隨機性對於量子糾錯算法的重要性及適用於 IBM Quantum 的量子糾錯算法

I. Abstract

迄今為止還沒有一個完美的量子電腦系統，量子位元總是存在錯誤。因此，糾錯變得極具挑戰性，需要與傳統方法完全不同的方法。

在我們的研究中，我們使用模擬器來實現 Shor 糾錯碼（SRCC），並針對位元翻轉錯誤問題開發了隨機方法 Shor 糾錯碼（SRCC-R）。我們考慮錯誤率為 0.007 和 0.002 的 IBM 量子電腦系統。 我們的結果表明，使用 SRCC，翻轉錯誤可以糾正少於 25 個 CNOT gate，正確率為 0.97。在我們的方法中，SRCC-R 的翻轉錯誤可以修正超過 500 個 CNOT gate，正確率為 0.95。

II. Implementation and Analysis

A. *Three-Bit Repetition Code*

根據 SRCC 的說法，重複一個量子位元操作三次應該會得到三個相同的結果。然而，操作過程中可能會出現錯誤，導致三個量子位元不同。三位元重複碼是將這三個量子位元統一校正為多數答案。

B. *Grouping Error Correction in Circle*

考慮 IBM 量子電腦系統的結構(圖 1)，與循環結構的肖爾糾錯碼。圖 2 示範如何將 6 -bit(後續實驗將以 20-bit 取代 6-bit)組成一個環並分組以進行糾錯。結果如圖 3 所示，顯然即使在錯誤率較低的情況下，經過 100 次 CNOT gate 操作後，正確率也降到 0.6 以下。假設這是由連續位錯誤引起的，如果連續出現錯誤，就沒有辦法修正。因此，若要打破無限循環，必須打亂採樣序列，於是我們設計了 C 、D 和 E 小節。

C. *Sampling According to Offset: [-2,0,1]*

首先，我們需要在模擬器中找出最佳結果。經過實驗，我們發現依照偏移量：[-2,0,1] 取樣糾錯效果最好（這裡的正負號分別視為順時針和逆時針）。例如，對應圖 2， [4,0,1]為一組。結果如圖 4。當錯誤率低於 0.002 且處於最佳指標時，SRCC 在循環結構中是有用的。然而，它很難在真實設備中實現，因為它無法找到真實設備中的最佳索引。

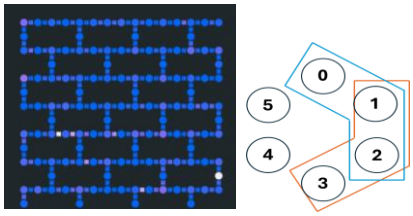

Fig. 1. (left)The arragement of the qubits and the current error rate map of IBM Heron r1 processor[9]

Fig. 2. (right)6 bits grouping error correction in circle

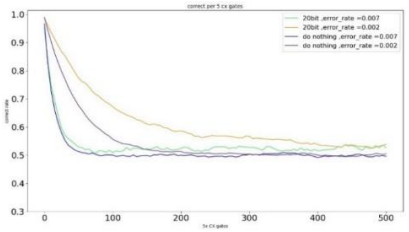

Fig. 3. Results of 6-Bit Grouping Error Correction in Circle. Consider the error rate equal to 0.007 and 0.002 in without error correction (dark blue and green) and SRCC error correction (purple and yellow).

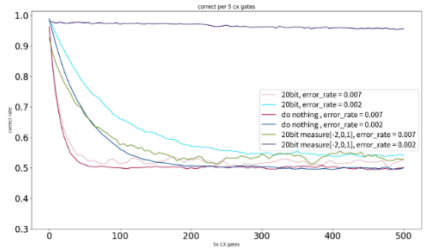

Fig. 4. Results of Sampling according to offset: [-2,0,1]

D. *Write Back After all bits are finished*

儘管我們要求最好的結果並寫回所有量子位元，錯誤仍然出現。「糾正」的量子位元越多，發生的錯誤就越多，如下：

100001 -> 110001 -> 111001…

我們推測原因是當第一組三位重複碼的糾錯結果是錯的，寫回後，將被第二組取樣進行糾錯，於是導致越改越錯。

更好的做法是將每組三位重複碼的糾錯結果暫時儲存到另一組量子位元，以避免每組糾錯互相影響。所有糾錯完成後再將結果寫回。結果如圖 5 所示。正確率比之前好。操作 500 次 CNOT gate 後，即使錯誤率等於 0.007，正確率也高於 0.8。圖五中，在第 100 次操作時圖形出現一個 V，因為我們在這次操作中放入了巨大的誤差(此巨大誤差等於 CNOT gate 的誤差的 20 倍)。因此可以得知，使用我們的方法可以恢復正確率。

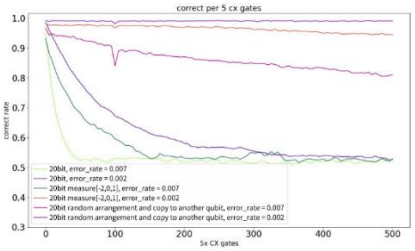

Fig. 5. The results of “write back after all bits are finished”. It is obviously that the correct rate is enhacend. It is should be mention that we give a huge error in 100th controlled-not gates operations in the system. In our method the correct rate will be recovered.

E. *Shuffle Randomly*

根據上述思路，如果我們能夠對隨機選擇的索引量子位元進行打亂，糾錯將會更有效率。考慮到量子位元的實際排列，雖然這種隨機改組的方法在現實中並不存在，但我們可以觀察「隨機改組」是否能夠幫助極大地提高糾錯能力。

以 IBM Brisbane 量子電腦的 qubit 排列為例，示範隨機打亂 qubit 的步驟：

1) 橘色框中的量子位元是待修正的數據，紅色框中的量子位元是用來儲存糾錯結果的額外空間（圖 6 左）。橘色框中的數字 40、41 和 42 是相鄰的量子位元。先取這三個量子位元進行糾錯，結果存入編號為 53 的量子位元中；編號 44、45 和 46 的糾錯結果則存入編號 54。

2) 將圖 6 左的橘框內的量子位元位置倆倆交換兩次(一左一右)。如圖 6 中、圖 6 右所示。

3) 交換完成後，編號 40、41、42 的內容將會是新的組合。將這三個 qubit 糾錯後的結果存到編號為 60；編號 44、45、46 的糾錯結果則存到編號 64。重複步驟 2 與 3，直到儲存空間(圖 6 左的紅框範圍)填滿，即得到所有糾錯結果。

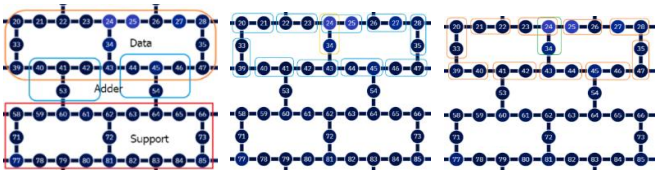

Fig. 6. Steps of Shuffle Randomly

Fig. 8 shows the results of our work.

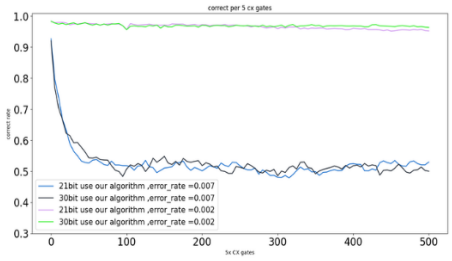

Fig. 7. The results of our algorithm. We can see that even more than 20 bits such as 21 bits and 30 bits can have a good effect when the error rate is 0.002

III. Conclusion

雖然我們提出的所有方法都無法在 0.007 的錯誤率下保持一致的準確率，但如果我們樂觀地使用 IBM 最新的量子電腦錯誤率（0.002）進行實驗，我們的方法確實可以將準確率保持在一定水平（0.95）。在這個過程中，我們也應用了特定時間的顯著錯誤率，看看我們的糾錯方法是否能夠靈活地將準確率恢復到原來的水平。結果非常好；我們的糾錯方法可以在 IBM 的量子電腦上保持一定的準確率。同時，我們也了解隨機排列對於所有糾錯演算法的重要性。今後開發糾錯演算法時，應注意該方法是否會造成隨機排列。