Quantum Computing

Alessandra Di Pierro

Appunti del corso - A.A. 2001-2002

Lezione 11-12-13: Algoritmi quantistici e complessitá

1 Complessitá computazionale quantistica

La ricerca nel campo della computazione quantistica ha prodotto risultati importanti anche dal punto di vista della complessitá computazionale. Per la computazione classica la teoria della complessitá fornisce una classificazione dei problemi computazionali in base alle risorse richieste per poter essre risolti su un computer classico. Due importantanti classi di complessitá sono note con i nomi di **P** e **NP**. Informalmente possiamo descrivere la prima come la classe di tutti i problemi che si possono risolvere *efficientemente*, cioé mediante un algoritmo deterministico in tempo polinomiale nella dimensione dei dati, mentre la seconda corrisponde alla classe dei problemi per i quali non é stato trovato nessun algoritmo in grado di risolverli efficientemente. Il problema di determinare se queste due classi coincidono é forse il piú importante dei problemi irrisolti in informatica teorica.

Un'altra importante classe di complessitá fa riferimento all risorsa "spazio" ed é nota con il nome di **PSPACE**. Questa classe comprende tutti quei problemi che si possono risolvere efficientemente dal punto di vista dello spazio utilizzato ma non necessariamente da punto di vista del tempo richiesto. I risultati sinora affermati in complessitá computazionale classica stabiliscono la seguente relazione tra queste classi:

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

Dopo l'avvento degli algoritmi randomizzati, é stata introdotta un'altra classe computazionale che comprende tutti i problemi per i quali esistono algoritmi probabilistici che li risolvono in tempo polinomiale e con probabilitá di errore piccola. Questa classe é nota come **BPP** (Bounded Polynomial

Probabilistic), e in un certo senso ha sostituito **P** come classe rappresentativa dei problemi effettivamente trattabili su un computer classico.

Tutte queste classi sono state definite rispetto al modello di computazione classica ovvero la Macchina di Turing e la sua versione probabilistica.

La teoria della complessitá computazionale quantistica fa riferimento al modello della Macchina di Turing Quantistica (cf. articolo de Bernstein-Vazirani) e ha stabilito finora risultati analoghi a quelli classici. In particolare, é stata definita la classe **BQP** che comprende tutti i problemi che possono essere risolti con probabilitá di errore piccola (bounded) usando un circuito quantistico polinomiale[1]. Il risultato piú importante finora stabilito in complessitá computazionale quantistica consiste nelle seguenti relazioni:

$$\mathbf{BPP} \subseteq \mathbf{BQP} \subseteq \mathbf{PSPACE}.$$

Non esistono al momento risultati che dimostrano che $\mathbf{BPP} \neq \mathbf{BQP}$ che sancirebbero la superioritá del modello quantistico rispetto a quello classico. La scoperta di alcuni algoritmi quantistici con prestazioni esponenzialmente migliori dei loro omologhi classici é un risultato promettente in questa direzione.

2 Algoritmi quantistici basati sulla Trasformata di Fourier quantistica

La scoperta finora piú importante in computazione quantistica é che fattorizzare un mumero di n bits su un computer quantistico richiede un numero di operazioni dell'ordine di $n^2 \log n \log\log n$, cioé un problema ritenuto intrattabile classicamente (il migliore algoritmo classico noto richiede tempo esponenziale) puó essere risolto efficientemente su un computer quantistico. Un ingrediente fondamentale dell'algoritmo quantistico per la fattorizzazione é la Trasformata di Fourier quantistica che ha applicazioni anche nella soluzione di altri problemi noti classicamente come problemi **NP**.

2.1 Trasformata di Fourier quantistica

La trasformata di Fourier é un metodo matematico per trasformare funzioni sul tempo in funzioni sulla frequenza[2]. Quindi questo metodo permette di

[1] Usiamo in questa definizione informale il modello della computazione quantistica basato sui circuiti che abbiamo studiato e che é equivalente a quello della Macchina di Turing Quantistica.

[2] La frequenza é l'inverso del periodo

trasformare una funzione di periodo r in una funzione che assume valori diversi da zero solo in corrispondenza dei multipli della frequenza $\frac{2\pi}{r}$. La *Trasformata di Fourier Discreta* (DFT) opera su N punti scelti a uguale distanza nell'intervallo $[0, 2\pi)$, per qualche intero N e restituisce una funzione il cui dominio sono i numeri tra 0 e $N-1$. La DFT di una funzione di periodo r é una funzione concentrata sui multipli di $\frac{N}{r}$: se r divide N allora la funzione risultante avrá valori diversi da zero solo sui multipli di $\frac{N}{r}$; altrimenti la funzione assumerá valori non nulli anche su interi vicini ai multipli di $\frac{N}{r}$.

Nella notazione matematica usuale, la DFT prende in input un vettore di numeri complessi $x_0, x_1, \ldots, x_{N-1}$ e restituisce il vettore trasformato $y_0, y_1, \ldots, y_{N-1}$ definito da

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N}.$$

La *Trasformata di Fourier Quantistica* (QFT) é una variante della versione della DFT dove N é una potenza di 2 (la Fast Fourier Transform o FFT). La QFT opera sulle ampiezze degli stati quantistici in modo simile alla DFT. In particolare, se $|0\rangle, \ldots, |N-1\rangle$ é una base ortonormale dello spazio degli stati, allora un generico stato $\sum_{j=0}^{N-1} x_j |j\rangle$ viene trasformato in $\sum_{k=0}^{N-1} y_k |k\rangle$, dove le ampiezze y_k sono le DFT delle ampiezze x_j.

La QFT é definita come un operatore lineare che trasforma gli stati della base nel seguente modo:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle.$$

Equivalentemente, la QFT si puó vedere come una matrice $N \times N$ il cui elemento generico (j,k) é $(e^{2\pi i/N})^{jk}$. Questo operatore é unitario. Per dimostrarlo costruiamo un circuito quantistico che lo implementa.

Assumiamo che $N = 2^n$ e che la base $|0\rangle, \ldots, |2^n - 1\rangle$ sia la base computazionale per gli stati su n qubits. Inoltre, é conveniente esprimere il generico stato $|j\rangle$ della base computazionale come il prodotto tensore $|j_1, \ldots, j_n\rangle$ dei bits della rappresentazione binaria di j, $j_1 2^{n-1} + j_2 2^{n-2} + \ldots, + j_n 2^0$. Possiamo quindi esprimere la QFT di $|j\rangle$ mediante il prodotto

$$\frac{(|0\rangle + e^{2\pi i 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1}j_n} |1\rangle) \ldots (|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n} |1\rangle)}{\sqrt{2^n}},$$

dove la notazione $0.j_l j_{l+1} \ldots j_m$ rappresenta la frazione binaria $j_l/2 + j_{l+1}/2^2 + \ldots + j_m/2^{m-l+1}$. Infatti, osservando che

$$\begin{aligned} \frac{k}{2^n} &= \frac{k_1 2^{n-1}}{2^n} + \ldots \frac{k_n 2^0}{2^n} \\ &= \frac{k_1}{2} + \ldots \frac{k_n}{2^n}, \end{aligned}$$

otteniamo

$$\begin{aligned} |j\rangle &\rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n |k\rangle} \\ &= \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \left(\sum_{l=1}^{n} k_l 2^{-l}\right)} |k_1 \ldots k_n\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \\ &= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right] \\ &= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right], \end{aligned}$$

Sviluppando la frazione binaria $j/2^{-l}$, l'ultima espressione corrisponde al prodotto che volevamo ottenere.

Da questa rappresentazione della QFT si puó derivare il circuito che la implementa (cf. Figura 1).

L'input é un registro di n qubits ognuno dei quali rappresenta una cifra binaria della rappresentazione di $j = j_1 \ldots j_n$. Applicando Hadamard al primo qubit $|j_1\rangle$ si ottiene

$$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i 0.j_1} |1\rangle \right) |j_2 \ldots j_n\rangle ,$$

in quanto se $j_1 = 1$, $e^{2\pi i 0.j_1} = e^{\pi i} = -1$, mentre se $j_1 = 0$, $e^{2\pi i 0.j_1} = 1$. Le altre operazioni impiegate nel circuito sono trasformazioni unitarie della forma

$$R_k = \left[\begin{array}{cc} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{array} \right],$$

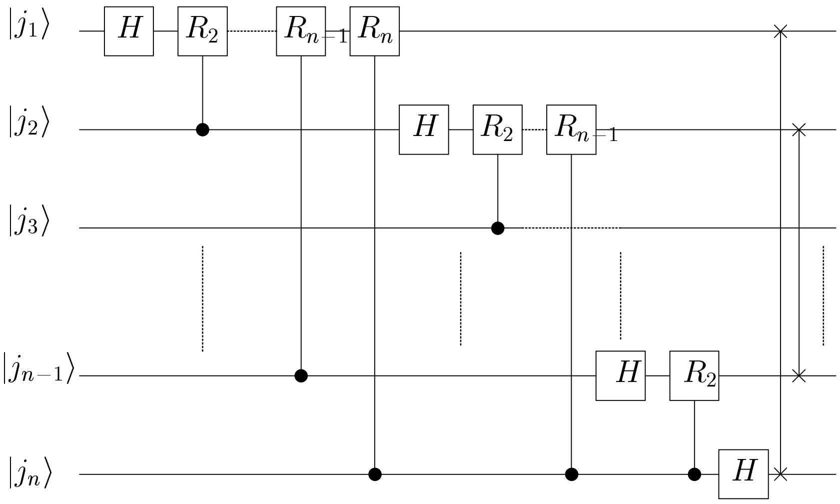

Figure 1: Circuito per la trasformata di Fourier quantistica

che usate come operazioni controllate permettono di aggiungere il bit di indice k alla fase del coefficiente di $|1\rangle$. Dopo l'applicazione dei controlled-R_i, $i = 2, \ldots, n$ al primo qubit si ottiene quindi

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 \ldots j_n}|1\rangle\right)|j_2 \ldots j_n\rangle .$$

A questo punto si esegue un procedimento analogo al secondo qubit ottenendo

$$\frac{1}{\sqrt{2^2}}\left(|0\rangle + e^{2\pi i 0.j_1 \ldots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}|1\rangle\right)|j_3 \ldots j_n\rangle ,$$

e si continua in questo modo per ogni qubit, fino ad ottenere lo stato

$$\frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i 0.j_1 \ldots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \ldots j_n}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 0.j_0}|1\rangle\right) .$$

Per ritrovare la QFT di $|j\rangle$ basta ora applicare $n/2$ operazioni di scambio per riportare i fattori nell'ordine giusto. Ricordiamo che é possibile scambiare due qubits utilizzando tre `CNOT`. Poiché tutte le operazioni utilizzate sono unitarie, possiamo concludere che QFT é una trasformazione unitaria. In totale il numero delle operazioni richieste per calcolare la QFT é dell'ordine di n^2 (ci vogliono $n(n+1)/2$ Hadamard e controlled-R_k). Il calcolo della trasformata di Fourier su un computer classico usando FFT

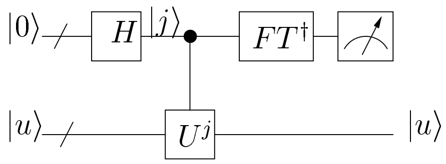

Figure 2: Circuito schematico per l'algoritmo di stima delle fasi

richiede invece un numero di operazioni dell'ordine di $n2^n$, cioé esponenzialmente piú elevato della versione quantistica. Notiamo tuttavia che il risultato dell'applicazione della QFT ad un generico stato ottenuto come sovrapposizione degli stati $|j\rangle$ della base computazionale risiede nelle ampiezze della sovrapposizione quantistica e non é quindi direttamente accessibile mediante misurazione. Gli usi della QFT in computazione quantistica sono tutti di tipo indiretto, ma fondamentali per la costruzione di tutti quegli algoritmi quantistici che esibiscono un comportamento esponenzialmente piú veloce rispetto alle loro controparti classiche.

Esercizio 2.1 *Verificare che l'inversa della trasformata di Fourier quantistica é*

$$QFT^\dagger : |j\rangle \to \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^{n-1}} e^{-2\pi ijk/2^n} |k\rangle .$$

2.2 Stima dell'autovalore di una matrice unitaria

Dato un operatore unitario U su m qubits con un autovalore λ corrispondente ad un autovettore $|u\rangle$, il problema é stabilire con una certa precisione il valore di λ disponendo di oracoli che eseguono controlled-U^{2^j}, $j = 0, \ldots, t$ per un dato intero positivo t, e di una preparazione dello stato $|u\rangle$. Nota che poiché U é unitaria, $|\lambda| = 1$ cioé $\lambda = e^{2\pi i\varphi}$ per qualche φ tale che $0 \leq \varphi \leq 1$. Il problema consiste quindi nel trovare una stima per φ.

Questo problema si puó risolvere efficientemente mediante l'algoritmo quantistico che descriviamo nel seguito. L'uso di questo algoritmo non é limitato a risolvere il problema descritto, ma é parte integrante di altri algoritmi, dal momento che molti problemi si possono ridurre al problema della stima delle fasi, tra cui quelli descritti in 2.3 e 2.4.

Il circuito schematizzato in Figura 2 implementa la procedura che si svolge in tre passi a partire da un input formato da due registri.

Il primo registro contiene t qubits inizialmente nello stato $|0\rangle$. Il secondo registro contiene l'autovettore $|u\rangle$ che non verrá modificato dall'applicazione

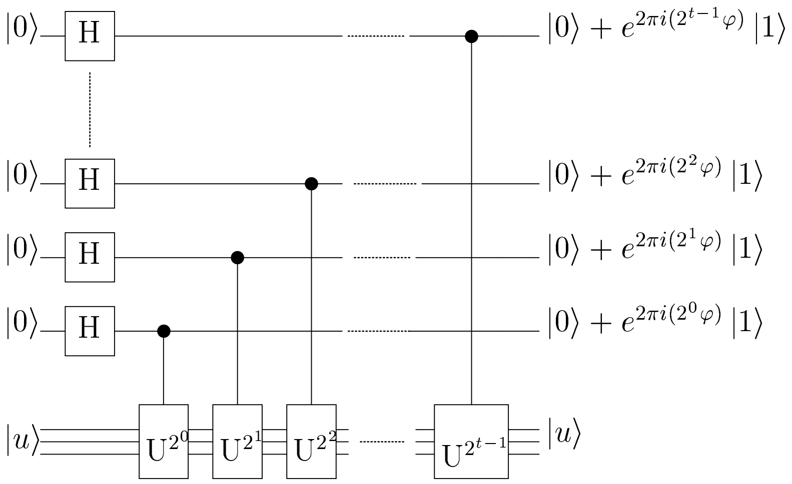

Figure 3: Descrizione dettagliata del primo stadio del circuito in Figura 2

di U^j. Nella fase iniziale si applica $H^{\otimes t}$ al primo registro $|0\rangle^{\otimes t}$ che diventa quindi

$$|j\rangle = \left[(|0\rangle + |1\rangle)/\sqrt{2}\right]^{\otimes t} = \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle \,.$$

Ciascun qubit nel primo registro viene usato come qubit di controllo dell'operazione controlled-U^{2^k} come illustrato in Figura 3.

L'effetto della sequenza di controlled-U^{2^k} é di trasformare $|j\rangle\,|u\rangle$ in $|j\rangle\,U^j\,|u\rangle$. Infatti, notando che la rappresentazione binaria di j é $j_1 2^{t-1} + j_2 2^{t-2} + \ldots j_t 2^0$,

$$U^j = U^{j_t 2^0} U^{j_{t-1} 2^1} \cdots U^{j_2 2^{t-2}} U^{j_1 2^{t-1}},$$

che viene applicata solo quando $j_k = 1$ per ogni $k = 1, \ldots, t$, cioé solo se $j = 1$.

Osserviamo che se $|u\rangle$ é una autovettore di U con autovalore λ, allora $|u\rangle$ é anche un autovettore di U^{2^k} con autovalore λ^{2^k}. Pertanto

$$\begin{aligned}
|j\rangle\,U^j\,|u\rangle &= |j\rangle\, e^{2\pi i j\varphi}\,|u\rangle \\
&= e^{2\pi i j\varphi}\,|j\rangle\,|u\rangle \\
&= \left[(e^{2\pi i j_1 2^{t-1}\varphi}\,|j_1\rangle) \cdots (e^{2\pi i j_{t-1} 2^1\varphi}\,|j_{t-1}\rangle)(e^{2\pi i j_t 2^0\varphi}\,|j_t\rangle)\right] |u\rangle \,.
\end{aligned}$$

Poiché ogni j_k é della forma $(|0\rangle + |1\rangle)/\sqrt{2}$, lo stato del primo registro dopo l'applicazione del circuito in Figura 3 risulta

$$\bigotimes_{k=0}^{t-1} \frac{|0\rangle + e^{2\pi i 2^k \varphi} |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2^t}} \sum_{y=0}^{2^t-1} e^{2\pi i \varphi y/2^t} |y\rangle \,.$$

Ripetendo il ragionamento algebrico fatto per ottenere l'espressione prodotto della QFT, si vede che l'ultima espressione é esattamente

$$\frac{1}{\sqrt{2^t}}(|0\rangle + e^{2\pi i 0.\varphi_t} |1\rangle)(|0\rangle + e^{2\pi i 0.\varphi_{t-1}\varphi_t} |1\rangle) \cdots (|0\rangle + e^{2\pi i 0.\varphi_1\varphi_2\ldots\varphi_t} |1\rangle).$$

Il secondo passo dell'algoritmo consiste nell'applicazione dell'inversa della trasformata di Fourier, $QFT^\dagger$. Supponiamo che φ si possa rappresentare esattamente con t bits. Allora il secondo passo produce $|\varphi_1\varphi_2 \ldots \varphi_t\rangle$. L'ultimo passo é una misurazione nella base computazionale che permette quindi di ottenere il valore esatto di φ.

Supponiamo ora che φ sia un qualsiasi numero reale in $[0, 1]$ la cui rappresentazione su t bit é approssimata a meno di un errore δ. In questo caso si dimostra che dopo l'applicazione della $QFT^\dagger$, una misurazione dello stato finale produce questa approssimazione su t bits di φ con probabilitá almeno $4/\pi^2 \approx 0.405$.

Il valore di t si puó fissare in modo da ottenere una stima di φ precisa quanto si vuole. Si puó cioé fissare un ϵ tale che l'algoritmo produca un'approssimazione di φ precisa fino all'n-simo bit con probabilitá almeno $1 - \epsilon$. Per ottenere questa stima si deve fissare t al valore

$$t = n + \log\left(2 + \frac{1}{2\epsilon}\right), \tag{1}$$

dove si prende la parte intera alta del logaritmo.

In generale, dati gli autovettori $|u\rangle\,, u \in T$ di un operatore unitario U con corrispondenti autovalori ϕ_u, l'algoritmo di stima della fasi trasforma lo stato normalizzato

$$|0\rangle\,(\sum_{u\in T} d_u\,|u\rangle)$$

nello stato

$$\sum_{u\in T} d_u \left|\tilde{\phi_u}\right\rangle |u\rangle),$$

dove $\left|\tilde{\phi_u}\right\rangle$ é una buona stima di $|\phi_u\rangle$. In questo caso la probabilitá che questa stima sia precisa fino all'n-simo bit é almeno $|d_u|^2(1-\epsilon)$ se t é scelto come in (1).

2.3 Trovare l'ordine di un numero

Dati due interi positivi x ed N che non hanno fattori comuni e tali che $x < N$, si definisce l'*ordine* di x modulo N come il piú piccolo intero r tale che $x^r = 1(modN)$.

Non si conoscono algoritmi classici che dati x ed N permettono di determinare r in tempo polinomiale in $L = \log N$. Nel seguito decsriviamo un algoritmo quantistico che risolve questo problema con un numero di operazioni dell'ordine di L^3.

L'algoritmo consiste essenzialmente nell'applicare il metodo di stima delle fasi all'operatore unitario U definito da:

$$U|y\rangle = |xy(modN)\rangle \qquad \text{se } 0 \leq y \leq N-1$$
$$U|y\rangle = |y\rangle \qquad \text{se } N \leq y \leq 2^L - 1.$$

Si verifica facilmente che ogni vettore della forma

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} \left| x^k(modN) \right\rangle,$$

per $0 \leq s \leq r-1$, é un autovettore di U con autovalore $e^{2\pi is/r}$. Infatti, usando l'ipotesi $x^r = 1(modN)$, si ha che:

$$\begin{aligned}
U|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} \left| x^{k+1}(modN) \right\rangle \\
&= \frac{1}{\sqrt{r}} \sum_{k=1}^{r} e^{-2\pi is(k-1)/r} \left| x^{k}(modN) \right\rangle \\
&= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi is(k-1)/r} \left| x^{k}(modN) \right\rangle \\
&= e^{-2\pi is/r} |u_s\rangle.
\end{aligned}$$

L'idea base dell'algoritmo é di ottenere un'approssimazione accurata della fase s/r da cui poter ricavare il valore di r.

Il primo problema da risolvere é preparare lo stato $|u_s\rangle$. Questo problema noné banale dal momento che $|u_s\rangle$ é definito in termini del valore r che vogliamo trovare. Invece di $|u_s\rangle$, prepariamo il secondo registro in input al circuito per la stima delle fasi nello stato $|1\rangle$. Pochi calcoli dimostrano infatti che questo stato corrisponde ad una combinazione lineare di tutti gli autovettori $|u_s\rangle$ di U per $0 \leq s \leq r-1$:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = \sum_{k=0}^{r-1} \left(\frac{1}{r} \sum_{s=0}^{r-1} e^{-2\pi isk/r} \right) \left| x^k (modN) \right\rangle ,$$

dove l'espressione in parentesi risulta uguale a 1 se $k=0$, mentre é 0 per tutti gli altri valori di k. Per vedere questo basta osservare che per $k>0$ la sommatoria in parentesi é la serie geometrica di ragione $e^{-2\pi ik/r}$ e quindi converge a $\frac{1-(e^{-2\pi ik/r})^r}{1-e^{-2\pi ik/r}} = 0$. Pertanto, la combinazione degli autovettori $|u_s\rangle$ coincide con lo stato $|1\rangle$.

Per ragioni che saranno chiarite in 2.3.2, prendiamo $n = 2L+1$. Di conseguenza, il numero di qubits del primo registro dell'algoritmo di stima delle fasi sará $t = 2L + 1 + log2 + 1/2\epsilon$. Inoltre una misurazione del primo registro produrrá per ogni s, $0 \leq s \leq r-1$, un'approssimazione φ_s di s/r precisa fino al bit $2L+1$ con una probabilitá di almeno $(1-\epsilon)/r$ (cf. 2.2).

Dobbiamo ora risolvere ancora due problemi:

Come eseguire la sequenza di controlled-U^{2^k},
Come ricavare r dalla stima φ_s trovata.

2.3.1 Complessitá delle operazioni controlled-U^{2^k}

Nell'algoritmo descritto in 2.2 le operazioni controllate U^{2^k} sono state trattate come "scatole nere". Nell'istanza particolare dell'algoritmo per il nostro problema di trovare l'ordine di un numero x modulo N, la sequenza di controlled-U^{2^k} corrisponde esattamente alla moltiplicazione modulo N del secondo registro per x elevato ad una potenza pari al contenuto del primo registro. Infatti, se $|k\rangle$ é lo stato del primo registro e $|u\rangle$ lo stato del secondo registro, allora dopo l'applicazione della sequenza di controlled-U^{2^k} si ottiene

$$\begin{aligned}
|k\rangle |u\rangle &\mapsto |k\rangle U^{k_t 2^{t-1}} U^{k_{t-1} 2^{t-2}} \ldots U^{k_1 2^0} |u\rangle \\
&= |k\rangle \left| x^{k_t 2^{t-1}} x^{k_{t-1} 2^{t-2}} \ldots x^{k_1 2^0} u(modN) \right\rangle \\
&= |k\rangle \left| x^{k_t 2^{t-1} + k_{t-1} 2^{t-2} \ldots k_1 2^0} u(modN) \right\rangle \\
&= |k\rangle \left| x^k u(modN) \right\rangle .
\end{aligned}$$

Questa operazione puó essere realizzata usando lo schema di computazione reversibile con un numero di operazioni elementari dell'ordine di L^3. L'idea é di eseguire $t-1=O(L)$ operazioni di moltiplicazione modulare per calcolare $x^2(modN)$, $x^4(modN)$ e cosí via fino a $x^{2^{t-1}}(modN)$. Ciascuna di queste operazioni costa $O(L^2)$ (se si usano le regole ordinarie per la moltiplicazione binaria). Quindi il calcolo costa in totale $O(L^3)$. A questo punto si usa l'identitá

$$x^k u(modN) = x^{k_t 2^{t-1} + k_{t-1} 2^{t-2} \ldots + k_1 2^0} u(modN),$$

per ottenere $x^k u(modN)$ con altre $t-1$ moltiplicazioni reversibili con complessitá $O(L^3)$.

2.3.2 Frazioni continue

Per ricavare r dalla stima φ_s si usa la tecnica delle frazioni continue che illustriamo brevemente nel seguito.

Le frazioni continue permettono di approssimare un qualsiasi numero reale con una sequenza di numeri razionali della forma

$$[a_0, a_1, a_2, \ldots, a_p] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ldots + \cfrac{1}{a_p}}}},$$

dove a_j sono interi positivi per $j \geq 1$.

Per un numero reale c, si definisce l'espansione in frazioni continue nel seguente modo. A partire da c si costruiscono ricorsivamente le sequenze $\{a_j\}_{j\geq 0}$ e $\{r_j\}_{j\geq 0}$ prendendo a_0 come la parte intera bassa di c, $r_0 = c - a_0$ e per ogni $j \geq 1$, $a_j =$ parte intera bassa di $\frac{1}{r_{j-1}}$ e $r_j = \frac{1}{r_{j-1}} -$ parte intera bassa di $\frac{1}{r_{j-1}}$. Allora per ogni $j \geq 0$ con $r_j > 0$,

$$c = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ldots + \cfrac{1}{a_j + r_j}}}}.$$

Il numero razionale $[a_0, a_1, a_2, \ldots, a_j]$ é detto il j-esimo convergente di c. Se $r_j = 0$ allora le frazioni continue terminano con a_j e si ottiene l'uguaglianza $c = [a_0, a_1, a_2, \ldots, a_j]$.

Esempio 2.2 *L'espansione in frazioni continue di* $47/13$ *é data da:*

$$\frac{47}{13} = 3 + \frac{8}{13} = 3 + \frac{1}{\frac{13}{8}}$$

$$
\begin{aligned}
&= 3 + \cfrac{1}{1 + \frac{5}{8}} = 3 + \cfrac{1}{1 + \frac{1}{\frac{8}{5}}} \\
&= 3 + \cfrac{1}{1 + \frac{1}{1+\frac{3}{5}}} = 3 + \cfrac{1}{1 + \frac{1}{1+\frac{1}{\frac{5}{3}}}} \\
&= 3 + \cfrac{1}{1 + \frac{1}{1+\frac{1}{1+\frac{2}{3}}}} = 3 + \cfrac{1}{1 + \frac{1}{1+\frac{1}{1+\frac{1}{\frac{3}{2}}}}} \\
&= 3 + \cfrac{1}{1 + \frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{2}}}}}.
\end{aligned}
$$

Pertanto $c = [3, 1, 1, 1, 1, 2]$.

Elenchiamo alcuni importanti risultati della teoria dei numeri che ci saranno utili per il nostro problema. In particolare, siamo interessati al caso in cui c é un numero razionale (come appunto s/r). In questo caso si dimostra la seguente proposizione:

Proposizione 2.3 *L'espansione in frazioni continue di un numero reale c termina se e solo se c é un numero razionale.*

Proposizione 2.4 *Il numero razionale $[a_0, a_1, a_2, \ldots, a_j]$ é il numero $\frac{p_j}{q_j}$ dove $p_0 = a_0$, $p_1 = 1 + a_0 a_1$, $q_1 = a_1$ e per ogni $j \geq 2$:*

$$p_j = 1 + a_j p_{j-1} + p_{j-2} \; q_j = a_j q_{j-1} + q_{j-2}.$$

Corollario 2.5 *L'espansione in frazioni continue di un numero razionale positivo p/q si puó ottenere in $O(m^3)$ operazioni, se p e q sono interi su m bits.*

Teorema 2.6 *Supponiamo che x e p/q siano numeri razionali tali che $|x - p/q| \leq 1/2q^2$. Allora p/q é un convergente della frazione continuata per x.*

Questo teorema ci permette di concludere che s/r é un convergente del numero razionale φ_s ottenuto dall'algoritmo di stima delle fasi, purché si scelga una precisione $n = 2L+1$. Infatti, in questo caso, poiché $r \leq N \leq 2^L$, si ha che

$$|\varphi_s - s/r| < 1/2^{2L+1} \leq 1/2r^2.$$

Inoltre, questo convergente s/r si puó calcolare con $O(L^3)$ operazioni elementari.

Avendo a disposizione questi risultati possiamo ora completare la descrizione dell'algoritmo quantistico per trovare l'ordine r di un numero positivo x modulo N.

Calcoliamo con il metodo delle frazioni continue un convergente p/q di φ_s, tale che $q \leq 2^L$ e $|\varphi_s - p/q| < 1/2^{2L+1}$ (si dimostra che esiste sempre almeno una frazione con queste proprietá). Il denominatore q é un candidato per r. Si controlla se $x^q = 1(modN)$. Se il test ha successo, allora q é l'ordine di x modulo N. Altrimenti l'algoritmo fallisce e bisogna eseguirlo nuovamente.

Una condizione importante per il successo dell'algoritmo é che r ed s non abbiano fattori comuni, altrimenti il numero restituito dall'algoritmo delle frazioni continue potrebbe essere un fattore di r piuttosto che r stesso. Per N molto grandi, la probabilitá che r ed s siano co-primi é almeno $1/\log N$. Basta quindi ripetere l'algoritmo un numero adeguato di volte per ottenere con alta probabilitá un istanza di s/r con r ed s siano co-primi. In particolare, ripetendo l'algoritmo $O(L)$ volte l'algoritmo ha successo con probabilitá $(1-\epsilon)(1-1/N)$, con un costo totale di $O(L^4)$ operazioni.

2.4 Fattorizzazione

Il problema della fattorizzazione é il seguente: Dato un numero intero positivo dispari e composto N, trovare i suoi fattori primi.

Determinare se un numero N é primo o composto é un problema computazionalmente "facile": l'algoritmo (probabilistico) di Miller-Rabin per il test di primalitá impiega $O(s \log N)$ operazioni aritmetiche con una probabilitá di errore $P \leq 2^{-s}$. Una volta stabilito che un numero é composto, non sembra altrettanto facile determinare i suoi fattori primi.

I sistemi crittografici oggi piú diffusi (come il sistema RSA[3]) sono basati sulla seguente congettura:

Congettura 2.7 *La fattorizzazione di interi é un problema computazionalmente piú difficile della moltiplicazione di interi: mentre esistono molti algoritmi polinomiali per la moltiplicazione di interi, non esistono algoritmi polinomiali per la fattorizzazione di interi.*

Questa assunzione é basata sul fatto che gli sforzi che per molte centinaia di anni hanno impegnato i piú grandi scenziati non sono stati sufficienti a produrre un algoritmo polinomiale per la fattorizzazione. Il piú

[3]RSA é un sistema crittografico a chiavi pubbliche inventato da Rivest, Shamir, Adleman da cui ha preso il nome.

efficiente algoritmo classico che si conosca al momento é il cosiddetto "number field sieve" la cui complessitá é stata euristicamente dimostrata essere $O(exp\left[c(\log N)^{1/3}(loglogN)^{2/3}\right])$, cioé l'algoritmo richiede un tempo super-polinomiale nel numero di cifre $O(\log N)$ in N.

Verso la metá degli anni '90 Peter Shor ha costruito (partendo da risultati precedenti dovuti a Benioff, Bennet, Deutsch, Feynmann, Simon ed altri) un algoritmo quantistico che puó fattorizzare un intero in tempo polinomiale. La complessitá di questo algoritmo risulta infatti $O\left((\log N)^2(\log\log N)(\log\log\log N)\right)$, cioé polinomiale nel numero di cifre $O(\log N)$ in N.

L'algoritmo consiste dei cinque passi descritti in 2.4.1, di cui solo il **Passo 3** richiede l'uso di un computer quantistico (in particolare questo passo invoca la procedura quantistica per trovare l'ordine di un numero descritta in 2.3). Tutti gli altri passi dell'algoritmo possono essere eseguiti su un computer classico.

Richiamiamo brevemente i principali risultati di teoria dei numeri necessari per capire come funziona l'algoritmo e qual é la sua complessitá.

Descriviamo per prima cosa l'algoritmo di Euclide per trovare il massimo comun divisore, $MCD(p,q)$, di due interi positivi p e q che si basa sul seguente risultato:

Proposizione 2.8 *Dati due interi p e q, supponiamo che r sia il resto della divisione di p per q e che sia $r \neq 0$. Allora*

$$MCD(p,q) = MCD(q,r).$$

L'algoritmo di Euclide é il seguente: supponendo che $p > q$, si divide p per q e si trova il resto $r_1 < q$. Per la proposizione 2.8, $MCD(p,q) = MCD(q,r_1)$. Si divide quindi q per r_1 e si trova il resto $r_2 < r_1$ con $MCD(q,r_1) = MCD(r_1,r_2)$. Induttivamente, si divide r_{n-2} per r_{n-1} per trovare il resto $r_n < r_{n-1}$ con

$$MCD(p,q) = MCD(q,r_1) = \ldots = MCD(r_n,r_{n+1}).$$

Poiché la sequenza r_j é una sequenza di interi positivi strettamente decrescente, esiste n tale che $r_{n+1} = 0$, cioé r_{n-1} é un multiplo di r_n. Quindi l'algoritmo termina con

$$MCD(p,q) = MCD(r_{n-1},r_n) = r_n.$$

La complessitá dell'algoritmo di Euclide é $O(L^3)$, dove L é il numero di bits necessari per rappresentare p e q. Infatti, l'algoritmi richiede al piú $O(L)$ divisioni binarie, ciascuna delle quali richiede $O(L^2)$ operazioni su bits.

Il seguente teorema ci fa vedere come il problema della fattorizzazione si riduce a quello di trovare l'ordine di un numero.

Teorema 2.9 *Supponiamo che N sia un numero composto e $x \in \{1, \ldots, N\}$ una soluzione non banale dell'equazione $x^2 = 1(modN)$, cioé tale che $x \neq 1(modN)$ e $x \neq N - 1 = -1(modN)$. Supponiamo inoltre che L sia il numero di bits necessari per rappresentare N (L é la parte intera alta di $\log N$). Allora almeno uno tra $MCD(x-1, N)$ e $MCD(x+1, N)$ é un fattore non banale di N che puó essere calcolato in $O(L^3)$) passi.*

Questo risultato combinato con il seguente teorema permette di determinare con alta probabilitá un fattore non banale di un qualsiasi numero composto N.

Teorema 2.10 *Supponiamo che la scomposizione in fattori primi del numero intero dispari composto N sia $N = p_1^{n_1} \cdots p_m^{n_m}$. Se x é un intero scelto casualmente tra 1 e N tale che $MCD(x, N) = 1$ e r é l'ordine di x modulo N, allora*

$$P(r \text{ é pari e } x^{r/2} \neq \pm 1(modN)) \geq 1 - 1/2^m.$$

2.4.1 L'algoritmo

Per calcolare un fattore non banale di un numero intero N di L bits, dispari e composto si procede come segue:

Passo 1 Scegliamo un numero casuale x tra 1 e $N - 1$.

Passo 2 Con l'algoritmo di Euclide calcoliamo il $MCD(x, N)$. Se $MCD(x, N) > 1$ allora abbiamo trovato un fattore non banale di N. Altrimenti si procede con il passo 3.

Passo 3 Usiamo l'algoritmo quantistico descritto in 2.3 per trovare l'ordine r di x modulo N.

Passo 4 Se r é dispari, oppure r é pari e $x^{r/2} = -1(modN)$, allora ritorniamo al passo 1.

Passo 5 Con l'algoritmo di Euclide calcoliamo $MCD(x^{r/2}-1, N)$ e $MCD(x^{r/2}+1, N)$ e se uno dei due interi calcolati risulta essere un fattore non banale di N, allora l'algoritmo termina con successo. Altrimenti fallisce e bisogna ripeterlo a partire dal passo 1.