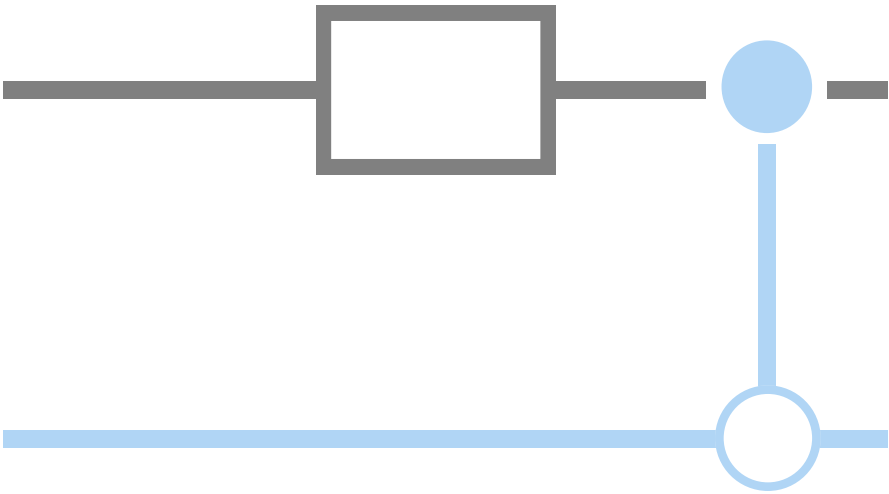
Simulating and Analyzing Two Protocols for the Concentration of Entanglement

Niv Bharos



to obtain the degree of Bachelor of Science in Applied Physics at Delft University of Technology

TU Delft Delft University of Technology

Simulating and Analyzing Two Protocols for the Concentration of Entanglement

by

Niv Bharos

Student number: 4437233
Date of defense: May 17^{th} at 15:30h
Thesis committee: Dr. D. Elkouss Coronas (supervisor), TU Delft
Prof.dr.ir. Ronald Hanson, TU Delft

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

TU Delft Delft University of Technology

Contents

Abstract v

1 Introduction 1

2 Background Information 3
2.1 Quantum States . . . 3
2.1.1 Single Qubit . . . 3
2.1.2 Multiple Qubits . . . 4
2.2 Quantum Gates . . . 5
2.2.1 Single qubit gates . . . 5
2.2.2 Multiple qubit gates . . . 6
2.3 Entanglement . . . 7
2.4 Von Neumann's protocol . . . 9
2.4.1 Extracting Randomness . . . 9
2.4.2 Entanglement Concentration . . . 11
2.5 Elias's Protocol . . . 13
2.5.1 Extracting Randomness . . . 13
2.5.2 Entanglement Concentration . . . 15

3 Methods 21
3.1 Von Neumann's Protocol . . . 21
3.1.1 Simulation . . . 21
3.1.2 Executing on a Quantum Computer . . . 24
3.2 Elias's Protocol . . . 27
3.2.1 Measuring the Hamming Weight . . . 27
3.2.2 Executing with Two Initial Pairs . . . 30

4 Results 33
4.1 Von Neumann's Protocol . . . 33
4.1.1 One run . . . 33
4.1.2 Iterating . . . 35
4.1.3 Executing on a Quantum Computer . . . 36
4.2 Elias's Protocol . . . 39

5 Discussion 43

6 Conclusion 45

References 46
References . . . 46

A Appendix 47

Abstract

Many applications of quantum communication require a high amount of entanglement between the states of two separated parties, for example quantum teleportation or superdense coding. However, when states are locally entangled and then separated, errors arise during transportation of the qubits. This results in a lower amount of entanglement between the states. Thus, we need protocols that apply local operations on the qubits of the two separate parties to achieve higher entanglement. This is called entanglement concentration. We analyze two protocols that classically can be used for the extraction of randomness and apply it to the concentration of entanglement: Von Neumann's protocol and Elias's protocol. These protocols are well-described in the literature. We analyze the performance of the protocols on quantum computers with varying error rates.

We find that Von Neumann's protocol extracts entanglement similar to the theory on the best 5-qubit processor (Athens): there is an average percentage change in the concurrence when the protocol succeeds of 49.0%, which is close to the theoretical value of 52.5%. Elias's protocol requires many more qubits than Von Neumann's protocol. The only quantum computer large enough to run the protocol (Melbourne) has high error rates. Also, Elias's protocol requires more operations. This results in higher errors when we execute the protocol on Melbourne. We execute the protocol with 2 initial states and find an average percentage decrease of the concurrence when the protocol succeeds of 72.6%. Thus, we find that Elias's protocol is not suited to use in practice and the Von Neumann protocol can be used to extract entanglement with the best quantum processors used in this work.

1

Introduction

Quantum physics is well-known for its spooky algorithms that can do things we classically deem hard or even impossible, like quantum teleportation or superdense coding. An underlying condition for these algorithms is strongly entangled states shared by two parties that are separated from each other. Entangled states are created by applying operations to both qubits involved, so they need to be together. The algorithms mentioned before require that the states are spatially separated. Therefore after the states are entangled, the qubits are transported via quantum communication channels. However, this transportation induces a lot of noise to the states, which end up not maximally entangled. Since the algorithms require a high amount of entanglement, we need protocols that apply local operations to extract entanglement. This is called entanglement concentration.

We take a closer look at two protocols for entanglement concentration: Von Neumann's protocol and Elias's protocol. These protocols are described extensively in the literature, for example by Blume-Kohout et all [1] and Kaye and Mosca [2]. However, current quantum computers have limitations in size (number of qubits) and induce large errors. In this work we test these protocols on quantum computers to see how well they perform. Thus the main question that we want to answer is:

How do the Von Neumann protocol and Elias's protocol perform in terms of entanglement extraction on real quantum devices, in comparison to the theory?

We start by explaining the necessary fundamental concepts of quantum information in Chapter 2, the Background Information. Here we also explain how the protocols work in theory. In Chapter 3, Methods, we discuss how these protocols are implemented on available simulators and quantum computers of IBM. In Chapter 4 we present the results of running the protocols on simulators and on quantum computers. We compare the results of executing on the perfect simulators and the real quantum devices. We will discuss and conclude this work in Chapters 5 and 6.

2

Background Information

In this chapter we first introduce a few fundamental concepts of quantum information. We discuss quantum states, single and multiple qubit gates and entanglement. This discussion is based on the book by Nielsen and Chuang [3]. We continue to discuss two protocols for the extraction of randomness and how these protocols can be applied to our problem of entanglement concentration.

2.1. Quantum States

2.1.1. Single Qubit

The quantum version of the classical bit is called a quantum bit or qubit. A qubit is a system that can be the Boolean states 0 and 1, though in quantum mechanics these states are represented as $|0\rangle$ and $|1\rangle$. The notation $|\rangle$ is called the Dirac notation. We can also represent these states in vector notation:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.1}$$

A difference between the classical bit and the qubit is that the latter can be in a superposition or linear combination of states. We consider a qubit in the general state $|\psi\rangle$.

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle \tag{2.2}$$

Here α_0 and α_1 are complex numbers. $|\psi\rangle$ is a valid quantum state as long as the state is normalized. When we measure the qubit in the computational basis, the state collapses to either $|0\rangle$ with a probability of $|\alpha_0|^2$ or to $|1\rangle$ with a probability of $|\alpha_1|^2$. Since the qubit must be in either of these two states, $|\psi\rangle$ is a valid quantum state only when the normalization condition is satisfied: $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

We can visualize single qubit state with the Bloch sphere. This requires yet another representation of the state $|\psi\rangle$:

$$|\psi\rangle = e^{i\delta}\left(\cos(\beta/2)\,|0\rangle + e^{i\phi}\sin(\beta/2)\,|1\rangle\right) \tag{2.3}$$

Here δ, β and ϕ are real numbers. We see that this representation also satisfies the normalization condition: $\cos^2(\beta/2)+\sin^2(\beta/2)=1$. We ignore the first exponent, since it has no observable effects. We see the Bloch sphere in Figure 2.1. Here, $\phi \in [0, 2\pi]$ is the azimuthal angle and $\beta \in [0, \pi]$ is the polar angle. In other works the polar angle is often called θ, but we don't follow this convention since we use θ for another angle (see Figure 3.2). We can only visualize the state of one qubit on the Bloch sphere. Every state that resides on the Bloch sphere is called a pure state. The interior points are called mixed states, which represent a probabilistic mixture of pure states.

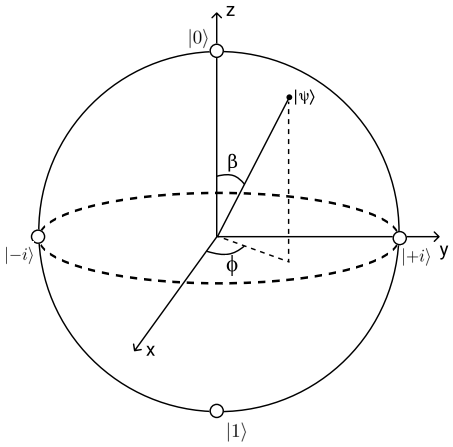


Figure 2.1: Bloch sphere

We see that the pure states on the z - axis are the computational basis states. Thus, the z - basis is commonly referred to as the standard basis. All other states on the Bloch sphere are superpositions. We see that the pure states on the y - axis are $|+i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$ and $|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle)$. For clarity we didn't show the pure state on the x - axis, but it is worth mentioning: on the positive axis the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ resides and on the negative axis $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

2.1.2. Multiple Qubits

When we have two qubits, they can be in each of the four computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$, or in a superposition of these states. Every computational basis state represents a tensor product of the two states inside the ket:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \tag{2.4}$$

In general, we can write the state vector $|\psi\rangle$ of a 2-qubit system as:

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle = \begin{bmatrix}\alpha_{00}\\\alpha_{01}\\\alpha_{10}\\\alpha_{11}\end{bmatrix} \tag{2.5}$$

The coefficients $\alpha \in \{0,1\}^2$ are complex and satisfy the normalization condition.

2.2. Quantum Gates

2.2.1. Single qubit gates

If we want to change the state of a qubit to another state, we can do so by applying a gate to the qubit. All quantum gates can be expressed in matrices, with the only constraint that they need to be unitary. A unitary matrix U satisfies:

$$UU^{\dagger} = I \tag{2.6}$$

Here $U^{\dagger}$ is the transpose and complex conjugate of U. All 1-qubit gates can be visualized as a rotation on the Bloch sphere. The Pauli gates allow us to rotate 180 degrees around an axis:

$$X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}\ ,\ Y = \begin{bmatrix}0 & -i\\i & 0\end{bmatrix}\ ,\ Z = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix} \tag{2.7}$$

Note that the Pauli X-gate acts similarly on the computational basis states as the classical NOT gate that flips a bit. The Pauli Z-gate can also be seen as a phase flip since it adds $180°$ to the phase ϕ. The Pauli gates are special cases of the rotation operators, which allow you to rotate about the x, y or z - axis with an angle θ. We will come across the R_y - gate later on:

$$R_y(\theta) = \begin{bmatrix}\cos(\theta/2) & -\sin(\theta/2)\\\sin(\theta/2) & \cos(\theta/2)\end{bmatrix} \tag{2.8}$$

Another important gate is the Hadamard gate. This gate represents a rotation around the $\frac{1}{\sqrt{2}}(\hat{x} + \hat{z})$ axis. A useful property is that it transforms the state $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$, as we see below.

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = |+\rangle$$
$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} = |-\rangle \tag{2.9}$$

2.2.2. Multiple qubit gates

We discuss three relevant quantum logic gates that act on multiple qubits: the CNOT gate, the SWAP gate and the Toffoli gate. The CNOT gate (controlled NOT gate) acts on two qubits. One of the qubits is the control qubit q_0 and the other the target qubit q_1. The gate stores the XOR $q_0 \oplus q_1$ in the target qubit. In other words, the NOT operation on the target qubit is only executed when the control qubit is $|1\rangle$. We find the circuit representation and the corresponding matrix in the basis $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ in Figures 2.2 and 2.3.

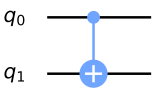


Figure 2.2: Circuit representation of the CNOT gate

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

Figure 2.3: Matrix of the CNOT gate

Then, two qubits can be swapped by the SWAP gate. The circuit representation of this gate and the corresponding matrix are shown in Figures 2.4 and 2.5.

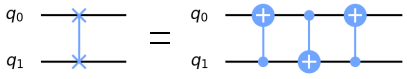


Figure 2.4: Circuit representation of the SWAP gate

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

Figure 2.5: Matrix of the SWAP gate

We see that we can represent the SWAP gate by applying 3 CNOT gates by writing the states as $|q_0\rangle = \begin{bmatrix}a\\b\end{bmatrix}$ and $|q_1\rangle = \begin{bmatrix}c\\d\end{bmatrix}$. Their joint state is given by:

$$|q_0 q_1\rangle = \begin{bmatrix}a\\b\end{bmatrix} \otimes \begin{bmatrix}c\\d\end{bmatrix} = ac\,|00\rangle + ad\,|01\rangle + bc\,|10\rangle + bd\,|11\rangle \tag{2.10}$$

We consider the joint state after each CNOT gate:

$$\begin{aligned}
&\text{CNOT 1: } ac\,|00\rangle + ad\,|11\rangle + bc\,|10\rangle + bd\,|01\rangle \\
&\text{CNOT 2: } ac\,|00\rangle + ad\,|10\rangle + bc\,|11\rangle + bd\,|01\rangle \\
&\text{CNOT 3: } ac\,|00\rangle + ad\,|10\rangle + bc\,|01\rangle + bd\,|11\rangle = \begin{bmatrix} c \\ d \end{bmatrix} \otimes \begin{bmatrix} a \\ b \end{bmatrix}
\end{aligned} \tag{2.11}$$

Thus, we see that the states are indeed swapped after the last CNOT gate.

The Toffoli gate is an extension of the CNOT gate and is called the CCNOT gate (controlled-controlled NOT gate). It needs three input qubits: two control qubits, q_0 and q_1, and one target qubit q_2. The Toffoli gate stores $q_2 \oplus q_0 q_1$ in the target qubit. We find the circuit representation and corresponding matrix in Figures 2.6 and 2.7.

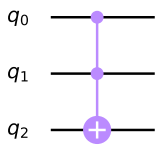


Figure 2.6: Circuit representation of the Toffoli gate

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

Figure 2.7: Matrix of the Toffoli gate

An important property of gates is that we can construct any unitary operation on any arbitrary number of qubits by using a finite set of qubit gates. We say that this set of gates is universal. There are multiple universal sets, one example are all single qubit gates and the CNOT gate.

2.3. Entanglement

A physicist who played a key role in our understanding of quantum entanglement today, was John Stewart Bell. He demonstrated, in response to the paper of Einstein, Podolsky and Rosen [4], that quantum theory predicts violations of the classical limit of correlations [5]. Nowadays, quantum entanglement is essential for many quantum algorithms, such as superdense coding or quantum teleportation. Two or more qubits are said to be entangled when the state of each individual qubit cannot be described independently of the quantum state of the other qubits. In this work we focus on the entanglement of two qubits. Let's assume Alice has one qubit in the state $|\psi\rangle$ and Bob has another qubit in the state $|\phi\rangle$. We can write their joint state $|\Psi\rangle$ as the tensor product of the individual qubit states, as shown below:

$$|\Psi\rangle = |\psi\rangle \otimes |\phi\rangle \tag{2.12}$$

If the joint state can be written like eq. 2.12, we say that the joint state is separable.

However, the states of Alice and Bob can also be correlated and then the joint state is said to be entangled. Four states that are maximally entangled are the Bell states or EPR pairs.

$$
\begin{aligned}
|\Phi^+\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle_A|0\rangle_B + |1\rangle_A|1_B\rangle\right) \\
|\Phi^-\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle_A|0\rangle_B - |1\rangle_A|1_B\rangle\right) \\
|\Psi^+\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle_A|1\rangle_B + |0\rangle_A|1_B\rangle\right) \\
|\Psi^-\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle_A|1\rangle_B - |0\rangle_A|1_B\rangle\right)
\end{aligned}
\tag{2.13}
$$

We see that whenever Alice's qubit is measured and collapsed to state $|0\rangle$ or $|1\rangle$, the state of Bob is fully determined too. In other words, the measurement outcomes are perfectly correlated. We use a quantity called the concurrence C to quantify the amount of entanglement. For the general state of eq. 2.5, the concurrence looks like [6]:

$$C(|\psi\rangle) = 2|\alpha_{00}\alpha_{11} - \alpha_{01}\alpha_{10}| \tag{2.14}$$

The states of two qubits are entangled if the concurrence is nonzero and maximally entangled when $C = 1$. We see that the Bell states indeed are maximally entangled since $2 \cdot |\frac{1}{2}| = 1$. For a concurrence between 0 and 1, the state is said to be entangled (not maximally entangled).

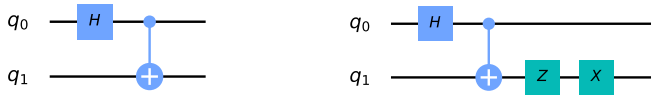
We can create the first Bell state from eq. 2.13 with the circuit in Figure 2.8a. The qubits in this circuit start in the state $|0\rangle$. After the Hadamard gate, the joint states of the qubits is:

$$\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \tag{2.15}$$

After the CNOT gate, the qubits are in the Bell state $|\Phi^+\rangle$:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.16}$$

By applying a Pauli-X and / or Pauli-Z gate to q_1 afterwards, we can initialize the other Bell states. For example, we see the circuit to create $|\Psi^-\rangle$ in Figure 2.8b.



(a) Circuit to create $|\Phi^+\rangle$ (b) Circuit to create $|\Psi^-\rangle$

Figure 2.8

2.4. Von Neumann's protocol

In computer science, a lot of research has been conducted to randomness. Randomness plays an important role in many algorithms, for example in cryptography to choose random secret keys [7]. We consider two randomness extractors: Von Neumann's protocol and Elias's protocol. A randomness extractor is a protocol that extracts a sequence of perfect independent random bits from a weak random source, which is a source of a sequence of bits that may contain biases or correlations. In the case of Von Neumann's and Elias's protocol the input is a stream of biased, independent bits. We will see that these protocols can also be used to generate perfect EPR pairs in theory.

2.4.1. Extracting Randomness

We start by considering a classical source of biased and independent binary information. In a sequence of bits, 'biased' means that the distribution of zeroes and ones is not uniform. When bits are independent, there is no correlation between two consecutive bits. We sample two bits at a time and call the probability of drawing a zero p_0 and the probability of one p_1.

We see that the probability of the sequence '01' is the same as the probability of the sequence '10': $p_0 p_1$. Von Neumann's protocol uses this property. If we draw a pair with odd parity the first bit is reported. If however '00' or '11' is drawn, we discard the bits and draw again. We find an overview of the protocol in Figure 2.9. We see that there is no bias in the output bits anymore: the output sequence is independent and uncorrelated.

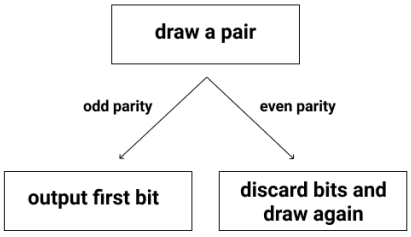


Figure 2.9: An overview of the Von Neumann protocol

The probability of succeeding once in $\frac{n}{2}$ executions of the protocol is a geometric distribution. Here n is the number of input bits. Since we need two input bits for one execution of the protocol, this is used in the exponent.

$$Pr(n) = 2p_0p_1(1-2p_0p_1)^{\frac{n}{2}-1} \tag{2.17}$$

We can now calculate the expected waiting time: the expected amount of input bits for one random output bit.

$$\begin{aligned}\langle n\rangle &= 2\cdot\sum_{n=0}^{\infty}\frac{n}{2}\cdot Pr(n) = \sum_{n=0}^{\infty} n\cdot 2p_0p_1(1-2p_0p_1)^{\frac{n}{2}-1}\\ &= \sum_{n=0}^{\infty}\frac{n}{2}\cdot 4p_0p_1(1-2p_0p_1)^{\frac{n}{2}-1}\end{aligned} \tag{2.18}$$

We substitute the following geometric series.

$$\begin{aligned}\sum_{n=1}^{\infty} nx^{n-1} &= \frac{1}{(1-x)^2}\\ \rightarrow \langle n\rangle &= \frac{4p_0p_1}{4p_0^2p_1^2} = \frac{1}{p_0p_1}\end{aligned} \tag{2.19}$$

We write the probability of getting one random bit per input bit as the inverse of Eq. 2.19 amount of random bits generated per input bit as R:

$$R = \frac{dN_{rbits}}{dn} = p_0p_1 \tag{2.20}$$

In classical information theory the Shannon entropy is often used to quantify the information stored in a variable X [3]. The Shannon entropy is defined as:

$$H(X) = H(p_1,...,p_n) = -\sum_{x=1}^{n} p_x\log_2 p_x \tag{2.21}$$

Now we calculate the Shannon entropy of the input bits n and output bits N and see how much information is lost. When the protocol succeeds, the probability of '0' and '1' is the same: 0.5. We can substitute these values to calculate the Shannon entropy of the output bits. However, the probability of getting one random bit per input bit is p_0p_1, as we saw in Equation 2.20. We need to multiply with p_0p_1, as we see below.

$$\begin{aligned}H(n) &= H(p_0,p_1) = -(p_0\log_2(p_0) + p_1\log_2(p_1))\\ H(N) &= H(p_0) = -p_0p_1\left(0.5\cdot\log_2(0.5) + 0.5\cdot\log_2(0.5)\right) = p_0p_1\end{aligned} \tag{2.22}$$

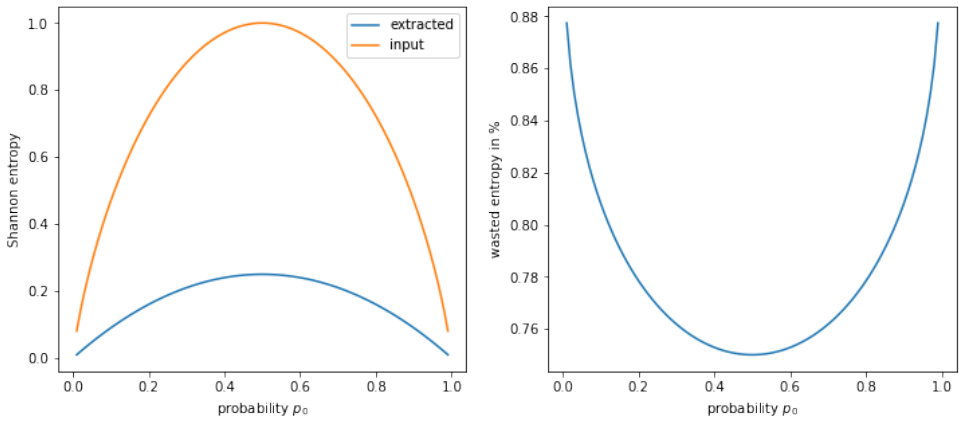


Figure 2.10: Left: the entropy of the input bit and output bit vs p_0. Right: the percentage wasted entropy.

In Figure 2.10 we find a plot of Eq. 2.22 for $0 < p_0 < 1$ and $p_1 = 1 - p_0$. We see that the Shannon entropy of the input bit is always larger than the entropy we extract: at minimum, the protocol wastes 75% of the initial entropy.

2.4.2. Entanglement Concentration

We apply a similar protocol for the concentration of entanglement. We assume that Alice and Bob share multiple copies of the entangled state:

$$|\psi\rangle = \sqrt{p_0}\,|0_A 0_B\rangle + \sqrt{p_1}\,|1_A 1_B\rangle \tag{2.23}$$

We use eq. 2.14 to see that the concurrence of this state is $C(|\psi\rangle) = 2\sqrt{p_0 p_1}$. Alice and Bob now each run the following algorithm:

1. Draw two qubits q_1 and q_2 from the input.
2. Perform a CNOT operation with q_1 as the control qubit and q_2 as the target qubit.
3. If the second qubits of both Alice and Bob are equal to '1', they both swap q_1 with the output register and halt. Otherwise they repeat the protocol.

Suppose we run the algorithm with two copies of the initial state (so the protocol is executed once). We look at the states after each part of the algorithm, starting with the initial state:

1: $|\psi\rangle^{\otimes 2} = p_0\,|00_A\rangle\,|00_B\rangle + p_1\,|11_A\rangle\,|11_B\rangle + \sqrt{p_0 p_1}\big(\,|01_A\rangle\,|01_B\rangle + |10_A\rangle\,|10_B\rangle\,\big)$

Now we store the parity $q_1 \oplus q_2$ in q_2 and reorder the qubits.

$$\textbf{2:}\ |\psi\rangle^{\otimes 2} = (p_0\,|0_A0_B\rangle + p_1\,|1_A1_B\rangle)\,|0_A0_B\rangle + \sqrt{2p_0p_1}\Big(\frac{1}{\sqrt{2}}\,|0_A0_B\rangle + \frac{1}{\sqrt{2}}\,|1_A1_B\rangle\Big)\,|1_A1_B\rangle \tag{2.24}$$

If the second qubits are measured, the protocol can either fail or succeed. The joint states of Alice and Bob for each case are:

$$\begin{aligned}\textbf{3:}\ |\psi\rangle_{fail} &= \frac{p_0}{\sqrt{1-2p_0p_1}}\,|0_A0_B\rangle + \frac{p_1}{\sqrt{1-2p_0p_1}}\,|1_A1_B\rangle \\ |\psi\rangle_{suc} &= \frac{1}{\sqrt{2}}(|0_A0_B\rangle + |1_A1_B\rangle)\end{aligned} \tag{2.25}$$

When the protocol succeeds the concurrence of the state is $C_{suc} = 1$. We see that the probability of retrieving a maximally entangled state $|\Phi^+\rangle$ is given by $2p_0p_1$. The probability distribution of the number of iterations $\frac{n}{2}$ required to get one EPR pair is a geometric distribution. Here, n is the number of input entangled states.

$$Pr(N) = 2p_0p_1(1-2p_0p_1)^{\frac{n}{2}-1} \tag{2.26}$$

In the same way as with the classical case, we see that the expected number of input states is $\langle n\rangle = \frac{1}{p_0p_1}$. The rate at which we extract maximally entangled states from the input states is the inverse, p_0p_1. Thus, we calculate the final concurrence via:

$$C_{final} = p_0p_1 \cdot C_{suc} = p_0p_1 \tag{2.27}$$

Here, we don't take into account the concurrence of the state when the protocol fails. In that case, we throw away the state so the concurrence is zero.

When we start with a maximally entangled state, $p_0 = p_1 = \frac{1}{2}$ and the concurrence of the initial state is 1. Now, the least amount of entropy is wasted: $C_{final} = p_0p_1 = 0.25$. This agrees with the classical analogue, since at least 75% of the total entanglement is wasted.

On-demand mode

By running the protocol multiple times, we can increase the probability of retrieving one EPR pair. This is called the on-demand mode. If we run the algorithm on $2N$ copies of $|\psi\rangle$ we get after part 2 of the algorithm:

$$\begin{aligned}|\psi\psi\rangle^{\otimes N} \rightarrow \sqrt{2p_0p_1}\,|\Phi^+\rangle \Bigg(&\sum_{k=0}^{N-1}(1-2p_0p_1)^{k/2}(|0_A0_B\rangle\,|\psi\rangle_{fail})^{\otimes k}\,|1_A1_B\rangle\,|\psi\psi\rangle^{\otimes(N-k-1)} \\ &+ (1-2p_0p_1)^{N/2}(|\psi\rangle_{fail}\,|0_A0_B\rangle)^{\otimes N}\Bigg)\end{aligned} \tag{2.28}$$

For $N = 2$, this yields:

$$|\psi\psi\rangle^{\otimes 2} \rightarrow \underbrace{\sqrt{2p_0p_1}\,|\Phi^+\rangle\,|1_A1_B\rangle\,|\psi\psi\rangle}_{1} + \underbrace{\sqrt{2p_0p_1}\,|\Phi^+\rangle\sqrt{1-2p_0p_1}\,|0_A0_B\rangle\,|\psi\rangle_{fail}\,|1_A1_B\rangle}_{2}$$
$$+ \underbrace{(1-2p_0p_1)(|\psi\rangle_{fail}\,|0_A0_B\rangle)^{\otimes 2}}_{3} \tag{2.29}$$

Part 1 describes the possibility to get an EPR pair on the first try. Since the algorithm halts after succeeding, we have two initial states $|\psi\psi\rangle$ that are not changed. Part 2 describes the possibility to fail the first time, but succeed the second time. The last part, 3, gives the probability to fail both times. This same principle applies to eq. 2.28: every kth term of the summation represents the possibility to fail k times and to succeed once and retrieve an EPR pair. Thus, we see that we have $N-k-1$ initial states $|\psi\psi\rangle$. The last term tells us when the protocol fails for all copies.

Streaming mode

In the streaming mode, we don't halt in part 3 of the algorithm. We execute the protocol N times and every time the protocol succeeds, Alice and Bob place their entangled qubit in an output register. If there already are entangled qubits saved in the output register, we shift all qubits by one qubit. From eq. 2.24 we know that the probability of success of one run is $p = 2p_0p_1$. Since the iterations are independent, the number of successes j follows a binomial distribution with the following probability distribution:

$$Pr(j) = \binom{N}{j} p^j (1-p)^{N-j} \tag{2.30}$$

2.5. Elias's Protocol

2.5.1. Extracting Randomness

The second protocol that we consider for randomness extraction is Elias's protocol [8]. Similarly to Von Neumann's protocol, we assume the input to be a sequence of biased, independent bits. Again, the probability of '0' is p_0 and the probability of '1' is p_1. From this sequence we draw a string of N bits. Every string of length N with T '1' bits (thus $N-T$ '0' bits) has $\binom{N}{T}$ permutations. Because the input bits are independent, all permutations have the same probability:

$$Pr(N,T) = p_0^{N-T} p_1^T \tag{2.31}$$

We assign to all possible permutations of a $\binom{N}{T}$ string an index $\alpha \in [0, 1, \ldots, \binom{N}{T}-1]$. We see that if $\binom{N}{T} = 2^L$, the index can be written in binary and we immediately have L random bits.

For example, if we draw a string of length 4 that contains three '1' bits, there are $\binom{4}{3} = 4 = 2^2$ permutations. According to Elias's protocol, we should have $L = 2$ random bits. We see this by writing out all possible permutations and assigning an index that we write in binary.

$$\begin{aligned} 0111 &\rightarrow 0 \rightarrow 00 \\ 1011 &\rightarrow 1 \rightarrow 01 \\ 1101 &\rightarrow 2 \rightarrow 10 \\ 1110 &\rightarrow 3 \rightarrow 11 \end{aligned} \tag{2.32}$$

Since these permutations all occur with the same probability, the binary index of the particular string that we draw immediately yields two random bits.

When $\binom{N}{T} \neq 2^L$ we write the binomial coefficient in its binary representation:

$$\binom{N}{T} = 2^{L_1} + 2^{L_2} + \ldots + 2^{L_n} \tag{2.33}$$

Here $L_n = \lfloor \log \binom{N}{T} \rfloor$. Now the indices are divided into bins α:

$$\begin{aligned} \alpha_{L_1} &= [0, 1, \ldots, 2^{L_1} - 1] \\ \alpha_{L_2} &= [2^{L_1}, 2^{L_1} + 1, \ldots, 2^{L_1} + 2^{L_2} - 1] \\ &\ldots \\ \alpha_{L_k} &= \left[\sum_{j=1}^{k-1} 2^{L_j}, \ldots, \sum_{j=1}^{k-1} 2^{L_j} + 2^{L_k} - 1 \right] \\ &\ldots \end{aligned} \tag{2.34}$$

When we draw a particular string with an index α in bin α_{L_k} we output the binary sequence $\alpha - \sum_{j=1}^{k-1} 2^{L_j}$ and we have generated L_k random bits.

We illustrate this with another example. We draw a string of four bits from the input and see that there are two '1' bits. There are $\binom{4}{2} = 6 = 2^2 + 2^1$ possible strings with the same probability. According to eq. 2.33, $L_1 = 1$ and $L_2 = 2$. We assign indices to the permutations, analogous to what we did in the previous example. We also divide the indices into bins, according to eq. 2.34.

$$\begin{aligned} 0011 &\rightarrow 0 \\ 0101 &\rightarrow 1 \end{aligned} \Big\} \alpha_1$$

$$\begin{aligned} 0110 &\rightarrow 2 \\ 1001 &\rightarrow 3 \\ 1010 &\rightarrow 4 \\ 1100 &\rightarrow 5 \end{aligned} \Big\} \alpha_2 \tag{2.35}$$

When our particular string is in bin α_1, we output either 0 or 1 with equal probability: we have one random bit. When the string is in bin α_2 the output is $\alpha - 2^1$, so either $00, 01, 10$ or 11 with equal probability. In that case we have generated two random bits.

Blume-Kohout et al [1] show that the protocol extracts at least $NH(p) - \log_2(N + 1) - 2$ bits of entropy from N input bits. We see that when $N \to \infty$, the entropy approaches $NH(p)$. This is the theoretical upper bound, since we cannot extract more entropy than the entropy of the input bits.

2.5.2. Entanglement Concentration

We can use Elias's protocol as an entanglement concentration protocol. Let's assume that Alice and Bob share N copies of a partially entangled state $|\psi\rangle$. Reordering this state gives:

$$\begin{aligned}
|\psi\rangle^{\otimes N} &= \left(\sqrt{p_0}\,|0_A 0_B\rangle + \sqrt{p_1}\,|1_A 1_B\rangle\right)^{\otimes N} \\
&= \sqrt{p_0}^{\,N}\,|0_A 0_B\rangle^{\otimes N} + \sqrt{p_0}^{\,N-1}\sqrt{p_1}\left(\sum_{H(x)=1}|x_A x_B\rangle\right) \\
&+ \sqrt{p_0}^{\,N-2}\sqrt{p_1}^{\,2}\left(\sum_{H(x)=2}|x_A x_B\rangle\right) + \ldots \\
&+ \sqrt{p_0}\sqrt{p_1}^{\,N-1}\left(\sum_{H(x)=N-1}|x_A x_B\rangle\right) + \sqrt{p_1}^{\,N}\,|1_A 1_B\rangle^{\otimes N} \\
&= \sum_{T=0}^{N}\sqrt{p_0}^{\,N-T}\sqrt{p_1}^{\,T}\sum_{H(x)=T}|x_A x_B\rangle
\end{aligned} \tag{2.36}$$

Here x is a string of length N and $H(x)$ the Hamming weight of this string. We see that for a given Hamming weight T, the state is a uniform superposition of the $\binom{N}{T}$ possible permutations of x. Now Alice and Bob both measure the Hamming weight of their qubits. The normalized state after this measurement is:

$$|\psi\rangle_T = \frac{1}{\sqrt{\binom{N}{T}}}\sum_{H(x)=T}|x_A x_B\rangle \tag{2.37}$$

We now need to define a function f that assigns a binary index from 0 to $\binom{N}{T} - 1$ to each permutation in the following way:

$$
\begin{aligned}
&f(00\ldots00\underbrace{11\ldots11}_{T}) = 00\ldots00\\
&f(00\ldots10\underbrace{11\ldots11}_{T-1}) = 00\ldots01\\
&\cdot\\
&\cdot\\
&\cdot\\
&f(\underbrace{11\ldots11}_{T}00\ldots00) = \binom{N}{T} - 1 = m = \underbrace{00\ldots00}_{N-L}\underbrace{m}_{L}
\end{aligned}
\tag{2.38}
$$

Here, $L = \lfloor \log \binom{N}{T} \rfloor$ Similarly to the classical case, when $\binom{N}{T}$ is equal to 2^L, we have L EPR pairs when Alice and Bob both throw away their first $N-L$ qubits. Kaye and Mosca [2] describe a way to implement the function f. Their implementation requires a quantum binary subtractor, which we won't discuss in this work.

Example with 2 shared pairs

We consider the case for $N = 2$, so Alice and Bob start with the state:

$$
\begin{aligned}
|\psi\rangle^{\otimes 2} &= \left(\sqrt{p_0}\,|0_A 0_B\rangle + \sqrt{p_1}\,|1_A 1_B\rangle\right)^{\otimes 2}\\
&= p_0\,|00\rangle_A\,|00\rangle_B + \sqrt{p_0 p_1}\left(\sum_{T=1}|x_A x_B\rangle\right) + p_1\,|11\rangle_A\,|11\rangle_B\\
&= p_0\,|00\rangle_A\,|00\rangle_B + \sqrt{2p_0 p_1}\left(\frac{1}{\sqrt{2}}\,|01\rangle_A\,|01\rangle_B + \frac{1}{\sqrt{2}}\,|10\rangle_A\,|10\rangle_B\right) + p_1\,|11\rangle_A\,|11\rangle_B
\end{aligned}
\tag{2.39}
$$

In eq. 2.39 the string $x \in \{0,1\}^2$. Now Alice and Bob both measure the Hamming weight. There are three possible measurement outcomes: $T = 0$, $T = 2$ and $T = 1$. In the first two cases the binomial coefficient $\binom{2}{0} = \binom{2}{2} = 1 = 2^0$, giving 0 EPR pairs according to Elias's protocol. Thus, we consider the case where $T = 1$, because for the binomial coefficient we have $\binom{N}{T} = 2 = 2^1$. Measuring this Hamming weight should result in 1 EPR pair. The state after the measurement and Hamming weight 1 is:

$$
\begin{aligned}
|\psi\rangle_{H(x)=1} &= \frac{1}{\sqrt{2}}\left(|01\rangle_A\,|01\rangle_B + |10\rangle_A\,|10\rangle_B\right)\\
&\rightarrow \frac{1}{\sqrt{2}}\left(|1_A 1_B\rangle + |0_A 0_B\rangle\right)
\end{aligned}
\tag{2.40}
$$

Note that in this case, we don't need to index the permutations. We already have one EPR pair if Alice and Bob both throw away their first qubit. Since we use less operations than applying the function, this method is also less prone to errors. We can see the final state in the last line of eq. 2.40.

For completeness, we also follow Elias's protocol: we assign a lexicographic index to the permutations.

$$01 \rightarrow 00\ , 10 \rightarrow 01 \tag{2.41}$$

Now Alice and Bob share the following state:

$$\frac{1}{\sqrt{2}}\left(|00\rangle_A |00\rangle_B + |01\rangle_A |01\rangle_B\right) \tag{2.42}$$

When they both throw away their first qubit, they share a Bell state $|\Phi\rangle^+$. The probability of measuring a Hamming weight $T = 1$ is $2p_0p_1$, as we see in eq. 2.39. The final concurrence is the same as when we apply the Von Neumann protocol and succeed: $C_{final} = p_0p_1$.

For this specific case of two shared pairs of entangled states, the two protocols might appear to be similar but the approach is quite different: Von Neumann's protocol requires a CNOT gate while in Elias's protocol we measure the Hamming weight.

Dividing the indices in bins

However, when $\binom{N}{T}$ is not a power of 2 we need to divide the indices in bins, similar to eq. 2.34. We use the notation $|y\rangle$ for the index of a permutation (a permutation after the function f is applied). Now we rewrite the binomial coefficient in the same way as eq. 2.33 but with a slightly different notation: $\binom{N}{T} = \beta_n 2^n + \beta_{n-1}2^{n-1} + \ldots + \beta_0 2^0$. We divide the indices into bins α as follows:

2

$$\frac{1}{\sqrt{\binom{N}{T}}}\sum_{y=0}^{\binom{N}{T}-1}|y\rangle_A|y\rangle_B = \frac{1}{\sqrt{\binom{N}{T}}}\underbrace{\sum_{y=0}^{\beta_n\cdot 2^n-1}|y\rangle_A|y\rangle_B}_{\alpha_n} + \frac{1}{\sqrt{\binom{N}{T}}}\underbrace{\sum_{y=0}^{\beta_n\cdot 2^n+\beta_{n-1}\cdot 2^{n-1}-1}|y\rangle_A|y\rangle_B}_{\alpha_{n-1}} + \ldots + \frac{1}{\sqrt{\binom{N}{T}}}\underbrace{\sum_{y=0}^{\beta_n\cdot 2^n+\beta_{n-1}\cdot 2^{n-1}+\ldots+\beta_0\cdot 2^0-1}|y\rangle_A|y\rangle_B}_{\alpha_0} \tag{2.43}$$

To see why this binning works, we look at an example. Let's assume that $0 \leq \binom{N}{T} \leq 7$, so we can write $\binom{N}{T} = \beta_2 \cdot 2^2 + \beta_1 \cdot 2^1 + \beta_0 \cdot 2^0$. We use the binning from eq. 2.43:

$$\frac{1}{\sqrt{\binom{N}{T}}}\sum_{y=0}^{\binom{N}{T}-1}|y\rangle_A|y\rangle_B = \frac{1}{\sqrt{\binom{N}{T}}}\underbrace{\sum_{y=0}^{\beta_2\cdot 2^2-1}|y\rangle_A|y\rangle_B}_{\alpha_2} + \frac{1}{\sqrt{\binom{N}{T}}}\underbrace{\sum_{y=0}^{\beta_2\cdot 2^2+\beta_1\cdot 2^1-1}|y\rangle_A|y\rangle_B}_{\alpha_1} + \frac{1}{\sqrt{\binom{N}{T}}}\underbrace{\sum_{y=0}^{\beta_2\cdot 2^2+\beta_1\cdot 2^1+\beta_0\cdot 2^0-1}|y\rangle_A|y\rangle_B}_{\alpha_0} \tag{2.44}$$

If $\beta_2 = 1$, α_2 includes $|000\rangle, |001\rangle, |010\rangle, |011\rangle$, otherwise this bin is empty. If $\beta_1 = 1$, α_1 contains $\beta_2 00$ and $\beta_2 01$ and if $\beta_0 = 1$ the bin α_0 contains $\beta_2\beta_1 0$. In other words, every bin where $\beta_j = 1$ contains 2^j strings. Thus, we rewrite eq. 2.44 to:

$$\sum_{y=0}^{\binom{N}{T}-1} |y\rangle_A |y\rangle_B = \underbrace{\beta_2 \sum_{y=000}^{011} |y\rangle_A |y\rangle_B}_{\alpha_2} + \underbrace{\beta_1 \sum_{y=\beta_2 00}^{\beta_2 01} |y\rangle_A |y\rangle_B}_{\alpha_1} + \underbrace{\beta_0 \sum_{y=\beta_2\beta_1 0}^{\beta_2\beta_1 0} |y\rangle_A |y\rangle_B}_{\alpha_0} \tag{2.45}$$

Since we measure the Hamming weight T and the length of the string N is known, we know the values of β_j. Now, Alice and Bob both measure their leftmost qubit, y_2 (they will both obtain the same result). There are three possible cases.

1. $y_2 = 0$ and $\beta_2 = 1$. We know from eq. 2.45 that we are in bin α_2, so the state is an equal superposition of the following strings y:

$$\frac{1}{2}\left(|000\rangle_A |000\rangle_B + |001\rangle_A |001\rangle_B + |010\rangle_A |010\rangle_B + |011\rangle_A |011\rangle_B\right) \tag{2.46}$$

   If we throw away y_2, Alice and Bob are left with 2 EPR pairs:

$$\begin{aligned}
&\frac{1}{2}\left(|00\rangle_A |00\rangle_B + |01\rangle_A |01\rangle_B + |10\rangle_A |10\rangle_B + |11\rangle_A |11\rangle_B\right) \\
&= \frac{1}{2}\left(|0_A 0_B\rangle |0_A 0_B\rangle + |0_A 0_B\rangle |1_A 1_B\rangle + |1_A 1_B\rangle |0_A 0_B\rangle + |1_A 1_B\rangle |1_A 1_B\rangle\right) \\
&= \frac{1}{\sqrt{2}} |0_A 0_B\rangle \left(\frac{1}{\sqrt{2}}(|0_A 0_B\rangle + |1_A 1_B\rangle)\right) + \frac{1}{\sqrt{2}} |1_A 1_B\rangle \left(\frac{1}{\sqrt{2}}(|0_A 0_B\rangle + |1_A 1_B\rangle)\right) \\
&= \left(\frac{1}{\sqrt{2}} |0_A 0_B\rangle + \frac{1}{\sqrt{2}} |1_A 1_B\rangle\right) \cdot \left(\frac{1}{\sqrt{2}} |0_A 0_B\rangle + \frac{1}{\sqrt{2}} |1_A 1_B\rangle\right)
\end{aligned} \tag{2.47}$$

2. $y_2 = 0$ and $\beta_2 = 0$. We know that bin α_2 is empty. If $\beta_1 = 1$, α_1 contains $|000\rangle$ and $|001\rangle$ and if $\beta_0 = 1$, α_0 contains $|0\beta_1 0\rangle$. The only option for y_2 is zero. After normalizing, the state is:

$$\frac{1}{\sqrt{3}}\left(\beta_1 |000\rangle_A |000\rangle_B + \beta_1 |001\rangle_A |001\rangle_B + \beta_0 |0\beta_1 0\rangle_A |0\beta_1 0\rangle_B\right) \tag{2.48}$$

3. If $y_2 = 1$, β_2 has to be one, otherwise these permutations would not occur. We can be in either α_1, that now contains the states $|100\rangle$ and $|101\rangle$ if $\beta_1 = 1$ or in α_0 that contains $|1\beta_1 0\rangle$ if $\beta_0 = 1$. Thus, the normalized state is:

$$\frac{1}{\sqrt{3}}\left(\beta_1 |100\rangle_A |100\rangle_B + \beta_1 |101\rangle_A |101\rangle_B + \beta_0 |1\beta_1 0\rangle_A |1\beta_1 0\rangle_B\right) \tag{2.49}$$

After discarding y_2, the state in case 2 and 3 looks like:

$$\frac{1}{\sqrt{3}}\left(\beta_1 |00\rangle_A |00\rangle_B + \beta_1 |01\rangle_A |01\rangle_B + \beta_0 |\beta_1 0\rangle_A |\beta_1 0\rangle_B\right) \tag{2.50}$$

We don't know how many EPR pairs we have extracted, if any at all. We need to measure the leftmost qubit again, y_1. Now we need to consider the same three cases. In case 1, Alice and Bob are in the state

$$\frac{1}{\sqrt{2}}\left(|00\rangle_A |00\rangle_B + |01\rangle_A |01\rangle_B\right) \tag{2.51}$$

After Alice and Bob both throw away their first qubit, they share one EPR pair. In the second case, the only possible state is $|00\rangle$ and in the third case $|10\rangle$. These cases don't yield any EPR pairs.

We can generalize this to a simple algorithm that extracts EPR pairs from an indexed state with k qubits: Alice (or Bob) measures the leftmost qubit y_k. If $y_k = 0$ and $\beta_k = 1$, they discard the y_k and are left with $k-1$ EPR pairs. If this is not the case, they repeat the process l times, until the leftmost qubit is zero and the corresponding bit in the binary expression of $\binom{N}{T}$ is one. The number of EPR pairs extracted is $k-l$. Note that only Alice or Bob needs to check how many EPR pairs are extracted, since they can communicate classically.

One the available quantum computers, a classical if-statement is not possible. In this procedure we would need to measure qubits and conditionally on the outcome, perform other quantum operations. To avoid this, Kaye and Mosca [2] designed a quantum network in their paper, which we won't discuss since it is beyond the extent of this work.

3

Methods

We want to know how the protocols perform in practice. To this end, we implement the protocols using Qiskit[1] [9]. We describe the circuits of all experiments in this chapter.

3.1. Von Neumann's Protocol

Von Neumann's protocol is a sequential protocol, so it consecutively applies an algorithm to two entangled states. We start by explaining how we run the protocol once. Next, we discuss ways to repeat the protocol and what modifications the circuits need to run on quantum processors.

3.1.1. Simulation

The protocol starts with multiple copies of the entangled state of eq. 2.23. To apply the protocol once, we need four qubits: two for Alice and two for Bob. We create a quantum register of six qubits and a classical register of two bits. The necessity of the two extra qubits and two classical bits will become clear later. By default, the initial state of every qubit is $|0\rangle$. We apply a Hadamard gate to the qubits of Alice. Each qubit of Alice will be in the superposition state:

$$q_A = H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \tag{3.1}$$

The starting point of this protocol is an entangled state, so we have to rotate Alice's qubits over the y - axis through an angle θ:

$$\begin{aligned} q_A = \begin{bmatrix}\cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2})\\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2})\end{bmatrix}\begin{bmatrix}1/\sqrt{2}\\ 1/\sqrt{2}\end{bmatrix} &= \begin{bmatrix}\cos(\frac{\theta}{2})/\sqrt{2} - \sin(\frac{\theta}{2})/\sqrt{2}\\ \sin(\frac{\theta}{2})/\sqrt{2} + \cos(\frac{\theta}{2})/\sqrt{2}\end{bmatrix} \\ &= \sqrt{p_0}\,|0\rangle + \sqrt{p_1}\,|1\rangle \end{aligned} \tag{3.2}$$

[1] All code is available via https://github.com/nivbharos/thesis.

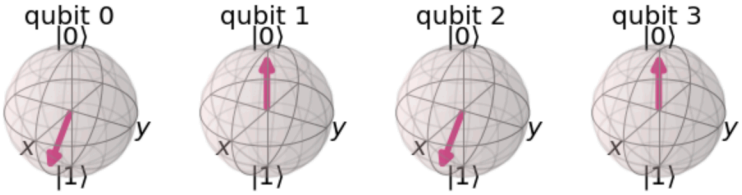


Figure 3.1: Qubits 0 and 2 are in the state: $R_y(\pi/4)H\,|0\rangle$, qubits 1 and 3 are in the default state $|0\rangle$

3

with

$$\sqrt{p_0} = \frac{\cos\theta/2 - \sin\theta/2}{\sqrt{2}}$$
$$\sqrt{p_1} = \frac{\cos\theta/2 + \sin\theta/2}{\sqrt{2}} \tag{3.3}$$

In Figure 3.1 we see an example of Alice's qubits (qubits 0 and 2) in the initial state with an angle $\theta = \frac{\pi}{4}$ rad. We have not performed any operations on Bob's qubits, 1 and 3, so these are in the default state $|0\rangle$.

We see that the initial state is uniquely defined by θ. We will often encounter θ in this work, so visualize it with a two-dimensional representation of the Bloch sphere in Figure 3.2.

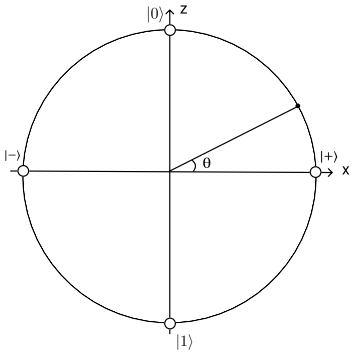


Figure 3.2: A two-dimensional representation of the Bloch sphere visualizing θ.

Now we can entangle the qubits of Alice and Bob by performing a CNOT operation from the qubits of Alice on the qubits of Bob, which stores $q_A \oplus q_B$ in q_B.

$$|\psi\rangle = \sqrt{p_0}\,|0_A 0_B\rangle + \sqrt{p_1}\,|1_A 1_B\rangle \tag{3.4}$$

This is the initial state and we execute Von Neumann's protocol as described in Section 2.4.2. We find the circuit in Figure 3.3. The operations before the barrier create the initial state 3.4. Here, $q0_0$ and $q0_2$ are the qubits of Alice, $q0_1$ and $q0_3$

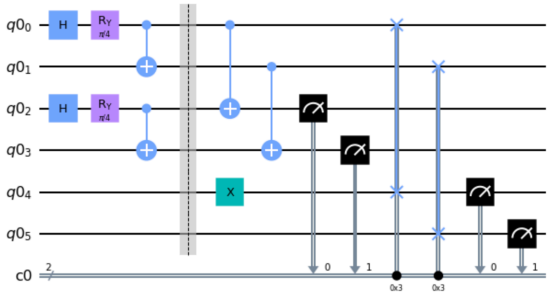


Figure 3.3: Circuit that executes the Von Neumann protocol once for $\theta = \frac{\pi}{4}$ rad.

the qubits of Bob, $q0_4$ and $q0_5$ the output register qubits. After the barrier, the protocol is applied and the second qubits of Alice and Bob are measured. For this measurement, we need the two extra classical bits to save the results. Then, the output register swaps with the first qubits if the measured states of the second qubits are '1'. Thus the two extra qubits are used to store the entanglement if the protocol succeeds.

If the second qubits are not equal to one, the protocol fails but the output register is still measured. This is why we apply an X-gate to one qubit of the output register: if the protocol fails, we will measure $|01\rangle$ in the output register. If we didn't apply the X-gate, we would measure the default state $|00\rangle$. Since this is also a state we measure when the protocol succeeds and the output register contains an EPR pair, we need the X-gate to see the difference between success and failure of the protocol in the results. At last, we measure the output register to check if it contains an entangled pair.

Iterating Von Neumann's protocol

Streaming mode

Another way of iterating is the fully streaming mode as described in Section 2.4.2. We find the circuit for three runs of the protocol in the appendix in Figure A.1. In this circuit, we create new initial states for every iteration and two extra qubits to save the entangled state in case of success. This implementation does require a lot of memory. By creating one quantum register that contains n qubits needed for the protocol, the classical computer we use to simulate this has to work with at most $2(2^n - 1)$ coefficients. To save a float we need four bytes, so we would need approximately 2^{n+3} bytes. In this implementation, we would need 6 qubits per iteration of the protocol (4 qubits to run the protocol and 2 qubits to save the entangled state), so it would be hard to execute many times. We use another

method to execute the protocol N times:

1. Create the circuit shown in Figure 3.3 for fixed θ
2. Ask the simulator to execute the circuit N times
3. Check how often the protocol succeeded and store this value in an array
4. Repeat this process as often as preferred

By repeating steps 1 to 3 we will be able to plot the distribution of the number of successes in a histogram. The memory necessary for this process is much lower: Repeating this process once only requires 5 qubits, so we need at most 2^8 bytes to store the coefficients. If we repeat the process n times, we would need at most $n \cdot 2^8$ bytes, which grows linearly. This is significantly more efficient than 2^{n+3}. We also apply this method to the on-demand mode.

On-demand mode

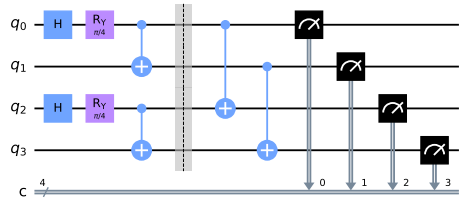
In practice, it's often useful to iterate the protocol to either retrieve one perfect EPR pair with higher probability. In Section 2.4.2 we have seen that the probability of producing at least one perfect EPR pair increases if the number of qubits increases and / or if $2p_0p_1$ increases. Also, we see that $2p_0p_1$ increases if θ approaches 0 rad. We use the method from the previous section but now we only check if the protocol has succeeded at least once in step 3 for varying θ and number of iterations N.

3.1.2. Executing on a Quantum Computer

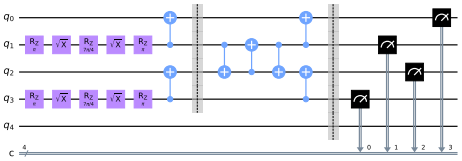
At the time of writing, there are four quantum computers publicly available on IBM Quantum Experience: three 5-qubit processors and one 15-qubit processor. We will execute the circuit as shown in Figure 3.4a on the following 5-qubit processors: Athens [10], Santiago [11] and Yorktown [12]. We need to adjust the circuit in order to run on these processors, so we use four qubits instead of six in Figure 3.3. Instead of swapping the entangled qubits with the output qubits, it suffices to measure all qubits and to check afterwards which runs have succeeded.

When we execute a circuit on a quantum processor, not all qubits can interact with each other. In other words, not all qubits are connected. When we need to apply operations on qubits that are not connected, we need SWAP operations to facilitate this. The connectivity of our circuit must be compatible with the connectivities of the quantum computers that we will use. In Figure 3.5 we see that the connectivity of the quantum computers is different from our circuit, hence we need to modify our circuit. When we execute a circuit, it gets automatically adapted to be able to run on the quantum computer, taking into account the connectivity and the available gates. Also, the three 5-qubit processors can only execute the following basis gates: CX, ID, RZ, SX and X.

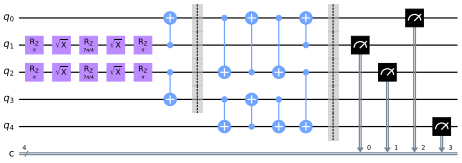
In Figure 3.4b we see the adapted circuit for Athens and Santiago. The order of the qubits is different: q_0 and q_2 are now the first and second qubits of Alice and q_1 and q_3 are now the first and second qubits of Bob, respectively. This minimizes



(a) Von Neumann circuit

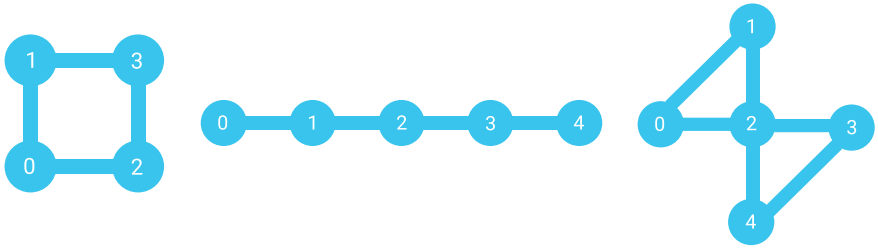


(b) Athens and Santiago



(c) Yorktown

Figure 3.4: Von Neumann circuit (a) and the adapted circuits for Athens and Santiago (b) and Yorktown (c) for $\theta = \frac{\pi}{4}$ rad



(a) Von Neumann circuit

(b) Athens and Santiago

(c) Yorktown

Figure 3.5: Topology diagrams of the Von Neumann circuit and the 5-qubit processors

	Athens			Santiago			Yorktown		
Error rates $[10^{-3}]$	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
CNOT	7.34	9.40	8.43	7.21	13.8	10.3	15.6	23.0	19.3
SX	0.221	0.288	0.259	0.210	0.605	0.338	0.458	1.28	0.820
Read out	8.30	21.8	15.5	10.0	40.5	19.5	25.9	88.8	46.5

Table 3.1: The minimum, maximum and average error rates of the 5-qubit processors Athens, Santiago and Yorktown in units of $[10^{-3}]$

the number of SWAP operations we need and thus minimize the error rates: if we would use the order of the original circuit, two SWAP operations would be necessary. From the topology diagram we see that before the third and fourth CNOT operations can be executed on Athens and Santiago, the compiler needs to swap qubit 2 and 3. One SWAP operation is implemented by using three CNOT gates. Also, the compiler uses equivalent RZ and SX gates to apply to Hadamard and R_y - gate.

For Yorktown the compiler uses all five qubits and at least two SWAP operations are needed due to different connectivities, as shown in 3.4c.

The 5-qubit processors have different error rates, which we find in Table 3.1. On average, Athens has the lowest error rates and Yorktown the highest.

Concurrence

We want to know if the qubits are more or less entangled after the Von Neumann protocol has succeeded and how much entanglement has been wasted. When we execute circuits on quantum computers, errors arise and there is a possibility that we end up with mixed states. In order to uniquely define the measured state, we need to apply quantum tomography. This is beyond the scope of this work, so we make the important assumption that after the protocol has been executed on

a quantum processor, the states that we measure are pure. This will simplify our calculations, though our results might deviate from the results when we do apply quantum tomography.

This assumption allows us to calculate the concurrence as described in Section 2.3. We create our initial state by using a Hadamard gate and a rotation around the y-axis, so the concurrence before we execute the protocol needs to be calculated after these gates have run on the appropriate simulator or quantum computer. Just using the theoretical value would be an overestimation of the initial entanglement because errors may arise. For completeness we can find the circuit that we use to calculate the initial concurrence in Figure 3.6.

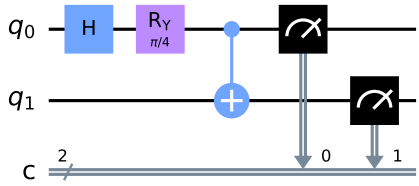


Figure 3.6: The circuit we will use to calculate the initial concurrence for different quantum processors with for example $\theta = \frac{\pi}{4}$ rad.

3.2. Elias's Protocol

As we saw in Section 2.5.2, Elias's protocol requires measuring the Hamming weight. We start by explaining the implementation with half adders. Next, we discuss how we implement Elias's protocol for two initial states on the simulator and on a quantum computer.

3.2.1. Measuring the Hamming Weight

We need to build a circuit that stores the Hamming weight of N qubits into $\log(N)$ ancilla qubits. To this end, we use half adders. A half adder adds two bits, a_0 and b_0 and returns the sum $a_0 \oplus b_0$ and carry $a_0 b_0$. The circuit is shown in Figure 3.7. Note that the RESET operation is usually implemented by measuring the qubit in the Z basis and applying an X gate if the measurement result is $|1\rangle$. Since this would cause qubit a_0 to collapse as well, we need another method to reset the 'zero' qubit. We use extra qubits that are with certainty in the state $|0\rangle$. Every time we need a RESET operation, we perform a SWAP operation from the 'zero' qubit to a qubit in the state $|0\rangle$. This preserves the input state of qubit a_0, which is also necessary to keep the half adder unitary.

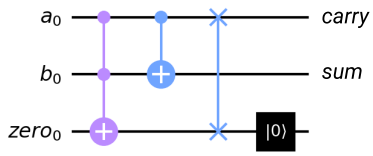


Figure 3.7: Implementation of the half-adder.

a_0	b_0	Carry	Sum
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

Table 3.2: Truth table of the half adder shown on the left.

The Toffoli gate stores the carry in the 'zero' qubit and the controlled NOT gate stores the sum in b_0. By performing a SWAP operation on a_0 and the 'zero' qubit, we can efficiently implement the Hamming weight since this allows us to reuse the input bit a_0 as the carry. After the SWAP, we reset the 'zero' qubit, which allows us to reuse this qubit as the 'zero' qubit again in the next half adder. Note that in Elias's protocol the two qubits we want to add also are the qubits that will store the concentrated entanglement. We can't simply SWAP these qubits with other qubits, because that would destroy the entanglement. Thus, we perform a CNOT operation from a_0 to one extra qubit. For completeness, we also show the truth table of the circuit of our half adder in Table 3.2.

Let's consider that we want to measure the Hamming weight of 15 qubits. We execute the following algorithm to add the value of each qubit n to the ancilla's:

1. Perform a CNOT operation from qubit n to a_0.

2. The half adder takes in a_0 and ancilla 1 and returns the sum in the same ancilla, which we call 'new ancilla 1'. a_0 now stores the carry and is the input of the next half adder on the left. Note that when we add the first qubit, ancilla 1 is always zero.

3. Repeat step 2 with the next ancilla as often as needed. The number of ancilla's needed to store the value of the n^{th} qubit is at most $\lceil \log_2(n+1) \rceil$.

4. Repeat steps 1 to 3 with the next qubit $n+1$.

The algorithm is visualized in Figure 3.8. Here the rows represent the algorithm from right to left and the algorithm is executed for all 15 qubits, which requires $\lceil \log_2(16) \rceil = 4$ ancilla qubits. Eventually, the ancilla qubits store the binary Hamming weight and we can read this out from left to right.

To see that this circuit indeed calculates the Hamming weight, we simulate the algorithm that returns the Hamming weight of three qubits. The circuit is shown in Figure 3.9.

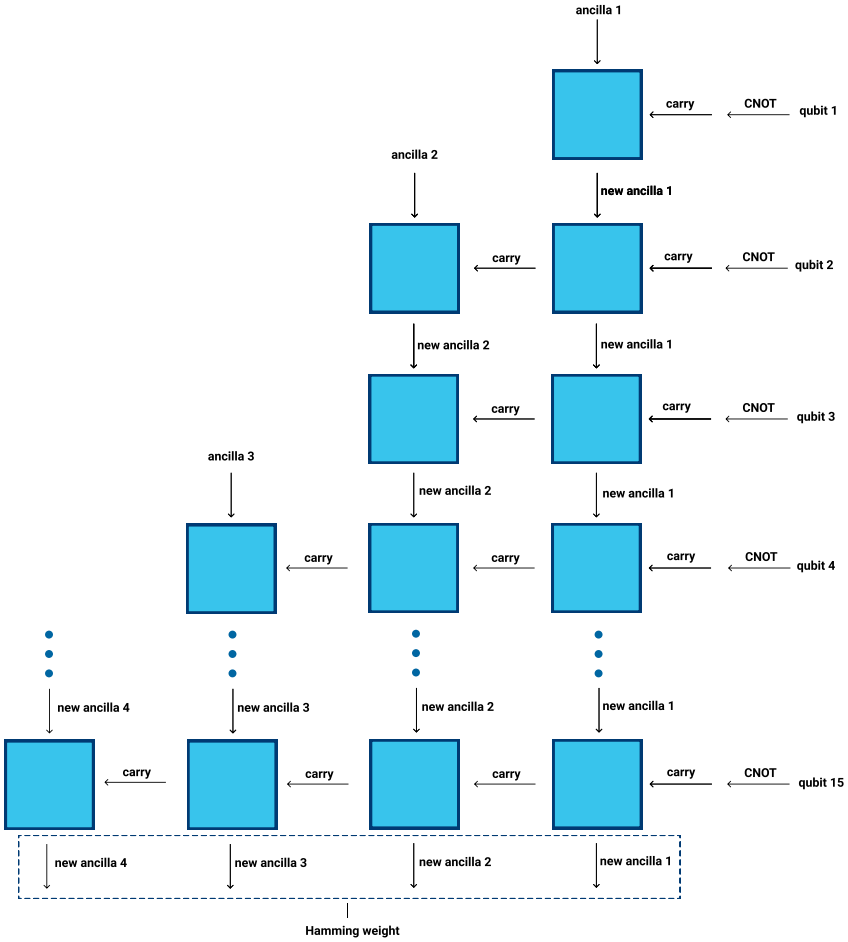


Figure 3.8: Circuit to calculate the Hamming weight of 15 qubits. Every blue square represents a half adder.

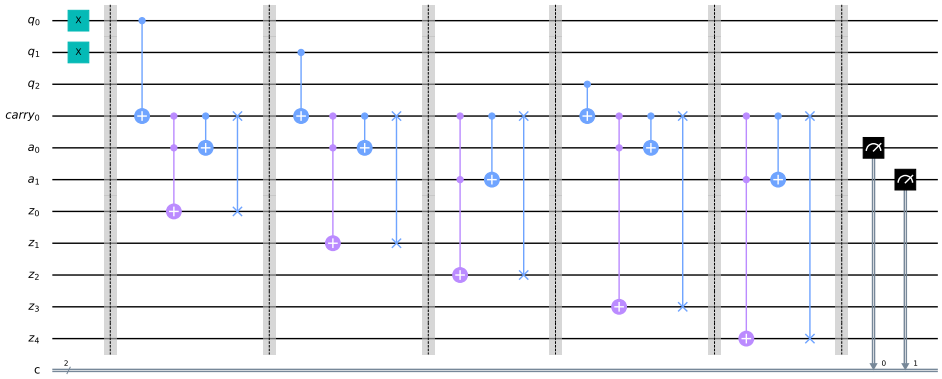


Figure 3.9: Circuit to measure the Hamming weight of three qubits.

Here, every part between two barriers represents a half adder. We apply 5 half adders, in accordance with the circuit in Figure 3.8 for 3 qubits. The first three qubits 'q' are the qubits of which we determine the Hamming weight. The fourth qubit is used as the 'carry', the qubits 'a' are the ancilla's that eventually store the Hamming weight and qubits 'z' are the zeroes that we use to perform the RESET operation. In the part of the circuit before the first barrier, we assign values to the qubits. By default the qubits are in the state $|0\rangle$ and by applying an Pauli-X gate (or Pauli-Y gate), we can flip the state to $|1\rangle$. In Figure 3.9 we apply a Pauli-X gate to the first qubit as an example, so the state of the qubits 'q' is $|\psi\rangle_q = |011\rangle$. When we measure the ancilla's that store the Hamming weight we indeed measure '10', which indeed corresponds to two qubits in the state $|1\rangle$.

3.2.2. Executing with Two Initial Pairs

We want to simulate Elias's protocol with two initial states, as described in Section 2.5.2. Executing with more than two initial states is not possible due to limitations on the amount of qubits we can use. For three initial states, we would need 21 qubits. This includes 6 qubit to create the initial states, 5 qubits to measure the Hamming weight (1 carry and 4 qubits to store the Hamming weight) and 10 qubits to apply the function according to Kaye and Mosca [2]. At the time of writing, there is no publicly available quantum processor of IBM with 21 qubits.

Thus, we only execute Elias's protocol with two initial states. Alice and Bob both measure the Hamming weight T as described in the section before. If they measure $T = 0$ or $T = 2$, they throw away both bits. When $T = 1$ however, they share one EPR pair when they throw away their leftmost qubit. Note that it is not necessary to apply the function f in this case, so we only measure the Hamming weight. We find the circuit that we execute on the simulator in Figure 3.10 below.

The qubits labeled 'q' are the partially entangled qubits that Alice and Bob start with (the first qubit is Alice's, the second qubit Bob's, and so on). In the part before

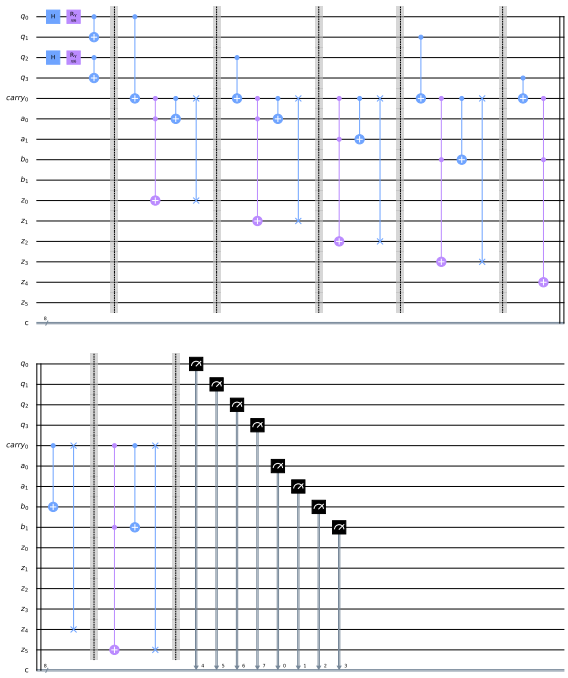


Figure 3.10: Elias's protocol for two initial partially entangled states with $\theta = \frac{\pi}{4}$ rad.

the first barrier we create the initial state (with $\theta = \pi/4$ rad). Then we store the Hamming weight of Alice in the qubits 'a' and Bob's in the qubits 'b'. After the last barrier, we measure the qubits q, a and b. When the protocol is applied in practice, only the qubits that store the Hamming weight are measured. We measure the qubits q as well, to check if the results of the Hamming weight agree with the number of ones. We also execute this circuit on a 16-qubit processor called Melbourne [13]. We use the same circuit. The error rates of Melbourne are shown in the table below. We see that these error rates are even higher than that of Yorktown (Table 3.1).

	Melbourne		
Error rates $[10^{-3}]$	Min	Max	Avg
CNOT	17.5	62.4	38.8
SX	0.4	2.9	1.4
Read out	24.7	81.2	47.0

Table 3.3: The minimum, maximum and average error rates of the 16-qubit processor Melbourne in units of 10^{-3}

4

Results

In this chapter we present the results of simulating the Von Neumann protocol and Elias's protocol and the results of executing experiments on the quantum computers.

4.1. Von Neumann's Protocol

We start with Von Neumann's protocol: first we present the results of simulating the protocol once and compare this to the theoretical values. After that we present the results of the on-demand and streaming mode. Finally, we execute the protocol on different quantum processors for varying initial states and compare the entanglement extraction expressed in the concurrence to the results of the simulator.

4.1.1. One run

We execute circuit 3.3 and measure the output register to check whether the results agree with our expectations. In the case where Alice and Bob both have two qubits, there are two possible outcomes: (1) if the second qubits of Alice and Bob are equal to one, we have seen in the previous chapter that the first qubits are perfectly entangled. The state we expect is $|\Phi^+\rangle$. Upon measurement, the state collapses to $|00\rangle$ or $|11\rangle$. (2) If the protocol fails, so the second qubits are not equal to one, the state will collapse to the default of the output register. We flipped the first qubit in the output register, so the default state collapses to $|10\rangle$.

We execute the circuit and measure the output register with 2^{15} shots for $\theta = \pi/4$ radians. The histogram of the measured states looks like:

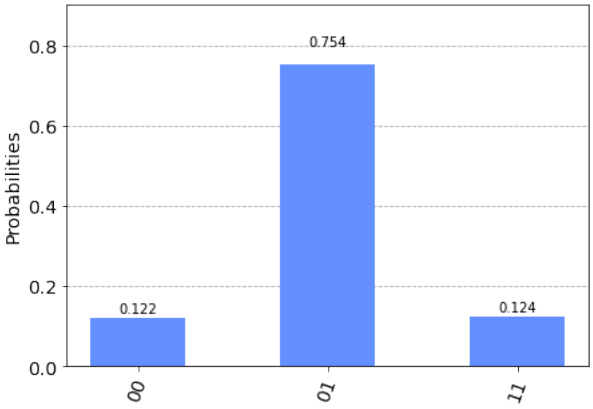


Figure 4.1: Histogram of states in the output register for $\theta = \pi/4$ radians and 2^{15} executions

From Section 3.1.1 we know that the bar of state $|01\rangle$ indicates that the protocol has failed. Thus, the protocol fails with a probability of 0.754. The bars of states $|00\rangle$ and $|11\rangle$ indicate when the protocol succeeds and they are approximately equal. In other words, whenever the protocol succeeds, we measure a state very close to the Bell state. We normalize the state after success $|\psi\rangle_{suc}$ and get:

$$|\psi\rangle_{suc} = \sqrt{0.496}\,|00\rangle + \sqrt{0.504}\,|11\rangle \tag{4.1}$$

The concurrence when the protocol succeeds is $C(|\psi\rangle_{suc}) = 2\sqrt{0.496 \cdot 0.504} \approx 1$, as expected. The final concurrence is $C(|\psi\rangle) = 0.123 \cdot 1 = 0.123$.

For $\theta = \pi/4$ radians we calculate the theoretical probability on success and failure by combining eq. 3.3 and the probability of success from eq. 2.28:

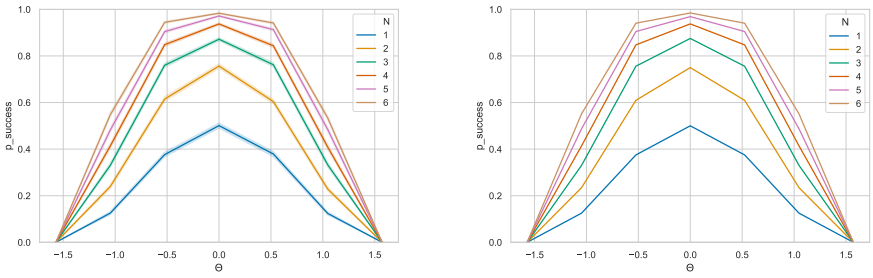
$$\begin{aligned} p_0 &= \left(\frac{\cos\pi/8 - \sin\pi/8}{\sqrt{2}}\right)^2 = 0.146 \\ p_1 &= \left(\frac{\cos\pi/8 + \sin\pi/8}{\sqrt{2}}\right)^2 = 0.854 \\ p_{suc} &= 2p_0p_1 = 0.250 \rightarrow p_{fail} = 0.750 \end{aligned} \tag{4.2}$$

If we compare this to Figure 4.1 we see that the theoretical values are in agreement with the experimental result. The theoretical final concurrence is $C_{final} = p_0p_1 \cdot 1 = 0.125 \cdot 1 = 0.125$, which is close to what we measured.

4.1.2. Iterating

On-demand mode

Now we simulate the on-demand mode, as described in 3.1.1. In the Figure 4.2 we plot the probability of retrieving at least one EPR pair for varying number of iterations N and varying θ.



(a) Simulated distributions

(b) Theoretical distributions

Figure 4.2: Probabilities of succeeding at least once with Von Neumann's protocol for varying N and θ.

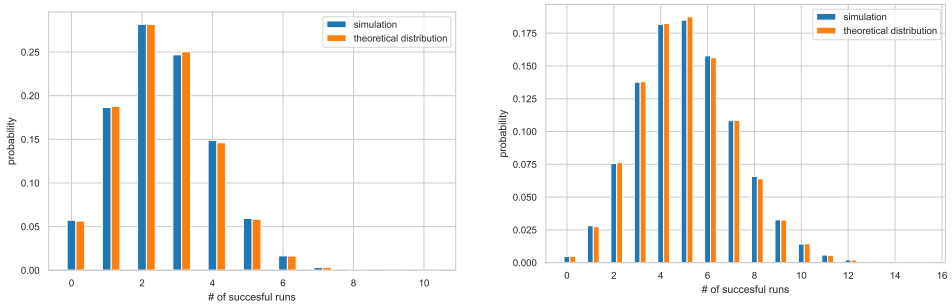
On the left we have plotted the simulated distributions with the 95% confidence intervals as shadows around the lines. We know from Section 3.1.1 that we can calculate the probability of success of one execution $p = 2p_0p_1$ from θ. We also saw in Section 2.4.2 that the theoretical distribution is a binomial distribution $\mathrm{Bin}(N, p)$. Now we calculate the probability of at least one success by subtracting the probability of zero successful runs from one:

$$Pr(j \geq 1) = 1 - N(1-p)^N \tag{4.3}$$

Here j is the number of successful runs. As we expected, the two graphs are very similar.

Streaming mode

Now we run the protocol in the streaming mode, as described in Section 3.1.1. We run the Von Neumann protocol 10 and 40 times for $\theta = \pi/4$ rad and we repeat both experiments 10^5 times. The results of the simulation are shown in the histograms in Figure 4.3. From Section 4.1.1 we know that the theoretical probability of success of one run is $p = 0.250$. Since the probability of success of multiple runs N is given by a binomial distribution, we plot the probability distribution of $\mathrm{Bin}(N, p)$ as well in the bar plot below. We see that the simulations are close to the theoretical distribution, so for more iterations the probability of no EPR pairs in the output register becomes smaller.



(a) 10 iterations

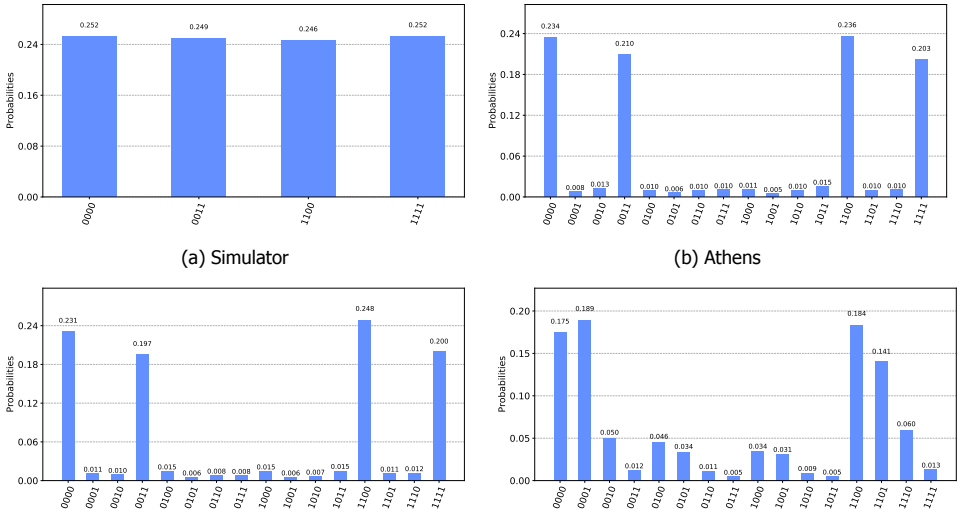
(b) 40 iterations

Figure 4.3: Barplot of the number of successes of Von Neumann's protocol with $\theta = \frac{\pi}{4}$ rad and the binomial distribution with $p = 0.250$

4

4.1.3. Executing on a Quantum Computer

To clearly visualize effects of the errors from Table 3.1 we first execute the circuits in Figure 3.4 on Athens, Santiago and Yorktown with a maximally entangled initial state, so $\theta = 0$ rad. As a comparison, the results of the simulation without errors are shown too in Figure 4.4. Here the two leftmost bits are the measurements of the second qubits of Alice and Bob (that should be $|11\rangle$ in the case of success) and the two other bits are the measurements of the first bits that are entangled in case of success.



(a) Simulator

(b) Athens

(c) Santiago

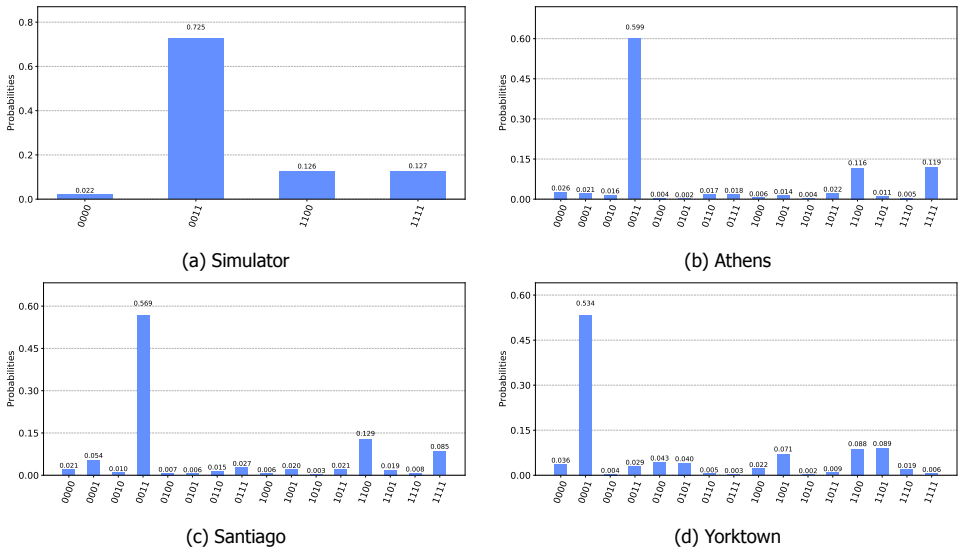
(d) Yorktown

Figure 4.4: Histograms of the measurement probabilities of one iteration of the Von Neumann protocol executed 8192 times with $\theta = 0$ rad on the simulator and different quantum processors.

According to Section 2.4.2 the probabilities of failure and success for one run

are $p_{\text{fail}} = \frac{1}{2} = p_{\text{suc}}$. We indeed see that the results of the simulator agree with our expectation. The bars of the states $|1100\rangle$ and $|1111\rangle$ together represent the probability to succeed: $0.246 + 0.252 = 0.498 \approx 0.5$. The results of the 5-qubit processor show errors, caused by the error rates of gates or by reading the qubits out. The errors of Athens and Santiago are very similar. However, the errors of Yorktown are so significant that it's hard to draw conclusions from these results.

Now we simulate the protocol with an entangled state. We execute circuit 3.4b 8192 times for $\theta = \frac{\pi}{4}$ rad. We find the results in the figure below. The probabilities of success (where the two leftmost bits are 1) of all 5-qubit processors differ from the simulators results. This is due to errors, though we don't have enough information to conclude to which degree every error contributes to these deviations.



(a) Simulator (b) Athens

(c) Santiago (d) Yorktown

Figure 4.5: Histograms of the measurement probabilities of one try of the Von Neumann protocol executed 8192 times with $\theta = \frac{\pi}{4}$ rad on the simulator and different quantum processors.

We now take a closer look at the results of Athens. From the data we can retrieve the probability of success for one try of the protocol: $p_{Athens} = 0.116 + 0.011 + 0.005 + 0.119 = 0.251$. This probability is very close to the theoretical $p = 0.250$ we retrieved in Section 4.1.1.

If the protocol succeeds, we read out all states from the histograms in Figure 4.5 with the two leftmost states equal to one. Due to errors, we also measure the states $|1101\rangle$ and $|1110\rangle$. Thus, when the protocol succeeds the state is not a perfect Bell state. We normalize the state $|\psi\rangle_{suc}$ we find for Athens:

$$
\begin{aligned}
|\psi\rangle_{success} =& \sqrt{0.119} \cdot A\,|00\rangle + \sqrt{0.116} \cdot A\,|11\rangle + \sqrt{0.011} \cdot A\,|01\rangle + \sqrt{0.005} \cdot A\,|10\rangle \\
& 0.116A^2 + 0.119A^2 + 0.011A^2 + 0,005A^2 = 1 \rightarrow A = 1.996 \\
& |\psi\rangle_{suc} = 0.69\,|00\rangle + 0.68\,|11\rangle + 0.14\,|01\rangle + 0.21\,|10\rangle
\end{aligned}
\tag{4.4}
$$

Where A is the normalization constant. We see that this state is not maximally entangled: the concurrence is $C[|\psi\rangle_{suc}] = 0.877 \neq 1$. We can also calculate the final concurrence, since the rate at which entanglement is extracted is $p_{Athens}/2$.

$$
C_{final} = \frac{1}{2} \cdot p_{Athens} \cdot C[|\psi\rangle_{suc}] = 0.123 \cdot 0.877 = 0.110 \tag{4.5}
$$

This is lower than the final concurrence in theory which we calculated in Section 4.1.1: 0.125. In the same way, we calculate the state when the protocol succeeds for Santiago and Yorktown and then calculate the concurrence when the protocol succeeds and the final concurrence. Now we can compare the entanglement before we executed the protocol to the entanglement when the protocol succeeds for each quantum processor. Also, we can compare how much entanglement is wasted.

We execute the circuit in Figure 3.6 for each quantum processor to calculate the initial concurrence. We calculate the concurrence of $|\psi\rangle_{suc}$ when the protocol has succeeded by executing the circuit in Figure 3.4b for different θ and one iteration. All circuits are executed with 8192 shots. We find the results in Table 4.1.

	$\theta = \frac{\pi}{3}$		$\theta = \frac{\pi}{4}$		$\theta = \frac{\pi}{6}$		
Concurrence	Initial	Success	Initial	Success	Initial	Success	Avg Change
Simulator	0.500	1	0.706	1	0.863	1	52.5%
Athens	0.451	0.859	0.616	0.877	0.799	0.911	**49.0**%
Santiago	0.517	0.816	0.738	0.856	0.851	0.900	26.5%
Yorktown	0.436	0.126	0.585	0.145	0.707	0.144	**-75.3**%

Table 4.1: The concurrence before the Von Neumann protocol is executed once (initial) and after the protocol has succeeded (success) for different initial states, together with the average percentage change.

The quantum computer that extracts entanglement best is Athens. Even when the initial concurrence is high, the protocol increases the concurrence. The quantum computer with the worst performance is Yorktown. Because Yorktown has a

different connectivity than Athens and Santiago, we saw in Figure 3.4 that Yorktown needs more SWAP operations to execute the circuit on Yorktown. Since more gates leads to more errors and since Yorktown has high error rates as we saw in Table 3.1, this leads to less entanglement than we started with for all θ.

We find the final concurrences for different θ in the following table, together with the final concurrence of the simulator and the average percentage change of the concurrence:

$$\text{Avg waste} = \frac{\text{final concurrence - initial concurrence}}{\text{initial concurrence}} \cdot 100\% \tag{4.6}$$

The final concurrence is always lower than the initial concurrence, so we call the average percentage change in concurrence the average waste.

	$\theta = \frac{\pi}{3}$		$\theta = \frac{\pi}{4}$		$\theta = \frac{\pi}{6}$		
Concurrence	Final	Wasted	Final	Wasted	Final	Wasted	Avg Waste
Simulator	0.063	87.4%	0.123	82.2%	0.187	78.4%	82.7%
Athens	0.051	88.7%	0.110	82.1%	0.151	81.1%	**84.0**%
Santiago	0.054	89.6%	0.103	86.0%	0.159	81.3%	85.6%
Yorktown	0.009	97.9%	0.015	97.5%	0.021	97.0%	**97.5**%

Table 4.2: Final concurrences after Von Neumann protocol is executed once with 8192 shots, together with the percentage wasted entanglement and average wasted entanglement.

We see that Athens and Santiago are the closest to the theoretical limit that we calculated with the simulator. Yorktown again performs worse. We also see that a lot of entanglement is wasted, in all cases more than the theoretical waste, varying from 1.3% to 14.8% more waste.

4.2. Elias's Protocol

We execute Elias's protocol with 2 initial pairs, as described in Section 2.5.2. First, we execute the circuit of Figure 3.10 on the simulator with $\theta = \frac{\pi}{4}$ rad. We find the histogram of measurement probabilities of the simulator (with 2^{16} shots) in the figure below.

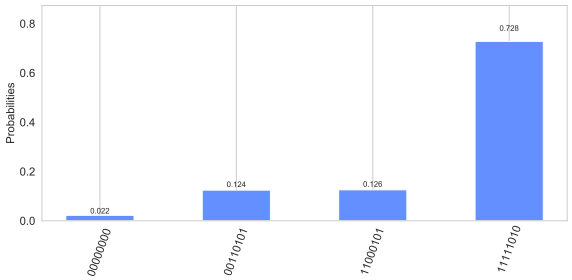


Figure 4.6: Histogram of the measurement probabilities of one try of the Elias's protocol executed on the simulator with $\theta = \frac{\pi}{4}$ rad and 2^{16} shots.

Here the four leftmost bits are the initial states of Alice and Bob: the first and third numbers represent the states of Alice's qubits and the second and fourth the states of Bob's qubits. The four rightmost bits represent the measurement of the Hamming weight of Alice and Bob. For example in the fourth bar we see that the states of Alice and Bob both are $|11\rangle$, and the measured Hamming weight of both Alice and Bob is 2 in binary: 10.

The probability of measuring a Hamming weight of $T = 1$, the only result that yields EPR pairs, is $0.124 + 0.126 = 0.250$, which is in agreement with the theoretical value from 2.39: $p_{suc} = 2p_0p_1 = 0.250$.

Next we execute the same circuit on Melbourne with 8192 shots. We find the histogram of measurement probabilities in Figure 4.7. Here we only kept the states with Hamming weight $T = 1$ (so with '0101' in the rightmost qubits) for clarity. If we compare this histogram to that of the simulator in Figure 4.6, we see that many errors arise. From Melbourne's error rates in Table 3.3, this is what we expected.

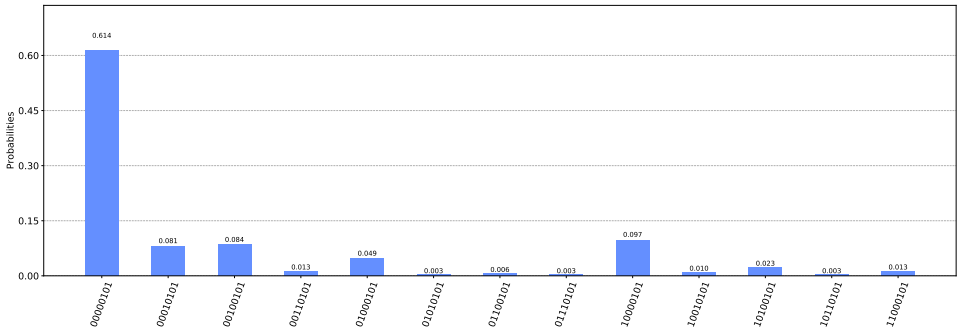


Figure 4.7: Histogram of the measurement probabilities of one try of the Elias's protocol executed on Melbourne with $\theta = \frac{\pi}{4}$ rad and 8192 shots.

We also execute Elias's protocol with 2 initial states for different values of θ. We calculate the concurrence when the protocol succeeds and to compare this

with the initial concurrence, we execute circuit 3.6 on Melbourne as well with 8192 shots. From the histograms of measurement probabilities we retrieve the initial concurrence. We find the results in the table below.

	$\theta = \frac{\pi}{3}$		$\theta = \frac{\pi}{4}$		$\theta = \frac{\pi}{6}$		
Concurrence	Initial	Success	Initial	Success	Initial	Success	Avg Change
Simulator	0.500	1	0.706	1	0.863	1	52.5%
Melbourne	0.3	0.131	0.465	0.141	0.588	0.048	-72.6%

Table 4.3: The concurrence before Elias's protocol is executed once (initial) and after the protocol has succeeded (success) for different initial states.

We see large deviations from the values of the simulator and the final entanglement is way worse than the initial entanglement.

From the measurement results we calculate the rate at which we extract more entangled states from one initial state. From this, we calculate the final concurrences for different θ in radians can be found in the following table, together with the final concurrence of the simulator. Since the concurrence of the resulting state when the protocol succeeds is so low, the final entanglement is low too. We see that Elias's protocol wastes almost all initial entanglement. We again calculate the average waste with eq. 4.6.

	$\theta = \frac{\pi}{3}$		$\theta = \frac{\pi}{4}$		$\theta = \frac{\pi}{6}$		
Concurrence	Final	Wasted	Final	Wasted	Final	Wasted	Avg Waste
Simulator	0.062	87.6%	0.125	82.3%	0.188	78.2%	82.7%
Melbourne	0.001	99.7%	0.002	99.6%	0.021	96.4%	98.6%

Table 4.4: Final concurrences after Elias's protocol is executed once with 8192 shots, together with the percentage wasted entanglement and average wasted entanglement.

5

Discussion

For Von Neumann we first ran experiments with one initial state on a simulator and three 5-qubit processors: Athens, Santiago and Yorktown. As expected the results of the 5-qubit processors contained errors, with the errors of Yorktown being the most significant. We compared the extracted entanglement after the protocol succeeded with the initial entanglement for different initial states. Athens extracted entanglement the best on average, with an average percentage change in the concurrence of 49.0%. Also for each experiment, the concurrence of Athens when the Von Neumann protocol succeeds was the closest to the theoretical concurrence of 1. Santiago also improved the concurrence, on average with 26.5%. The errors of Yorktown are so bad that the concurrence is lower when the protocol succeeds, with an average change of -75.3%.

We also iterated Von Neumann's protocol on the simulator and we see that for more iterations, the probability on one perfect EPR pair increases. However, the Von Neumann protocol wastes a lot of entanglement in theory, on average 82.7%. In practice the waste is even higher, varying from 1.3% to 14.8% more waste than in theory.

In theory, Elias's protocol should reach the theoretical bound for many iterations of the protocol. However, due to limitations on the number of qubits we can use, we only executed Elias's protocol for two initial states on the simulator and on the 15-qubit processor Melbourne. We see that the concurrence when the measured Hamming weight is 1, is always worse than the initial concurrence when we execute the protocol on Melbourne. Also, a lot of concurrence is wasted , the average percentage waste is 98.6%. This is probably in part caused by the error rates of Melbourne, which are higher than error rates of the other quantum processors that we used.

Limitations and Recommendations

In Table 4.1 we see that the initial entanglement of Santiago is larger than that of the simulator. This appears to be counter intuitive, since errors arise on the quantum processors. Thus we expect that the initial entanglement of the quantum processors is lower than the initial concurrence of the simulator. We do observe a this for the other quantum processors.

Also the final concurrence after Elias's protocol for two initial pairs is very low. We do expect that the final concurrence is low, since Elias's protocol requires many operations and thus is more likely to induce errors to the states. However, the final concurrence that we calculate may be too low as a result of the assumption we made in Section 3.1.2: that we measure pure states after the protocols are executed on the quantum processors. This may be too simple, since some quantum processors like Yorktown and Melbourne have large error rates, it is likely that the measured states contain mixed states as well. The mixed states can be uniquely identified by performing quantum tomography. This could affect the results of all experiments with quantum processors, so this it would be an interesting topic for further research.

An interesting result in Table 4.1 is that the concurrence when the protocol succeeds appears to be proportional to the initial concurrence. This is probably caused by the errors of the circuit, though further research would give more insights into specific responsible errors. One way this research could be performed is by executing Von Neumann's protocol multiple times on a simulator by using the Qiskit Aer Noise module[1]. This module allows us to run a circuit with different errors: only read-out errors and perfect gates, only CNOT errors and only SX errors. By comparing these results with the results of running the protocol on a quantum processor, we could examine which errors are responsible the most for the decrease in final entanglement.

When we compare the error rates of Melbourne to the 5-qubit processors, we see that Melbourne has higher error rates. These error rates probably have negative consequences for the results of Elias's protocol and we should be careful to compare the two protocols on the basis of these results. There are IBMQ devices available to researchers with similar error rates to Athens and Santiago (for example Montreal[2]). The Von Neumann's and Elias's protocol could also be compared in a fair way by executing them on the same quantum processor, for example on Melbourne. This ensures that the protocols are both subjected to the same errors. Also, we only executed Elias's protocol for two initial states due to size limitations. For future research it would be interesting to run experiments on a quantum processor with more qubits.

[1] https://qiskit.org/documentation/apidoc/aer_noise.html
[2] https://quantum-computing.ibm.com/services?skip=0&systems=all&system=ibmq_montreal.

6

Conclusion

In this work we wanted to study the performance of two protocols, Von Neumann's and Elias's protocol, for entanglement concentration and compare this to their theoretical performance.

We found that for the first protocol, Von Neumann, the performance depends on the error rate of the backend. The backend with the lowest error rates is Athens and yielded results that were close to the theory. We find that the probability of success is very close to the theoretical value for an initial state with $\theta = \frac{\pi}{4}$: $p_{suc} = 0.251$ versus $p_{theory} = 0.250$. The final state when the protocol succeeds is not a perfect EPR pair, in contrast to the final state of the simulations. Nevertheless Athens can be used for entanglement concentration, since the concurrence when the protocol succeeded improved with on average 49%. Santiago could also be used, though it extracted less concurrence from the initial states on average than Athens: 26.5%. We conclude that Yorktown cannot be used for entanglement concentration: in every experiment the concurrence when the protocol succeeded was less than the initial concurrence, on average -75.3%.

For Elias's protocol, there was only one quantum processor with enough qubits to simulate the protocol for two initial states: the 15-qubit processor Melbourne. The error rates of Melbourne are larger than the error rates of the 5-qubit processors. Also, Elias's protocol requires more operations and qubits than Von Neumann's protocol. Since all quantum processors induce errors for every gate and readout, the performance in terms of entanglement concentration is worse. On average the concurrence of the state with Hamming weight $T = 1$ was 72.6% lower than the initial concurrence and almost all initial concurrence was wasted: on average 96.4%.Thus we conclude that Elias's protocol cannot be used to extract entanglement on Melbourne.

References

[1] R. Blume-Kohout, S. Croke, and D. Gottesman, *Streaming universal distortion-free entanglement concentration,* IEEE transactions on information theory **60**, 334 (2013).

[2] P. Kaye and M. Mosca, *Quantum networks for concentrating entanglement,* Journal of Physics A: Mathematical and General **34**, 6939 (2001).

[3] M. A. Nielsen and I. L. Chuang, *Quantum information and quantum computation,* Cambridge: Cambridge University Press **2**, 23 (2000).

[4] A. Einstein, B. Podolsky, and N. Rosen, *Can quantum-mechanical description of physical reality be considered complete?* Physical review **47**, 777 (1935).

[5] J. S. Bell, *On the einstein podolsky rosen paradox,* Physics Physique Fizika **1**, 195 (1964).

[6] W. An-Min, *A simplified and obvious expression of concurrence in wootters' measure of entanglement of a pair of qubits,* Chinese physics letters **20**, 1907 (2003).

[7] L. Trevisan and S. Vadhan, *Extracting randomness from samplable distributions,* in *Proceedings 41st Annual Symposium on Foundations of Computer Science* (IEEE, 2000) pp. 32–42.

[8] P. Elias, *The efficient construction of an unbiased random sequence,* The Annals of Mathematical Statistics , 865 (1972).

[9] *Qiskit 0.20.0: An open-source framework for quantum computing,* Retrieved from `https://qiskit.org/` (2019).

[10] *5-qubit backend: IBM Q team, IBM Q Athens backend specification: V1.3.16,* Retrieved from `https://quantum-computing.ibm.com` (2021).

[11] *5-qubit backend: IBM Q team, IBM Q Athens backend specification: V1.3.19,* Retrieved from `https://quantum-computing.ibm.com` (2021).

[12] *5-qubit backend: IBM Q team, IBM Q Athens backend specification: V2.3.5,* Retrieved from `https://quantum-computing.ibm.com` (2021).

[13] *15-qubit backend: IBM Q team, IBM Q Melbourne backend specification: V2.3.19,* Retrieved from `https://quantum-computing.ibm.com` (2021).

A

Appendix

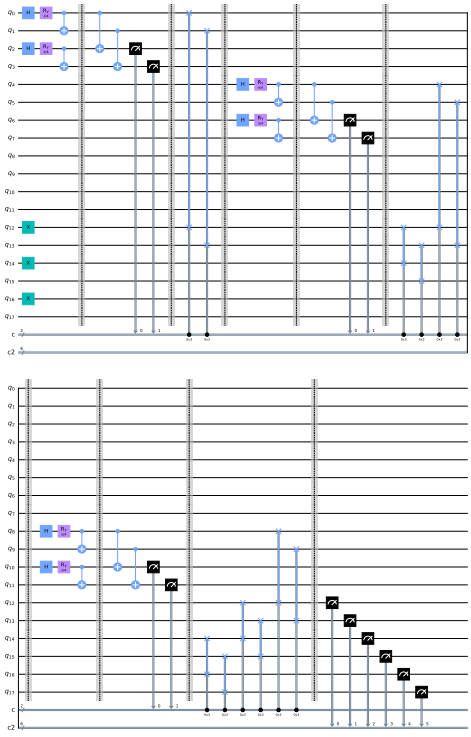


Figure A.1: The implementation of the streaming mode of Von Neumann's protocol.