THỰC THI MỘT SỐ THUẬT TOÁN LƯỢNG TỬ CƠ BẢN

IMPLEMENTATING SOME QUANTUM BASIC ALGORITHMS

Dụng Văn Lũ[1*], Lê Lệ Hằng[2]

[1]Trường Đại học Sư phạm – Đại học Đà Nẵng
[2]Trường Đại học Kinh tế - Kỹ thuật Công nghiệp

*Tác giả liên hệ: dvlu@ued.udn.vn

(Nhận bài: 26/4/2022; Chấp nhận đăng: 11/7/2022)

Tóm tắt - Trong bài báo này, nhóm tác giả thực thi các thuật toán lượng tử Deutsch-Jozsa, Bernstein-Vazirani, Simon và Grover, chạy chúng trên máy tính lượng tử IBM thông qua icloud của trình mô phỏng Qiskit (Qiskitv0.35.0). Nhóm tác giả sử dụng ngôn ngữ lập trình python để mô tả mạch lượng tử của hệ gồm 5 qubit và mô phỏng kết quả đo được (ở dạng xác suất) ứng với mỗi thuật toán trên. Kết quả thực hiện cho thấy, các thuật toán lượng tử có số lần truy vấn ít hơn và tối ưu hơn thuật toán cổ điển vì chúng hoạt động dựa trên tính chất của cơ học lượng tử (tính chồng chất và vướng víu lượng tử). Các thuật toán này tạo cơ sở ý tưởng để xây dựng các thuật toán tối ưu hơn có thể giải các bài toán phức tạp hơn như phân phối khóa lượng tử, sửa lỗi lượng tử, tìm kiếm không cấu trúc, hệ phá mật mã khoá công khai.

Từ khóa - Thuật toán lượng tử; thuật toán Deutsch; thuật toán Bernstein-Vazirani; thuật toán Simon; thuật toán Grover

Abstract - In this paper, the authors implement Deutsch-Jozsa's, Bernstein-Vazirani's, Simon's and Grover's quantum algorithms, run them on IBM quantum lab (Qiskitv0.35.0). The authors use python programming language to describe the quantum circuit of the system of 5 qubits and return the measurement results (probability) for each of the above algorithms. The performance results show that, the quantum algorithms have fewer queries and are more optimal than the classical algorithms because they operate based on the properties of quantum mechanics (superposition and quantum entanglement). These algorithms help us to create more optimal algorithms that can solve more complex problems such as quantum key distribution, quantum error correction, unstructured search, breaking public-key cryptography schemes.

Key words - Quantum algorithm; Deutsch's algorithm; Bernstein-Vazirani; Simon's algorithm; Grover's algorithm

1. Giới thiệu

Ý tưởng về điện toán lượng tử lần đầu tiên được đề xuất độc lập bởi các nhà khoa học Benioff [1], Manin [2] và Feynman [3] vào những năm đầu thập niên 80. Benioff xây dựng một mô hình cơ học lượng tử vi mô của máy tính được biểu diễn bởi máy Turing có sử dụng trạng thái dừng thoả mãn phương trình Schrodinger [1]. Nhà toán học Manin cho rằng không gian trạng thái lượng tử có dung lượng rất lớn so với không gian cổ điển vì nó có sự chồng chất trạng thái, tức là tổ hợp của các trạng thái cơ sở, nên mô hình toán học của nó đòi hỏi phải sử dụng các nguyên lí lượng tử [2]. Năm 1981, Feynman đặt vấn đề: Loại máy tính nào chúng ta sẽ sử dụng để mô phỏng vật lí? Liệu vật lí có thể được mô phỏng tốt bởi một máy tính phổ thông không? Ông cho rằng thế giới vật lí là cơ học lượng tử do đó vấn đề thích hợp là mô phỏng vật lí lượng tử, mà máy tính sẽ hoạt động trên cơ chế này [3].

Khác với điện toán cổ điển hoạt động trên cơ chế điện tử, trong đó đơn vị thông tin là các bit 1 hoặc 0 được thực hiện về mặt vật lí dưới dạng bật và tắt của các bóng bán dẫn riêng lẻ và trạng thái được mô tả bằng một chuỗi nhị phân (10100110…), điện toán lượng tử sử dụng hai trạng thái lượng tử cơ bản $|1\rangle$ và $|0\rangle$ làm đơn vị thông tin gọi là các bit lượng tử (qubit) kết hợp với nguyên lí chồng chất (superposition principle) và vướng víu lượng tử (quantum entanglement) trong cơ học lượng tử [4]. Hệ lượng tử cho qubit có thể là điện tử với hai trạng thái spin (xuống -1 hoặc lên -0), hay trong hệ lượng tử hai mức năng lượng (kích thích -1, cơ bản -0), hay trạng thái photon phân cực. Nhờ nguyên lí chồng chất mà ta có thể biểu diễn trạng thái của một qubit thông qua hàm sóng $\psi = a|1\rangle + b|0\rangle$. Trong đó, các biên độ lượng tử a và b là các số phức tùy ý thỏa mãn điều kiện chuẩn hóa $|a|^2 + |b|^2 = 1$, đồng thời $|a|^2$ và $|b|^2$ cho ta xác suất để qubit ψ lần lượt ở trạng thái $|1\rangle$ và $|0\rangle$. Trái ngược với bit cổ điển chỉ có thể ở một trong hai trạng thái 1 hoặc 0, qubit có thể ở trong một chuỗi liên tục của các trạng thái được xác định bởi các biên độ lượng tử a và b do tính chồng chất. Với N qubit, có 2^N trạng thái cơ bản, do đó trạng thái chung của N qubit tăng theo cấp số nhân và được xác định bởi một hàm sóng với 2^N biên độ lượng tử. Ví dụ, nếu có 3 qubit thì sẽ có $2^3 = 8$ trạng thái cơ bản: {$|000\rangle$, $|001\rangle$, $|010\rangle$, $|100\rangle$, $|011\rangle$, $|101\rangle$, $|110\rangle$ và $|111\rangle$}. Trạng thái chung của 3 qubit được mô tả bằng hàm sóng $|\psi\rangle = a|000\rangle + b|001\rangle + c|010\rangle + d|100\rangle + e|011\rangle\ f|101\rangle + g|110\rangle + h|111\rangle$, với $|a|^2 + |b|^2 + |c|^2 + |d|^2 + |e|^2 + |f|^2 + |g|^2 + |h|^2 = 1$. Khi một trạng thái lượng tử không thể tách rời thành các trạng thái độc lập thì gọi là vướng víu. Tức là, một trạng thái $|\psi\rangle$ vướng víu sẽ không thể biểu diễn ở dạng: $|\psi_1\rangle \otimes |\psi_2\rangle$, với các hàm sóng $|\psi_1\rangle$ và $|\psi_2\rangle$ cho hai hệ con, và $\otimes$ là tích tensor hai trạng thái. Ví dụ, trạng thái vướng víu của hệ 3 qubit có thể là $|GHZ\rangle = (|000\rangle + |111\rangle)/\sqrt{2}$, hay $|W\rangle = (|001\rangle + |010\rangle + |100\rangle)/\sqrt{3}$. Sự vướng víu xảy ra với các trạng thái đa qubit, khi đó trạng thái kết hợp của các qubit chứa nhiều thông tin hơn các qubit hoạt động độc lập. Với một cặp đối tượng lượng tử vướng víu, một phép đo được thực hiện trên một đối tượng lượng tử này sẽ ngay lập tức có ảnh hưởng đến đối tượng kia mà không cần chờ đợi độ trễ trong quá trình truyền tin.

[1] The University of Danang - University of Science and Education (Dung Van Lu)
[2] University of Economics - Technology for Industries (Le Le Hang)

Như vậy, tốc độ tính toán của điện toán lượng rất nhanh vì nó có khả năng tính toán song song do trạng thái chồng chất với các tham số phức tuỳ ý. Do đó, máy tính lượng tử hoạt động trên cơ chế này là một máy tương tự làm việc với các tham số liên tục, trái ngược với các máy tính kỹ thuật số thông thường hoạt động với các tham số rời rạc. Máy tính lượng tử phổ dụng hiện nay sử dụng mô hình mạch lượng tử (quantum circuit model). Mạch lượng tử gồm các cổng lượng tử, các lệnh (thuật toán lượng tử) và logic điều khiển cổ điển. Các thuật toán lượng tử hoạt động nhờ các tính chất cơ học lượng tử nên số bước ít hơn dẫn đến thời gian thực hiện nhanh hơn so với thuật toán cổ điển. Những thuật toán lượng tử đầu tiên là Deutsch-Jozsa [5], Bernstein-Vazirani [6], Simon [7] và Grover [8]. Các thuật toán này tạo cơ sở ý tưởng để xây dựng các thuật toán tối ưu hơn có thể giải các bài toán phức tạp hơn như phân phối khóa lượng tử, sửa lỗi lượng tử, tìm kiếm không cấu trúc, hệ phá mật mã khoá công khai

Năm 2018, Mandviwalla và các cộng sự đã thử nghiệm hoạt động của các thuật toán trên máy tính lượng tử và chỉ ra sự khác biệt đáng chú ý giữa các lựa chọn khác nhau của các qubit trong thiết kế triển khai và giữa các thiết bị lượng tử khác nhau thực thi thuật toán. Nhưng nghiên cứu này chỉ thực thi với thuật toán Grover và với 4 qubit [9].

Trong nghiên cứu này, nhóm tác giả thực thi các thuật toán lượng tử Deutsch-Jozsa, Bernstein-Vazirani, Simon và Grover trên máy tính lượng tử IBM thông qua icloud của trình mô phỏng Qiskit (Qiskitv0.35.0).

2. Các thuật toán lượng tử cơ bản

2.1. Thuật toán Deutsch-Jozsa

Thuật toán lượng tử Deutsch-Jozsa [5] là một thuật toán xác định, tức luôn luôn trả về đáp án và đáp án luôn chính xác bởi một truy vấn. Nó là thuật toán đầu tiên cho thấy sự khác biệt về độ phức tạp tính toán giữa lượng tử và cổ điển. Thuật toán này thể hiện tầm quan trọng của việc cho phép biên độ lượng tử nhận cả giá trị âm và dương, trái ngược với các xác suất cổ điển luôn không âm.

Bài toán Deutsch-Jozsa được định nghĩa như sau. Xét một hàm với vào là một chuỗi n ký tự nhị phân và trả về giá trị 0 hoặc 1, tức là hộp đen thực thi hàm $f: \{0,1\}^n \rightarrow \{0,1\}$, nhiệm vụ là xác định xem liệu f là bất biến (hoặc nhận toàn 0, hoặc nhận toàn 1) hay cân bằng (một nửa nhận 0 và một nửa nhận 1). Vì tổng số đầu vào có thể có là 2^n nên về mặt cổ điển, trường hợp xấu nhất chúng ta cần $2^{n-1}+1$ phép thử để chắc chắn rằng $f(x)$ là bất biến. Trong khi thuật toán Deutsch-Jozsa có thể giải quyết vấn đề này với độ tin cậy 100% chỉ sau một lần gọi hàm $f(x)$ trong một hộp đen lượng tử (hay oracle). Mô hình toán học cho thuật toán Deutsch-Jozsa được thể hiện ở Hình 1 và nó hoạt động qua các bước sau [10]:

(1) Chuẩn bị hai thanh ghi lượng tử. Một thanh ghi n-qubit được khởi tạo ở trạng thái $|0\rangle$, và thanh ghi thứ hai là một qubit được khởi tạo $|1\rangle$: $|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$.

(2) Áp dụng cổng Hadamard H [4] cho mỗi qubit:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle\,(|0\rangle - |1\rangle).$$

(3) Áp dụng hộp đen lượng tử (oracle) $|x\rangle|y\rangle$ thành $|x\rangle|y\oplus f(x)\rangle$:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle\,(|0\rangle - |1\rangle).$$

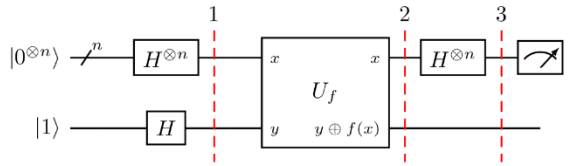
(4) Bỏ qua thanh ghi qubit thứ hai. Áp dụng cổng Hadamard cho mỗi qubit ở thanh ghi thứ nhất:

$$|\psi_3\rangle = \frac{1}{2^n}\sum_{y=0}^{2^n-1}\Big[\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{xy}\Big]\,|y\rangle.$$

(5) Đo thanh ghi thứ nhất. Chú ý rằng xác suất đo

$$|0\rangle^{\otimes n} = \left|\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}\right|^2$$

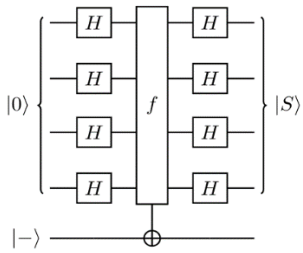
cho kết quả 1 nếu $f(x)$ bất biến và 0 nếu $f(x)$ là cân bằng. Như vậy, ta chỉ có đầu vào, một lần truy vấn vào hộp đen lượng tử và thực hiện phép đo là đã thu được kết quả loại hàm f cần tìm.



Hình 1. *Mô hình toán học thuật toán Deutsch-Jozsa*

2.2. Thuật toán Bernstein-Vazirani

Thuật toán Bernstein-Vazirani [6] có thể được coi là một phần mở rộng của thuật toán Deutsch-Jozsa. Nó chỉ ra rằng, có thể có lợi thế khi sử dụng máy tính lượng tử như một công cụ tính toán cho các bài toán phức tạp hơn bài toán Deutsch-Jozsa. Bài toán Bernstein-Vazirani được mô tả như sau: Một hàm hộp đen f nhận đầu vào là một chuỗi bit $\{x\}$ và trả về 0 hoặc 1, nghĩa là: $f(\{x_0, x_1, x_2,...\}) \rightarrow 0$ hoặc 1, trong đó x_n là 0 hoặc 1. Hàm được đảm bảo trả về tích số bit của đầu vào với một số chuỗi s. Nói cách khác, với một đầu vào x: $f(x) = s \cdot x \ (mod\ 2)$, chúng ta cần tìm s. Về mặt cổ điển, chúng ta sẽ cần n lần gọi hàm $f_s(x)$.



Hình 2. *Mô hình thuật toán Bernstein-Vazirani*

Thuật toán lượng tử Bernstein-Vazirani để tìm chuỗi bit ẩn rất đơn giản, chỉ sau một lần gọi hàm $f(x)$. Hình 2 biểu diễn mô hình cho thuật toán này và nó hoạt động qua các bước [10]:

(1) Khởi tạo các qubit đầu vào ở trạng thái $|0\rangle^{\otimes n}$, và xuất qubit thành $|-\rangle$.

(2) Áp dụng các cổng Hadamard lên thanh ghi đầu vào.

(3) Truy vấn hộp đen.

(4) Áp dụng các cổng Hadamard cho thanh ghi đầu vào.

(5) Đo kết quả.

Có thể vắn tắt các bước thông qua chuỗi sau:

$$|0^{\otimes n}\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}}|x\rangle \xrightarrow{f_s} \frac{1}{\sqrt{2^n}}\sum(-1)^{s\,x}|x\rangle \xrightarrow{H^{\otimes n}} |s\rangle.$$

Như vậy, ở bước cuối cũng, ta chỉ cần đo trạng thái $|s\rangle$. Theo cách hoạt động này, ta có thể xây dựng thuật toán sửa lỗi lượng tử.

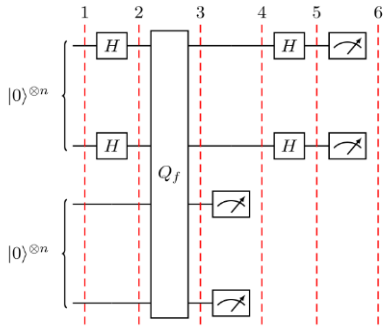
2.3. Thuật toán Simon

Bài toán Simon là một bài toán thuộc dạng cây quyết định hay dạng truy vấn, được diễn tả bởi Daniel Simon năm 1994 [7]. Simon đã đưa ra một thuật toán lượng tử để giải quyết bài toán nhanh hơn rất nhiều (số mũ lần) so với bất kì thuật toán xác định hay xác suất cổ điển nào. Bài toán Simon được mô tả như sau. Có một hàm hộp đen không xác định *f*, được đảm bảo là một-một (1-1) hoặc hai-một (2-1), trong đó các hàm một-một và hai-một có các thuộc tính sau:

(i) Một-một: Ánh xạ chính xác mỗi đầu vào khác nhau cho một đầu ra. Ví dụ với một hàm có 4 đầu vào là: $f(1) \rightarrow 1$, $f(2) \rightarrow 2$, $f(3) \rightarrow 3$, $f(4) \rightarrow 4$.

(ii) Hai-một: Ánh xạ chính xác mỗi hai đầu vào cho cùng một kết quả đầu ra. Ví dụ với một hàm nhận 4 đầu vào là: $f(1) \rightarrow 1$, $f(2) \rightarrow 2$, $f(3) \rightarrow 1$, $f(4) \rightarrow 2$. Ánh xạ hai-một này theo một ẩn chuỗi bit *b*, trong đó: cho trước *x*, *y*: $f(x) = f(y)$ với $y = x \oplus b$.

Với hộp đen *f* này, làm thế nào để có thể xác định *f* là một-một hay hai-một. Nếu *f* là hai-một, thì làm thế nào để có thể xác định *b*? Theo cổ điển, nếu chúng ta muốn biết *b* với độ chắc chắn 100% đối với *f* đã cho, chúng ta phải kiểm tra tối đa $(2^{n-1}+1)$ đầu vào, trong đó *n* là số bit trong đầu vào, tức là độ phức tạp luỹ thừa bậc *n*. Trong khi thuật toán Simon chỉ cần tối đa *n* bước lặp. Hình 3 là mô hình toán học thuật toán Simon, nó thực hiện qua các bước sau:



Hình 3. *Mô hình thuật toán Simon*

(1) Khởi tạo hai thanh ghi đầu vào n-qubit ở trạng thái 0: $|\psi_1\rangle = |0\rangle^{\otimes n}|0\rangle^{\otimes n}$.

(2) Áp dụng các cổng Hadamard cho thanh ghi thứ nhất:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle|0\rangle^{\otimes n}.$$

(3) Áp dụng hàm truy vấn Q_f:

$$|\psi_3\rangle = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle|f(x)\rangle.$$

(4) Đo thanh ghi thứ hai. Giá trị của $f(x)$ sẽ được quan sát. Do cách thiết lập của bài toán, giá trị quan sát $f(x)$ có thể tương ứng với hai đầu vào có thể có: *x* và $y = x \oplus b$. Do đó, thanh ghi đầu tiên trở thành: $|\psi_4\rangle = (|x\rangle + |y\rangle)/\sqrt{2}$, trong đó bỏ qua thanh ghi thứ hai vì nó đã được đo.

(5) Áp dụng Hadamard trong thanh ghi đầu tiên:

$$|\psi_5\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{z\in\{0,1\}^n}[(-1)^{xz} + (-1)^{yz}]|z\rangle.$$

(6) Việc đo thanh ghi đầu tiên sẽ chỉ cho kết quả đầu ra nếu $(-1)^{xz} = (-1)^{yz}$.

Thuật toán Simon đã tạo ra một sự tăng tốc theo hàm mũ so với các thuật toán cổ điển và cũng là cơ sở lý tưởng để xây dựng các thuật toán phá hệ mã khoá công khai.

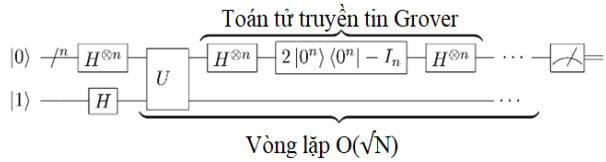
2.4. Thuật toán tìm kiếm Grover

Thuật toán của Grover [8] thể hiện khả năng tìm kiếm cơ sở dữ liệu với tốc độ vượt trội của nó. Thuật toán này có thể tăng tốc độ một bài toán tìm kiếm không có cấu trúc theo phương pháp bậc hai, nhưng việc sử dụng nó còn mở rộng hơn thế nữa. Ví dụ, để giải quyết một vấn đề thỏa mãn Boolean tổ hợp cụ thể, tối ưu cho xác suất trường hợp xấu nhất (3-SAT). Hay các thuật toán để ước tính tài nguyên lượng tử, tiêu chuẩn mã hóa nâng cao (AES) cũng như giao thức chia sẻ bí mật lượng tử, tối ưu hóa độ sâu của các thuật toán tìm kiếm lượng tử như giải mã lược đồ mã hóa DES [11]. Ý tưởng của thuật toán Grover có những điểm vượt trội đó là thủ thuật khuếch đại biên độ bằng cách đổi pha mục cần tìm, thực hiện phép quay lặp nhiều lần để tăng biên độ, kết quả đo biên độ của trạng thái nào lớn nhất chính là đáp án. Hình 4 là mô hình toán học của thuật toán Grover, nó có thể hoạt động qua các bước sau:

(1) Bắt đầu khuếch đại biên độ trong chồng chất đồng nhất $|s\rangle = H^{\otimes n}|0\rangle^n$.

(2) Áp dụng phản xạ hộp đen U_f cho trạng thái $|s\rangle$.

(3) Thực hiện một phản xạ bổ sung U_s về trạng thái $|s\rangle$: $U_s = 2|s\rangle\langle s| - 1$. Phép biến đổi này ánh xạ trạng thái thành $U_sU_f|s\rangle$ và hoàn thành phép biến đổi. Có thể thấy, biên độ tăng tuyến tính với số lần quay (vòng lặp) tỉ lệ với $t/\sqrt{N}$. Tuy nhiên, vì đang xử lý các biên độ chứ không phải xác suất, do đó, biên độ được khuếch đại trong quy trình này.



Hình 4. *Mô hình thuật toán Grover*

Nhìn chung, quy trình làm việc của thuật toán lượng tử điển hình bao gồm:

(1) Vấn đề cần giải quyết;

(2) Một thuật toán cổ điển tạo ra một mô tả về một mạch lượng tử;

(3) Mạch lượng tử cần được chạy trên phần cứng lượng tử;

(4) Và phép đo cổ điển áp dụng ở đầu ra.

Ở mục tiếp theo, nhóm tác giả thực hiện code python mô phỏng để vẽ các mạch lượng tử và thực hiện đo kết quả cho các thuật toán trên với đầu vào 5 qubits sử dụng mã nguồn mở của Qiskit chạy trên icoud IMB [10].

3. Kết quả thực hiện và thảo luận

Các thuật toán lượng tử phải được thực thi trên máy tính lượng tử, nhưng ở Việt Nam chưa có máy tính lượng tử, nên phải chạy trên nền icloud của các máy tính lượng tử thật của các hãng máy tính lớn như IBM, Google, Microsoft… Trong số đó, có các môi trường lập trình lượng tử khác nhau với các mã nguồn mở [12]: Qiskit từ IBM,

Forest (pyQuil) từ Rigetti, Q# từ Microsoft và ProjectQ từ ETH, hay máy ảo lượng tử Qsun. Nhóm tác giả đã thử nghiệm trên các trình khác nhau, Forest là ngôn ngữ chương trình lượng tử mã nguồn mở được phát triển bởi Rigetti bao gồm pyQuil dựa trên Quil, tuy nhiên câu lệnh pyQuil phức tạp và hình ảnh mạch lượng tử xuất dạng text nên không hài hoà [12]. Đối với Microsoft thì hiện nay không có thiết bị nào mà người dùng có thể kết nối thông qua bộ phát triển lượng tử, đồng thời, cú pháp của Q# khá khác so với các ngôn ngữ lập trình, nó gần giống với C# và dài dòng hơn Python. Trong khi ProjectQ chỉ sử dụng ngôn ngữ Python, tài liệu hướng dẫn của ProjectQ rời rạc và không đề cập đến các thuật toán cơ bản, nên rất khó cho những người mới bắt đầu. Với Qsun thì chưa có hướng dẫn cụ thể, khó tiếp cận và chỉ chạy trên máy tính lượng tử ảo [13]. Bên cạnh đó, Qiskit sử dụng ngôn ngữ lập trình Python, JavaScript and Swift rất tiện lợi cho người dùng, đồng thời có tóm tắt nội dung lí thuyết và tiếp cận từ những thuật toán cơ bản nhất rất hữu ích cho người dùng. Vì vậy, nhóm tác giả sử dụng mã nguồn mở trên nền Qiskit, dùng code python và cải tiến một số lệnh để thực thi các thuật toán tương ứng.

Khi sử dụng Qiskit, quy trình làm việc của người dùng về danh nghĩa bao gồm bốn bước sau:

(1) Xây dựng: Thiết kế (các) mạch lượng tử biểu diễn cho vấn đề cần giải quyết.

(2) Biên dịch: Biên dịch các mạch cho một máy chủ lượng tử cụ thể.

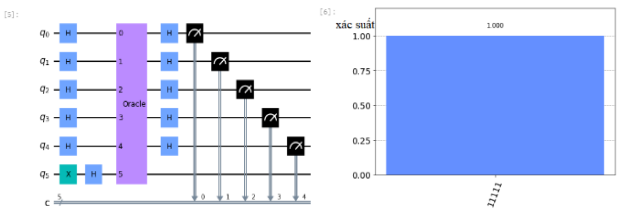
(3) Chạy: Chạy các mạch đã biên dịch trên (các) máy chủ lượng tử được chỉ định. Ở đây máy chủ này dựa trên đám mây.

(4) Phân tích: Tính toán số liệu thống kê tóm tắt và hình dung kết quả của các thí nghiệm.

Nhóm tác giả thực thi thuật toán Deutsch-Jozsa cho 5 qubit sử dụng code lập trình python chạy trên icloud của máy tính lượng tử IBM [10] theo những dòng lệnh như sau:

```
import numpy as np
from qiskit import IBMQ, Aer
from qiskit.providers.ibmq import least_busy
from qiskit import QuantumCircuit, assemble, transpile
from qiskit.visualization import plot_histogram
def dj_oracle(case, n):
    oracle_qc = QuantumCircuit(n+1)
    if case == "balanced":
        b = np.random.randint(1,2**n)
        b_str = format(b, '0'+str(n)+'b')
        for qubit in range(len(b_str)):
            if b_str[qubit] == '1':
                oracle_qc.x(qubit)
        for qubit in range(n):
            oracle_qc.cx(qubit, n)
        for qubit in range(len(b_str)):
            if b_str[qubit] == '1':
                oracle_qc.x(qubit)
    if case == "constant":
        output = np.random.randint(2)
        if output == 1:
            oracle_qc.x(n)
    oracle_gate = oracle_qc.to_gate()
    oracle_gate.name = "Oracle" # To show when we display the circuit
    return oracle_gate
def dj_algorithm(oracle, n):
    dj_circuit = QuantumCircuit(n+1, n)
    dj_circuit.x(n)
    dj_circuit.h(n)
    for qubit in range(n):
        dj_circuit.h(qubit)
    dj_circuit.append(oracle, range(n+1))
    for qubit in range(n):
        dj_circuit.h(qubit)
    for i in range(n):
        dj_circuit.measure(i, i)
    return dj_circuit
n = 5
oracle_gate = dj_oracle('balanced', n)
dj_circuit = dj_algorithm(oracle_gate, n)
dj_circuit.draw() # vẽ mạch lượng tử
#----- chạy đoạn code trên trước để vẽ mạch lượng tử; sau đó mới chạy đoạn code sau:
aer_sim = Aer.get_backend('aer_simulator') #tiếp tục đoạn code này để đo kết quả.
qobj = assemble(dj_circuit, aer_sim)
transpiled_dj_circuit = transpile(dj_circuit, aer_sim)
qobj = assemble(transpiled_dj_circuit)
results = aer_sim.run(qobj).result()
answer = results.get_counts()
plot_histogram(answer) #trả kết quả (xác suất đo) dạng đồ thị.
```

Kết quả được thể hiện ở Hình 5. Mạch lượng tử gồm hai thanh ghi, thanh ghi thứ nhất gồm 5 qubit được đánh dấu q_0 đến q_4, thanh ghi thứ hai là một qubit được đánh dấu là q_5. Kết quả đo trả về với xác suất p =100% của đầu vào "11111", nghĩa là hàm *f* trong oracle là hàm bất biến. Theo cách hoạt động này, thuật toán Deutsch-Jozsa có thể được sử dụng để phân phối khóa lượng tử bằng cách sử dụng trạng thái vướng víu GHZ.



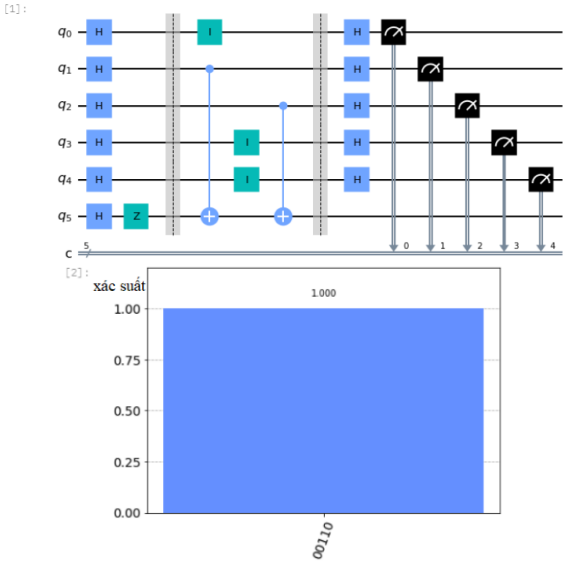
Hình 5. *Mạch lưởng tử và kết quả trả về của thuật toán Deustch-Jozsa*

Với thuật toán Bernstein-Vazirani, ta sử dụng đoạn code sau:

```
import matplotlib.pyplot as plt
import numpy as np
from qiskit import IBMQ, Aer
from qiskit.providers.ibmq import least_busy
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, transpile, assemble
from qiskit.visualization import plot_histogram
n = 5 # Số qubit để biểu diễn
s = '00110'   #chuỗi nhị phân ẩn
bv_circuit = QuantumCircuit(n+1, n)
bv_circuit.h(n)
bv_circuit.z(n)
for i in range(n):
    bv_circuit.h(i)
bv_circuit.barrier()
```

```
    s = s[::-1]
    for q in range(n):
        if s[q] == '0':
            bv_circuit.i(q)
        else:
            bv_circuit.cx(q, n)
    bv_circuit.barrier()
    for i in range(n):
        bv_circuit.h(i)
    for i in range(n):
        bv_circuit.measure(i, i)
    bv_circuit.draw() #vẽ mạch lượng tử
    #---dừng đến đây để vẽ mạch lượng tử, sau đó xuất kết quả đo
    aer_sim = Aer.get_backend('aer_simulator')
    shots = 1024
    qobj = assemble(bv_circuit)
    results = aer_sim.run(qobj).result()
    answer = results.get_counts()
    plot_histogram(answer)
```

Hình 6 thể hiện kết quả thực thi thuật toán Bernstein-Vazirani, hình bên trái là mạch lượng tử với 5 qubit và hình bên phải là kết quả trả về của chuỗi *s* là "00110" với xác suất 100%. Thuật toán này dùng trong truyền thông và sửa lỗi lượng tử. Ngoài ra, sử dụng nhiều hệ lượng tử song song cùng với thuật toán này để tính toán nhiều hàm cùng một lúc.

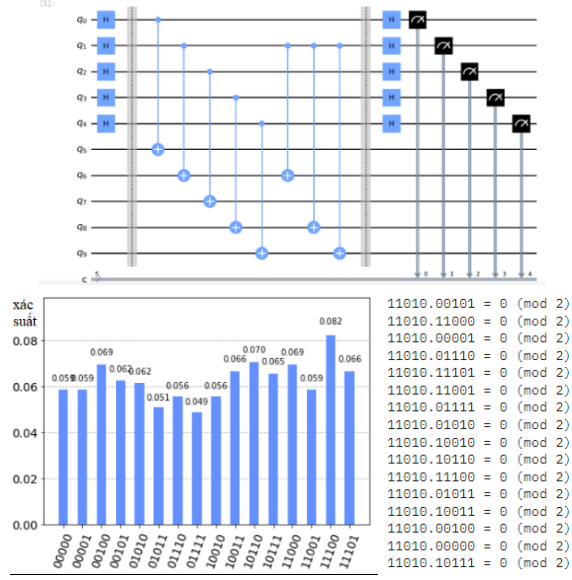


Hình 6. *Mạch lưởng tử (trên) và kết quả trả về (dưới) của thuật toán Bernstein-Vazirani*

Nhóm tác giả sử dụng code sau cho thuật toán Simon:

```
    from qiskit import IBMQ, Aer
    from qiskit.providers.ibmq import least_busy
    from qiskit import QuantumCircuit, transpile, assemble
    from qiskit.visualization import plot_histogram
    from qiskit_textbook.tools import simon_oracle
    b = '11010'
    n = len(b)
    simon_circuit = QuantumCircuit(n*2, n)
    simon_circuit.h(range(n))
    simon_circuit.barrier()
    simon_circuit += simon_oracle(b)
    simon_circuit.barrier()
    simon_circuit.h(range(n))
    simon_circuit.measure(range(n), range(n))
    simon_circuit.draw()
    # Sau đó xuất kết quả
    aer_sim = Aer.get_backend('aer_simulator')
    shots = 1024
    qobj = assemble(simon_circuit, shots=shots)
    results = aer_sim.run(qobj).result()
    counts = results.get_counts()
    plot_histogram(counts)
    # Sau đó, thêm các code sau để tính tích bên trong của hai chuỗi
    def bdotz(b, z):
        accum = 0
        for i in range(len(b)):
            accum += int(b[i]) * int(z[i])
        return (accum % 2)
    for z in counts:
        print( '{}.{} = {} (mod 2)'.format(b, z, bdotz(b,z)) )
```

Hình 7 mô tả mạch lượng tử của thuật toán Simon với 5 qubit và trả về kết quả của chuỗi z với các xác suất tương ứng, trong đó chuỗi z "11100" cho xác suất cao nhất, 8,2%.



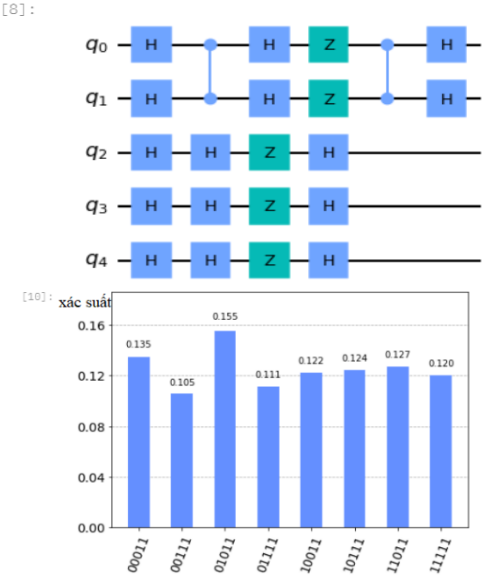
Hình 7. *Mạch lưởng tử (trên); kết quả trả về ứng với xác suất (dưới trái) và xuất chuỗi tring z (dưới phải) của thuật toán Simon*

Cuối cùng là thuật toán Grover, ta cho thực thi với đoạn code sau:

```
    import matplotlib.pyplot as plt
    import numpy as np
    from qiskit import IBMQ, Aer, assemble, transpile
    from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
    from qiskit.providers.ibmq import least_busy
    from qiskit.visualization import plot_histogram
    n = 5
    grover_circuit = QuantumCircuit(n)
    def initialize_s(qc, qubits):
        """Áp dụng cổng H cho qubit"""
        for q in qubits:
            qc.h(q)
        return qc
    grover_circuit = initialize_s(grover_circuit, [0,1,2,3,4])
    grover_circuit.cz(0,1) # Oracle
    grover_circuit.h([0,1,2,3,4])
    grover_circuit.z([0,1,2,3,4])
    grover_circuit.cz(0,1)
```

```
grover_circuit.h([0,1,2,3,4])
grover_circuit.draw()
#-dừng ở đây để vẽ mạch, tiếp theo đo xác suất.
sim = Aer.get_backend('aer_simulator')
grover_circuit_sim = grover_circuit.copy()
grover_circuit_sim.save_statevector()
qobj = assemble(grover_circuit_sim)
result = sim.run(qobj).result()
statevec = result.get_statevector()
grover_circuit.measure_all()
aer_sim = Aer.get_backend('aer_simulator')
qobj = assemble(grover_circuit)
result = aer_sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)
```

Kết quả thực thi được thể hiện ở Hình 8 với 5 qubit và trả về các kết quả của chuỗi *s* tương ứng với xác suất khác nhau. Trong đó chuỗi kết quả "01011" cho kết quả với xác suất cao nhất, 15,5%.



Hình 8. *Mạch lưởng tử (trên) và kết quả trả về (dưới) của thuật toán tìm kiếm Grover*

Thuật toán Grover có nhiều áp dụng, như giải các bài toán 3-SAT hay giải trò chơi sudoku gồm các ô latinh với quy tắc: Không cột nào, không hàng nào và không nhóm hình nào có thể chứa cùng một giá trị hai lần. Thuật toán tìm kiếm lượng tử có thể là một lựa chọn hấp dẫn và thích hợp trong kỷ nguyên công nghệ lượng tử ồn ào quy mô trung gian [14].

Việc triển khai trên các thuật toán lượng tử trên nền Qiskit rất tiện lợi cho các nghiên cứu về thuật toán. Ta thấy, thời gian chạy các thuật toán trên với 5 qubit là rất nhanh. Vậy khi số qubit tăng lên gấp nhiều lần thì thời gian chạy sẽ như thế nào? Mức độ hiệu quả và ứng dụng cụ thể của các thuật toán trên như thế nào? Nhóm tác giả sẽ thực hiện nó trong nghiên cứu tiếp theo.

4. Kết luận

Nhóm tác giả đã thực thi các thuật toán lượng tử trên nền Qiskit. Kết quả cho thấy, với số lần truy vấn rất ít nên thuật toán lượng tử xử lí rất nhanh so với thuật toán cổ điển. Có những bài toán với hệ *n*-bit thì thuật toán cổ điển phải dùng $2^{n-1}+1$ lần truy vấn, trong khi thuật toán lượng tử chỉ cần 1 lần truy vấn vì nó có thể thực hiện đồng thời các trạng thái với kết quả là sác xuất nhờ tính chồng chất trạng thái. Bốn thuật toán này dù cơ bản nhưng có những ứng dụng giải quyết bài toán đã phân tích ở trên và đặc biệt là có ảnh hưởng lớn cho các phát triển thuật toán có ứng dụng lớn và thể hiện uy quyền lượng tử như thuật toán sửa lỗi lượng tử, thuật toán Shor phá mã khoá công khai (RSA). Trong các nghiên cứu tiếp theo, nhóm tác giả sẽ cải tiến thuật toán Shor và xây dựng các mã sửa lỗi mới.

Lời cảm ơn: Nghiên cứu này được tài trợ bởi Quỹ phát triển Khoa học Trường Đại học Sư phạm – Đại học Đà Nẵng trong đề tài có mã số T2022-TN-09.

TÀI LIỆU THAM KHẢO

[1] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines", *J. Stat. Phys*. 22, 1980, 563-591.

[2] Yu. I. Manin, "Vychislimoe i nevychislimoe" [Computable and Noncomputable] (in Russian), *Sov. Radio*, 1980, 13-15.

[3] R. Feynman, "Simulating physics with computers". *International Journal of Theoretical Physics*, 21(6–7), 1982, 467-488.

[4] G. Benenti, G. Casati and G. Strini, *Principles of quantum computation and information*, World Scientific, 2004.

[5] D. Deutsch and R. Jozsa, "Rapid solutions of problems by quantum computation", *Proceedings of the Royal Society of London A*, 439, 1992 553–558.

[6] E. Bernstein and U. Vazirani, "Quantum Complexity Theory", *SIAM Journal on Computing*, 26(5), 1997, 1411-1473.

[7] D. R. Simon, "On the Power of Quantum Computation", *SIAM Journal on Computing*, 26(5), 1997, 1474–1483.

[8] L. K. Grover, "A fast quantum mechanical algorithm for database search", *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing* (STOC), 1996, 212-219.

[9] A. Mandviwalla, K. Ohshiro and B. Ji, "Implementing Grover's Algorithm on the IBM Quantum Computers", *IEEE International Conference on Big Data*, 2018, 2531-2537.

[10] Amira Abbas et al., "Learn Quantum Computation Using Qiskit", *Community.qiskit.org/textbook*, 2020 [online] https://lab.quantum-computing.ibm.com, 20/04/2022.

[11] M. Grassl, B. Langenberg, M. Roetteler, R. Steinwandt, "Applying Grover's Algorithm to AES: Quantum Resource Estimates", *Post-Quantum Cryptography*, 9606, 2016, 29-43.

[12] R. LaRose, "Overview and Comparison of Gate Level Quantum Software Platforms", arXiv:1807.02500v2, 2019, 10/06/2022.

[13] Quoc C Nguyen, Le Bin Ho, Lan Nguyen Tran, Hung Q Nguyen, "Qsun: an open-source platform towards practical quantum machine learning applications", *Mach. Learn.: Sci. Technol*. 3, 2022, 015034.

[14] Kunal Das, Arindam Sadhu, "Experimental study on the quantum search algorithm over structured datasets using IBMQ experience", *Journal of King Saud University - Computer and Information Sciences*, 2022, 12, https://doi.org/10.1016/j.jksuci.2022.01.012.