local_qasm_simulator", shots
    =1024)
print(result.get_data("qft"))
```

```
OPENQASM 2.0;
```

```
include "qelib1.inc";
qreg qr[4];
creg qc[4];
h qr[3];
cu1(1.57079632679490) qr[2],qr[3];
cu1(0.785398163397448) qr[1],qr[3];
cu1(0.392699081698724) qr[0],qr[3];
h qr[2];
cu1(1.57079632679490) qr[1],qr[2];
cu1(0.785398163397448) qr[0],qr[2];
h qr[1];
cu1(1.57079632679490) qr[0],qr[1];
h qr[0];
swap qr[0],qr[3];
swap qr[1],qr[2];
measure qr[0] -> qc[0];
measure qr[1] -> qc[1];
measure qr[2] -> qc[2];
measure qr[3] -> qc[3];

{'counts': {'0011': 53, '0000': 59, '0111': 75, '1010': 56, '1011': 70,
    '0101': 59, '0001': 70, '1110': 72, '0100': 59, '0010': 48, '1001':
    71, '1101': 68, '1100': 67, '1111': 56, '0110': 79, '1000': 62}}
```

Nelle prossime sezioni verranno descritti dei brevi programmi che fanno uso della QFT per testare la validità del codice appena descritto. Per ciascuno di essi si effettuerà prima il calcolo teorico dei risultati attesi, che saranno confrontati con i risultati forniti dal programma eseguito sul simulatore locale di QISKit.

Costruire uno stato la cui trasformata è un dato stato

Trasformata di Fourier quantistica inversa:

$$\mathcal{F}\left|j\right\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i2\pi(-jk)/N} \left|k\right\rangle$$

scomposta nei singoli qubit diventa:

$$\frac{1}{2^{n/2}} \left(\left|0\right\rangle + \left|1\right\rangle e^{i2\pi(-0.j_0)}\right) \otimes \cdots \otimes \left(\left|0\right\rangle + \left|1\right\rangle e^{i2\pi(-0.j_{n-1}\ldots j_0)}\right)$$

Caso in cui lo stato da ottenere è $\left|001\right\rangle$:

$$\frac{1}{\sqrt{2}} \left(\left|0\right\rangle + e^{i2\pi(-0.1)} \left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}} \left(\left|0\right\rangle + e^{i2\pi(-0.01)} \left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}} \left(\left|0\right\rangle + e^{i2\pi(-0.001)} \left|1\right\rangle\right) =$$
$$\frac{1}{\sqrt{2}} \left(\left|0\right\rangle + e^{i2\pi(-1/2)} \left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}} \left(\left|0\right\rangle + e^{i2\pi(-1/4)} \left|1\right\rangle\right) \otimes \frac{1}{\sqrt{2}} \left(\left|0\right\rangle + e^{i2\pi(-1/8)} \left|1\right\rangle\right)$$

Verifica che questo è lo stato cercato svolgendo i prodotti:

$$\frac{1}{\sqrt{8}}(|000\rangle + e^{i2\pi(-1/8)}|001\rangle + e^{i2\pi(-2/8)}|010\rangle + e^{i2\pi(-3/8)}|011\rangle + e^{i2\pi(-4/8)}|100\rangle + e^{i2\pi(-5/8)}|101\rangle + e^{i2\pi(-6/8)}|110\rangle + e^{i2\pi(-7/8)}|111\rangle)$$

e applicando $\mathcal{F}$, in coordinate:

$$\frac{1}{8}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix} \begin{bmatrix} 1 \\ \omega^{-1} \\ \omega^{-2} \\ \omega^{-3} \\ \omega^{-4} \\ \omega^{-5} \\ \omega^{-6} \\ \omega^{-7} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

infatti gli addendi del prodotto matriciale sono tutti 1 nel caso della coordinata relativa a $|001\rangle$, mentre in tutti gli altri casi si nota la cancellazione a due a due di termini che, interpretati come punti sulla circonferenza unitaria, differiscono di un angolo piatto. Si possono calcolare esplicitamente tali somme in questo modo:

$$(1+\omega+\omega^2+\cdots+\omega^p)(1-\omega) = 1-\omega+\omega-\omega^2+\omega^2+\cdots+\omega^{p+1}$$

$$\sum_{k=0}^{p}\omega^k = \frac{1-\omega^{p+1}}{1-\omega} \rightarrow \sum_{k=0}^{7}\omega^k = \frac{1-\omega^8}{1-\omega} = \frac{1-1}{1-\omega} = 0$$

A questo punto si devono individuare le operazioni per poter costruire questo stato a partire da $|000\rangle$. I gate da applicare sono Hadamard più una rotazione di fase di un opportuno angolo negativo. Ad esempio per il LSB:

$$\mathcal{H}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad \text{e} \qquad \mathcal{U}1(-\frac{\pi}{4})\mathcal{H}|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi(-1/8)}|1\rangle\right)$$

Segue il codice, il circuito risultante è presentato in Figura 3.4:

```
import math
from qiskit import QuantumProgram
import Qconfig

qp = QuantumProgram()
qp.set_api(Qconfig.APItoken, Qconfig.config["url"])

n = 3
qr = qp.create_quantum_register("qr", n)
```

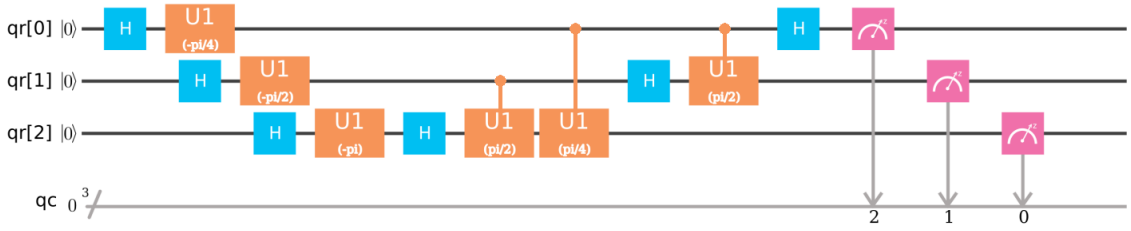


Figura 3.4: Un circuito di verifica per la QFT

```
cr = qp.create_classical_register("qc", n)

qc = qp.create_circuit("qft", [qr], [cr])

# Creazione dello stato che, trasformato, dovrebbe diventare 001
for i in range(n):
  qc.h(qr[i])
  qc.u1(-math.pi/(2**(n-1-i)), qr[i])

# Applicazione della qft
for i in range(n-1, -1, -1):
  qc.h(qr[i])
  for j in range(i-1, -1, -1):
    qc.cu1(math.pi/(2**(i-j)), qr[j], qr[i])
for i in range(n // 2):
  qc.swap(qr[i], qr[n-1-i])

qc.measure(qr, cr)

print(qp.get_qasm("qft"))
result = qp.execute(["qft"], backend="local_qasm_simulator", shots
    =1024)
print(result.get_data("qft"))
```

L'output del codice conferma il risultato previsto:

```
OPENQASM 2.0;
include "qelib1.inc";
qreg qr[3];
creg qc[3];
h qr[0];
u1(-0.785398163397448) qr[0];
h qr[1];
u1(-1.57079632679490) qr[1];
h qr[2];
u1(-3.14159265358979) qr[2];
h qr[2];
```

```
cu1(1.57079632679490) qr[1],qr[2];
cu1(0.785398163397448) qr[0],qr[2];
h qr[1];
cu1(1.57079632679490) qr[0],qr[1];
h qr[0];
swap qr[0],qr[2];
measure qr[0] -> qc[0];
measure qr[1] -> qc[1];
measure qr[2] -> qc[2];

{'counts': {'001': 1024}}
```

Si esegue questo programma anche su uno dei processori reali (l'ibmqx2 è sufficiente in quanto possiede 5 qubit), ricordando che in questo caso ci saranno deviazioni dal risultato ideale. L'output generato è riportato di seguito e nell'istogramma di Figura 3.5:

```
{'time': 22.692749977111816, 'counts': {'00000': 114, '00001': 548,
   '00010': 32, '00011': 98, '00100': 64, '00101': 75, '00110': 34,
   '00111': 59}, 'date': '2018-02-23T16:26:52.451Z'}
```

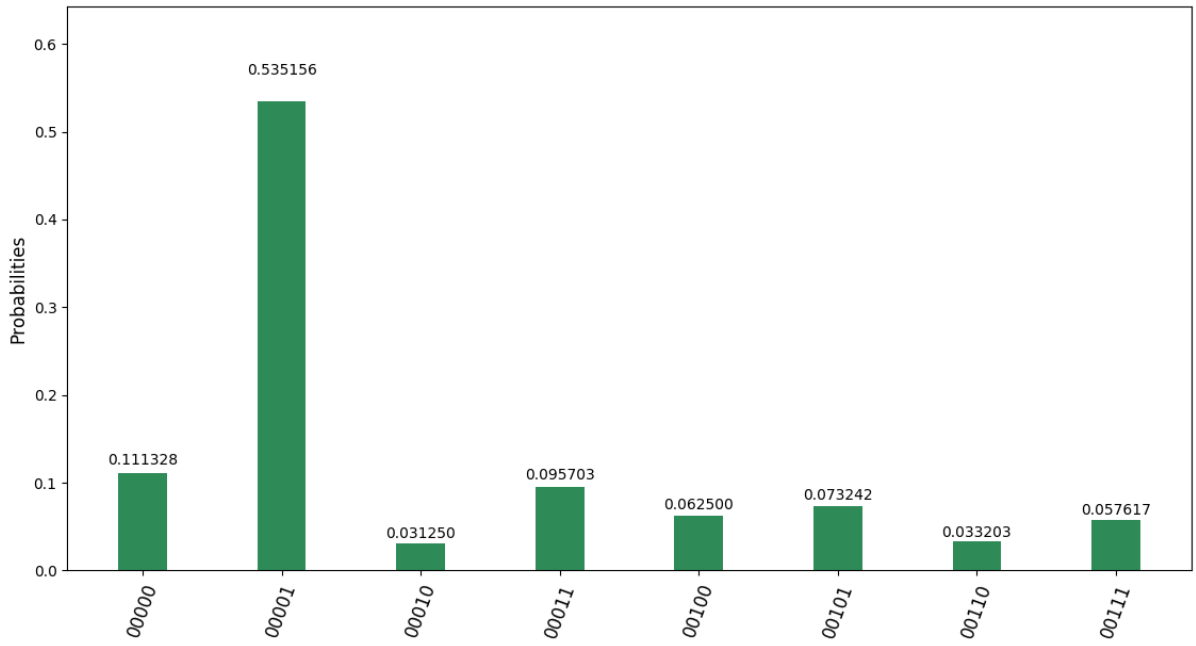


Figura 3.5: Distribuzione statistica dello stato finale

Trasformata della trasformata

Si osserva cosa comporta applicare due volte la QFT allo stato $|j\rangle$:

$$\mathcal{F}\mathcal{F}|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{i2\pi jk/N}\mathcal{F}|k\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{i2\pi jk/N}\left(\frac{1}{\sqrt{N}}\sum_{l=0}^{N-1} e^{i2\pi kl/N}|l\rangle\right) = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\frac{1}{N}e^{i2\pi jk/N}e^{i2\pi kl/N}|l\rangle = \sum_{l=0}^{N-1}\left(\sum_{k=0}^{N-1}\frac{1}{N}e^{i2\pi k(j+l)/N}\right)|l\rangle$$

A questo punto se $j + l = N$, ovvero se $l = N - j$ l'argomento dell'esponenziale si semplifica in $i2\pi k$. Dunque per ogni valore di k l'esponenziale avrà valore 1, che sommato sugli N valori di k fornisce N. Ma dividendo per la costante di normalizzazione N si ottiene 1 come coefficiente relativo all'elemento[4] $|N - j\rangle$, e dato che la trasformazione è unitaria nessun'altra componente potrà essere diversa da 0, altrimenti la norma del vettore immagine non sarebbe 1. In conclusione applicare due volte la QFT porta lo stato $|j\rangle$ nello stato $|N - j\rangle$ (e lascia invariato lo stato $|0\rangle$). Si riporta il codice Python per il circuito:

```
import math
from qiskit import QuantumProgram

# Definizione della qft
def qft(qc, qr, n):
  for i in range(n-1, -1, -1):
    qc.h(qr[i])
    for j in range(i-1, -1, -1):
      qc.cu1(math.pi/(2**(i-j)), qr[j], qr[i])
  for i in range(n // 2):
    qc.swap(qr[i], qr[n-1-i])

qp = QuantumProgram()
n = 4
qr = qp.create_quantum_register("qr", n)
cr = qp.create_classical_register("qc", n)
qc = qp.create_circuit("qft", [qr], [cr])

qc.x(qr[0])
qc.x(qr[2])
qft(qc, qr, n)
qft(qc, qr, n)

qc.measure(qr, cr)

print(qp.get_qasm("qft"))
```

[4]Nel caso in cui j sia 0, l'espressione va considerata modulo N, ovvero il risultato è ancora il ket 0.

```
result = qp.execute(["qft"], backend="local_qasm_simulator", shots
    =1024)
print(result.get_data("qft"))
```

Output del codice:

```
OPENQASM 2.0;
include "qelib1.inc";
qreg qr[4];
creg qc[4];
x qr[0];
x qr[2];
h qr[3];
cu1(1.57079632679490) qr[2],qr[3];
cu1(0.785398163397448) qr[1],qr[3];
cu1(0.392699081698724) qr[0],qr[3];
h qr[2];
cu1(1.57079632679490) qr[1],qr[2];
cu1(0.785398163397448) qr[0],qr[2];
h qr[1];
cu1(1.57079632679490) qr[0],qr[1];
h qr[0];
swap qr[0],qr[3];
swap qr[1],qr[2];
h qr[3];
cu1(1.57079632679490) qr[2],qr[3];
cu1(0.785398163397448) qr[1],qr[3];
cu1(0.392699081698724) qr[0],qr[3];
h qr[2];
cu1(1.57079632679490) qr[1],qr[2];
cu1(0.785398163397448) qr[0],qr[2];
h qr[1];
cu1(1.57079632679490) qr[0],qr[1];
h qr[0];
swap qr[0],qr[3];
swap qr[1],qr[2];
measure qr[0] -> qc[0];
measure qr[1] -> qc[1];
measure qr[2] -> qc[2];
measure qr[3] -> qc[3];

{'counts': {'1011': 1024}}
```

3.3 Stima di fase

La trasformata di Fourier gioca un ruolo fondamentale all'interno di una procedura generale nota come *stima di fase*, che a sua volta è la chiave per molti algoritmi quantistici.

Dato un operatore unitario U avente un autovettore $|u\rangle$ di autovalore $e^{i2\pi\phi}$, lo scopo della procedura è di stimare ϕ, ovvero la fase dell'autovalore (si ricorda che autovalori di un operatore unitario hanno modulo unitario). Supponiamo inoltre di riuscire a preparare lo stato $|u\rangle$ e di riuscire ad eseguire delle operazioni controllate da U^{2^j}, dove j è un numero intero non negativo. La stima di fase quantistica utilizza due registri. Il primo registro contiene t qubit inizialmente nello stato $|0\rangle$ (in seguito si mostrerà che dal valore di t dipenderanno il numero di cifre con cui vogliamo ottenere il risultato e la probabilità di riuscita della procedura). Il secondo registro si trova inizialmente nello stato $|u\rangle$, e contiene un numero di qubit adeguato a contenere $|u\rangle$. L'algoritmo si articola in due stadi:

1. tutti i qubit del primo registro vengono messi nello stato di sovrapposizione (applicando gate di Hadamard), mentre al secondo registro vengono applicate operazioni U^{2^j}-controllate dal qubit q_j.

2. viene applicata la trasformata di Fourier inversa sul primo registro

Per il fatto che $|u\rangle$ è autovettore di U si ha:

$$\mathcal{U}^{2^j}|u\rangle = e^{i2\pi\phi}\mathcal{U}^{2^j-1}|u\rangle = e^{i2\pi 2^j\phi}|u\rangle$$

Dopo aver portato tutti i qubit del primo registro nello stato di sovrapposizione, si applica a $|u\rangle$ l'operatore U^{2^j} controllato dal qubit j-esimo:

$$C\mathcal{U}^{2^j}\left[\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\otimes|u\rangle\right] = C\mathcal{U}^{2^j}\left(\frac{|0u\rangle+|1u\rangle}{\sqrt{2}}\right) = \left(\frac{|0u\rangle+e^{i2\pi 2^j\phi}|1u\rangle}{\sqrt{2}}\right) = \left(\frac{|0\rangle+e^{i2\pi 2^j\phi}|1\rangle}{\sqrt{2}}\right)\otimes|u\rangle \quad (3.23)$$

A questo punto, per capire il motivo per cui applicare la QFT inversa risolve il problema, ci si limiterà al caso in cui ϕ sia esprimibile esattamente come frazione binaria, ovvero $\phi = 0.\phi_{-1}\phi_{-2}\ldots\phi_{-t} = \sum_{k=-t}^{-1} 2^k\phi_k$. Sostituendo tale espressione nella 3.23:

$$\frac{|0\rangle+e^{i2\pi\sum_{k=-t}^{-1}2^{k+j}\phi_k}|1\rangle}{\sqrt{2}}\otimes|u\rangle$$

Ma data la periodicità dell'esponenziale complesso, tutti i termini della somma che possiedono una potenza di 2 a esponente non negativo sono nulli. Restano i dunque i termini che soddisfano la condizione $k < -j$:

$$\frac{|0\rangle+e^{i2\pi\sum_{k=-t}^{-j-1}2^{k+j}\phi_k}|1\rangle}{\sqrt{2}}\otimes|u\rangle$$

Ora si esplicita l'espressione dello stato totale del primo registro, scrivendo il prodotto tensoriale dei qubit da q_{t-1} a q_0:

$$\frac{|0\rangle + e^{i2\pi 2^{-1}\phi_{-t}}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{i2\pi(2^{-1}\phi_{-t+1}+2^{-2}\phi_{-t})}|1\rangle}{\sqrt{2}} \otimes \cdots$$
$$\cdots \otimes \frac{|0\rangle + e^{i2\pi(2^{-1}\phi_{-t}+2^{-2}\phi_{-t}+\cdots+2^{-t+1}\phi_{-t})}|1\rangle}{\sqrt{2}}$$

Che scritta con la notazione di frazione binaria diventa:

$$\frac{|0\rangle + e^{i2\pi 0.\phi_{-t}}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{i2\pi 0.\phi_{-t+1}\phi_{-t}}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{i2\pi 0.\phi_{-1}\ldots\phi_{-t+1}\phi_{-t}}|1\rangle}{\sqrt{2}}$$

Immagazzinando infine ϕ in un array di t qubit, in modo che $q_{t-1} = \phi_{-1}$, $q_{t-2} = \phi_{-2}$,... $q_0 = \phi_{-t}$, otteniamo esattamente la 3.22. Adesso è chiaro che operando una trasformata di Fourier inversa si ottiene uno stato che contiene la rappresentazione binaria della frazione ϕ cercata.

3.3.1 Realizzazione del circuito

Per testare tale algoritmo si sceglie l'operatore di rotazione di fase $\mathcal{U}1(\theta)$, già incontrato nel corso della QFT:

$$U1(\theta) \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

Ponendo $\theta = 2\pi\phi$ si ha che ϕ è proprio la fase dell'autovalore relativo all'autovettore $|1\rangle$. Limitandoci a $n = 3$ qubit, si può scrivere:

$$\phi = 2^{-1}c_2 + 2^{-2}c_1 + 2^{-3}c_0$$

e con questo ϕ il risultato che deve fornire il circuito è proprio $c_2c_1c_0$. Nel programma viene fissata una particolare scelta per le cifre della frazione binaria:

```
import math
from qiskit import QuantumProgram
import Qconfig

# Definizione della qft inversa
def qft_inverse(qc, qr, n):
  for i in range(n // 2):
    qc.swap(qr[i], qr[n-1-i])
  for i in range(n):
    for j in range(i):
      qc.cu1(math.pi/(2**(i-j)), qr[j], qr[i]).inverse()
    qc.h(qr[i]).inverse()
```

```
13
14 qp = QuantumProgram()
15 qp.set_api(Qconfig.APItoken, Qconfig.config["url"])
16
17 qr1 = qp.create_quantum_register("qr1", 3)
18 qr2 = qp.create_quantum_register("qr2", 1)
19 cr = qp.create_classical_register("cr", 3)
20 qc = qp.create_circuit("phi_est", [qr1,qr2], [cr])
21
22 # Si pone ad esempio c2=1, c1=1, c0=0
23 phi = 2**(-1)*1+2**(-2)*1+2**(-3)*0
24
25 # Preparo l'autostato
26 qc.x(qr2[0])
27
28 for j in range(3):
29   qc.h(qr1[j])
30   for k in range(2**(j)):
31     qc.cu1(2*math.pi*phi, qr1[j], qr2[0])
32
33 qft_inverse(qc, qr1, 3)
34
35 qc.measure(qr1, cr)
36
37 print(qp.get_qasm("phi_est"))
38
39 result = qp.execute(["phi_est"], backend="local_qasm_simulator", shots
     =1024)
40 print(result.get_data("phi_est"))
```

Output del codice:

```
OPENQASM 2.0;
include "qelib1.inc";
qreg qr1[3];
qreg qr2[1];
creg cr[3];
x qr2[0];
h qr1[0];
cu1(4.71238898038469) qr1[0],qr2[0];
h qr1[1];
cu1(4.71238898038469) qr1[1],qr2[0];
cu1(4.71238898038469) qr1[1],qr2[0];
h qr1[2];
cu1(4.71238898038469) qr1[2],qr2[0];
cu1(4.71238898038469) qr1[2],qr2[0];
cu1(4.71238898038469) qr1[2],qr2[0];
cu1(4.71238898038469) qr1[2],qr2[0];
swap qr1[0],qr1[2];
h qr1[0];
```

```
cu1(-1.57079632679490) qr1[0],qr1[1];
h qr1[1];
cu1(-0.785398163397448) qr1[0],qr1[2];
cu1(-1.57079632679490) qr1[1],qr1[2];
h qr1[2];
measure qr1[0] -> cr[0];
measure qr1[1] -> cr[1];
measure qr1[2] -> cr[2];

{'counts': {'110': 1024}}
```

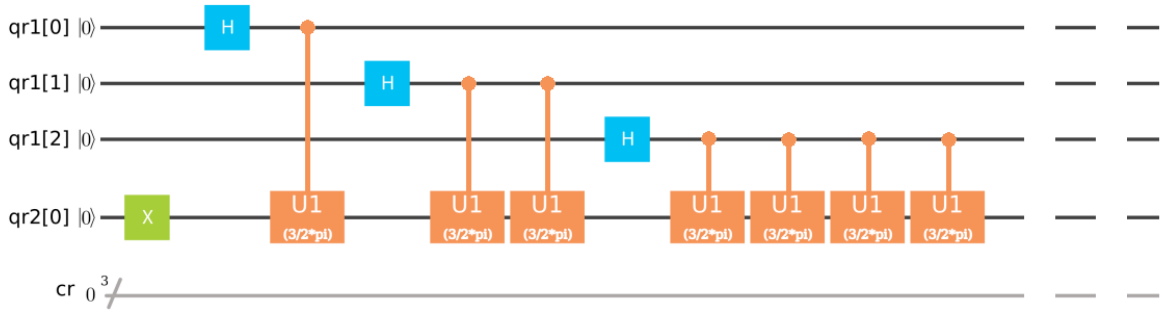


Figura 3.6: Prima parte del circuito di stima di fase (omettendo QFT inversa e misura)

Conclusioni

Gli algoritmi trattati in questa tesi (in particolare la trasformata di Fourier quantistica e la stima di fase) sono interessanti di per sé, ma la loro reale importanza deriva dal fatto che costituiscono il nucleo di altre procedure, ad esempio la fattorizzazione degli interi. Questo significa che se la computazione quantistica si dovesse sviluppare abbastanza da essere utilizzabile nel concreto, le attuali tecniche crittografiche andrebbero riconsiderate in quanto non sarebbero più ritenute sicure.
L'interesse per questa disciplina è notevole perché è un campo di ricerca aperto che promette considerevoli miglioramenti della complessità computazionale di algoritmi molto usati, ma, come si può notare confrontando i risultati attesi con quelli realmente ottenuti nelle esecuzioni dei programmi descritti in questa tesi, l'affidabilità non è ancora ottimale. I problemi principali da risolvere riguardano l'isolamento del sistema quantistico utilizzato come computer: come si è visto dai risultati degli esperimenti, le interazioni con l'esterno introducono rumore. Un'altra sfida importante è la realizzazione di computer quantistici che operino a temperatura ambiente, una condizione decisamente più pratica per l'uso comune rispetto alle temperature vicine allo zero assoluto a cui operano i dispositivi IBM utilizzati.
In ogni caso QISKit si è rivelato uno strumento efficace per toccare con mano lo stato di questo campo di ricerca (grazie alla possibilità di eseguire reali esperimenti online) ed anche un valido supporto didattico, in quanto il suo simulatore locale consente di testare la comprensione degli argomenti studiati tramite la stesura di semplici programmi. La documentazione e i programmi di tutorial forniti sono chiari ma non ancora nello stato ottimale: ciò è normale visto che tale strumento è in via di sviluppo; in più il fatto che sia ospitato sulla piattaforma *GitHub* è positivo poiché essa permette una rapida interazione con gli utenti, i quali possono contribuire e segnalare bug facilmente.

Bibliografia

[1] Michael A. Nielsen, Isaac L. Chuang, *Quantum Computation and Quantum Information*, Cambridge, 10th Anniversary Edition, 2010.

[2] Mikio Nakahara, Tetsuo Ohmi, *Quantum Computing: From Linear Algebra to Physical Realizations*, CRC Press, 2008.

[3] Sergueï Fedortchenko, *A quantum teleportation experiment for undergraduate students*, arXiv:1607.02398v2, 2016.

[4] J. Fessler, *The Discrete Fourier Transform*, `https://web.eecs.umich.edu/~fessler/course/451/l/pdf/c5.pdf`, 2004.