Università di Bologna

L'Algoritmo di Deutsch-Jozsa in QScript

Relatore

Dal Lago Ugo

Candidato

Peconi Federico

20 Dicembre 2017

Overview

- Funzioni Booleane Bilanciate
- Il Calcolo Quantistico
- L'algoritmo di Deutsch-Jozsa
  - Definizione del problema
  - Analisi della correttezza
  - Complessità relativa
- Implementazione in Quantum Playground
  - Scelta delle funzioni bilanciate
  - Costruzione dell'oracolo
  - Utilizzo di QScript

Il Problema

Sia $f : \{0,1\}^n \rightarrow \{0,1\}$ una funzione Booleana, diremo che:

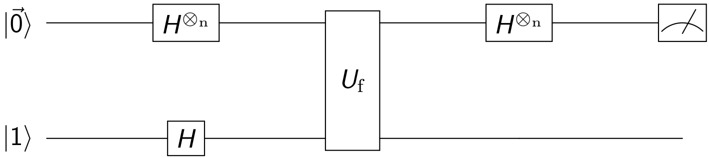
- f **bilanciata** se metà degli input viene mappata in 0 e l'altra metà in 1
- f **costante** se tutti gli input sono mappati in 0 oppure tutti gli input sono mappati in 1

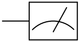
Problema: *sia data f Booleana di cui non conosciamo la definizione interna, dire se f è costante oppure bilanciata. Sapendo che sarà sicuramente o bilanciata o costante.*

Risoluzione Classica: classicamente, nel caso pessimo in cui f è costante, il problema è risolvibile valutando f su $2^{n-1} + 1$ input diversi così da poter escludere che sia bilanciata. La complessità del problema è $\Omega(2^{n-1})$ e quindi esponenziale

L'Algoritmo di Deutsch-Jozsa

L'Algoritmo di Deutsch-Jozsa fa uso di una serie di trasformazioni per giungere alla soluzione rappresentabili dal seguente circuito:



 indica dove viene effettuata la misurazione finale.

- Per garantire la reversibilità, al calcolo di f viene associato un qubit ausiliario 1 tale che $U_f(|\vec{0}, 1\rangle) = |\vec{0}, f(\vec{0}) \oplus 1\rangle$

Teorema

L'algoritmo di Deutsch-Jozsa risolve il problema valutando U_f una sola volta

Dimostrazione

- Applichiamo Hadamard su tutti i qubit del sistema e otteniamo lo stato

$$\frac{1}{2^n} \sum_{\vec{x} \in \{0,1\}^n} |\vec{x}\rangle (|0\rangle - |1\rangle)$$

- Applichiamo U_f e raccogliamo concentrandoci solo sui primi n qubit

$$\frac{1}{\sqrt{2^n}} \sum_{\vec{x} \in \{0,1\}^n} (-1)^{f(\vec{x})} |\vec{x}\rangle$$

- Riapplichiamo $H^{\otimes n}$, f verrà valutata per ogni possibile sovrapposizione

$$\frac{1}{2^n} \sum_{\vec{z} \in \{0,1\}^n} \sum_{\vec{x} \in \{0,1\}^n} (-1)^{f(\vec{x})} (-1)^{\vec{x} \cdot \vec{z}} |\vec{z}\rangle$$

- Sfruttando gli effetti dovuti all'interferenza degli stati sovrapposti, se andiamo a misurare la probabilità di misurare lo stato $|\vec{z}\rangle = |\vec{0}\rangle$ otteniamo

$$\frac{1}{2^n}\sum_{\vec{x}\in\{0,1\}^n}(-1)^{f(\vec{x})}|\vec{0}\rangle = \begin{cases} \pm\frac{2^n}{2^n}|\vec{0}\rangle = \pm 1|\vec{0}\rangle & f \textit{ costante} \\ \frac{2^{n-1}}{2^n}|\vec{0}\rangle - \frac{2^{n-1}}{2^n}|\vec{0}\rangle = 0|\vec{0}\rangle & f \textit{ bilanciata} \end{cases}$$

$\square$

Verrà quindi misurato il valore $|\vec{0}\rangle$ se e solo se f è costante, altrimenti f sarà bilanciata

- Deutsch-Jozsa è ottimo e risulta esponenzialmente più veloce di ogni altro algoritmo classico

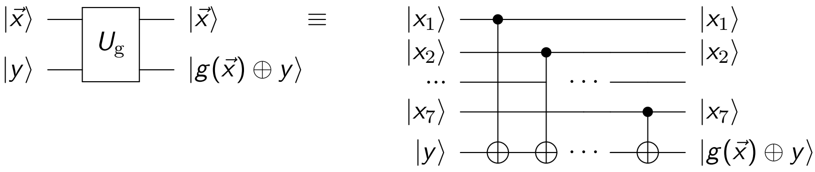
Quantum Playground

- Per l'implementazione dell'algoritmo è stata usata l'applicazione web QuantumPlayground che simula un registro quantistico fino a 22 qubit programmabile attraverso un D.S.L. chiamato QScript
- A supporto della programmazione, QuantumPlayground fornisce un sistema di debugging e un motore grafico che permette di visualizzare l'evoluzione dello stato del sistema durante l'esecuzione
- L'obiettivo è quello di tradurre il circuito di Deutsch-Jozsa in uno script; è necessario scegliere una funzione f su cui costruire U_f

Come primo esempio è stata scelta la funzione Booleana lineare $g(x_1, x_2, x_3, \cdots, x_7) = x_1 \oplus x_2 \oplus x_3 \oplus \cdots \oplus x_7$, che è anche bilanciata

- L'operatore "xor" è attuabile attraverso la trasformazione unitaria Controlled-Not
- U_g sarà quindi : $U_g(|\vec{x}, y\rangle) = |\vec{x}, g(\vec{x}) \oplus y\rangle = |\vec{x}, x_1 \oplus \cdots \oplus x_7 \oplus y\rangle$

Il circuito risultante per costruire U_g è allora



$$(\cdots(((x_1 \oplus y) \oplus x_2) \oplus x_3) \oplus \cdots \oplus x_7) = x_1 \oplus x_2 \oplus \cdots \oplus x_7 \oplus y = g(\vec{x}) \oplus y$$

Risultato esecuzione g

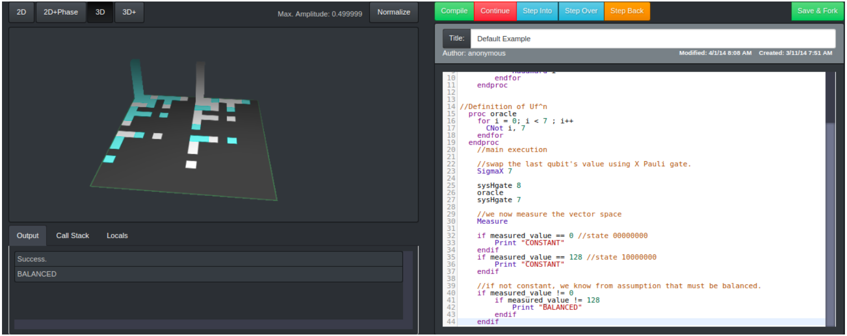


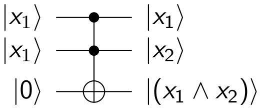
Figura: QuantumPlayground, Deutsch–Jozsa su g

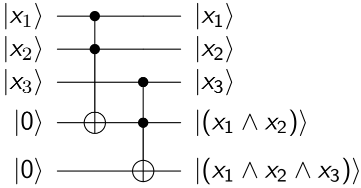
Come secondo esempio è stata scelta la funzione Booleana non-lineare

$$h(x_1, x_2, x_3, x_4, x_5) = (x_1 \wedge x_2) \oplus x_3 \oplus (x_2 \wedge x_3 \wedge x_4) \oplus (x_2 \wedge x_3 \wedge x_5) \oplus (x_3 \wedge x_4) \oplus (x_4 \wedge x_5)$$

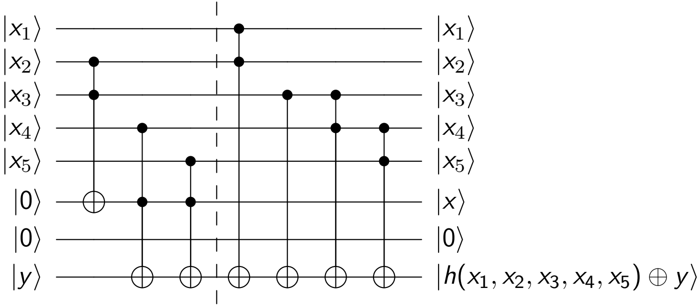
che è anche bilanciata

- L'operatore "and" è attuabile attraverso la trasformazione Toffoli, che viene usata 2 volte aggiungendo un qubit ausiliario per "and" a 3 stati





U_h risulta allora nella seguente composizione di CNot e Toffoli



$$y \oplus (x_2 \wedge x_3 \wedge x_4) \oplus (x_2 \wedge x_3 \wedge x_5) \oplus (x_1 \wedge x_2) \oplus x_3 \oplus (x_3 \wedge x_4) \oplus (x_4 \wedge x_5)$$

Risultato esecuzione h

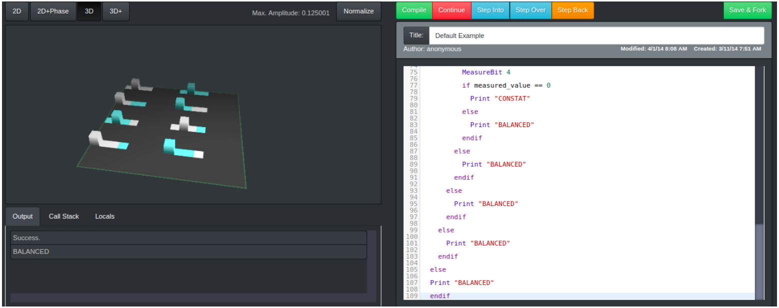


Figura: QuantumPlayground, Deutsch–Jozsa su h

Conclusioni, Future Work

- È stato quindi messo in luce come il calcolo quantistico permetta, per determinati problemi, lo sviluppo di algoritmi che sono più efficienti di ogni possibile procedura classica. Altri esempi famosi sono:
  - Shor, Groover, ...
- L'utilizzo di QScript, oltre a favorire la comprensione dell'algoritmo, permette di esercitarsi con la programmazione quantistica a basso livello
  - Nonostante l'industria sia ancora agli albori, la programmazione di computer quantistici general purpose è già realtà