Dynamic Software Visualization of Quantum Algorithms with Rainbow Boxes

Jean-Baptiste Lamy
LIMICS, Université Paris 13, Sorbonne Université, Inserm, 93017 Bobigny, France

Keywords: Quantum Computing, Multiple-qubit State, Quantum Teleportation, Software Visualization, Set Visualization.

Abstract: Quantum computing has emerged recently as a new computational paradigm. It considers quantum bits (qubits) instead of classical bits. However, quantum algorithms are often very difficult to understand. In this paper, we propose a tool for quantum software visualization. It presents visually the state of multiple-qubits and its evolution at runtime during the execution of a quantum program. This tool allows a unique representation of a quantum state, contrary to the usual vector notation. We show how the problem of visualizing a quantum state can be reduced to a set visualization problem, and our tool uses rainbow boxes to visualize the resulting sets. We also present the application of the proposed tool to quantum teleportation, an algorithm of high importance in cryptography. Finally, we discuss the limit of this approach and its perspectives, in particular for teaching quantum computing.

1 INTRODUCTION

Quantum computing combines quantum physics, computer science and information theory. In the recent years, it has emerged as a new computational paradigm, targeting quantum computers. It has been showed that quantum algorithms could outperform classical algorithms in terms of complexity (Rieffel and Polak, 2014). However, quantum theory is not intuitive. It is considered as almost impossible to understand properly: R. Feynman, a renowned physician, said “I think I can safely say that nobody understands quantum mechanics”. Consequently, it is also difficult to teach.

Software visualization (Price et al., 1993; Bassil and Keller, 2001) is a field that aims at representing graphically software: *e.g.* algorithms, source codes, and/or runtime data. In particular, dynamic software visualization tools have been proposed to explore visually the behavior of a program at run-time, for example in Java (De Pauw et al., 2001). Such tools aim at improving the understanding of what a program is doing, and helping identifying memory leaks, deadlocks and performance bottlenecks. However, to our knowledge, no software visualization approaches were proposed for tracing the execution of quantum programs, despite the use of visualization tools has demonstrated some success in the teaching of quantum physics (Rebello and Zollman, 1999).

In the present paper, we propose a tool for quantum software visualization. Its objective is to facilitate the understanding of a quantum algorithm. It presents visually the quantum state of the system and its evolution at runtime during the execution of a quantum program. We propose to reduce the problem of the visualization of an entangled multiple-qubit state to a set visualization problem. Then, we resolve this problem using rainbow boxes, a technique for set visualization that was introduced a few years ago (Lamy et al., 2016; Lamy et al., 2017). We illustrate the proposed approach through its application to the visualization of quantum teleportation, an algorithm of importance for cryptography.

The rest of the paper is organized as follows. Section 2.1 gives background on quantum computing and set visualization. Section 3 describes how we formalize a quantum state uniquely. It also describes the visual design of our approach and the implementation details. Section 4 provides two examples of the proposed visual approach. Finally, section 5 discusses the proposed approach and its perspectives, before concluding.

Lamy, J.
Dynamic Software Visualization of Quantum Algorithms with Rainbow Boxes.
DOI: 10.5220/0007247801550163
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 155-163
ISBN: 978-989-758-354-4

2 RELATED WORKS

2.1 Quantum Computing

The field of quantum computing (Rieffel and Polak, 2014) has produced some interesting new algorithms, and it has been proved that quantum algorithms can perform better than their classical counterparts, in terms of complexity. However, quantum computing is still a matter of research, and it remains widely not understood.

In a quantum computer, classical bits are replaced by *quantum bits* (qubits). A qubit can be in two states, noted $|0\rangle$ and $|1\rangle$, but it can also be in a *superposition* of these two states, noted $a|0\rangle + b|1\rangle$, where a and b are complex numbers such as $|a|^2 + |b|^2$. In the Dirac's BraKet notation, $|0\rangle$ and $|1\rangle$ are two vectors defined by $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ in their matrix form[1]. These vectors are the orthonormal basis of the qubit's state space, and the general state of a qubit corresponds to $a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$.

However, when measured, the qubit return only a single value, either 0 or 1, with probabilities $|a|^2$ and $|b|^2$, respectively. Future measurement of the same qubit will always return the same value, because measuring the qubit causes a quantum *decoherence* in which the superposition is lost. Thus, a single qubit can store much more information than a classical bit, but all this information cannot be obtained: only one classical bit of information can be extracted from one qubit. Nevertheless, the additional information in the qubit can be used during quantum computation, before measurement.

In addition to the probability of measuring 0 or 1, a 1-qubit state is characterized by its *relative phase* angle: an infinity of superpositions yielding *e.g.* a 50%-50% ratio of 0 and 1 exist. Each of the two terms $a|0\rangle$ and $b|1\rangle$ has a *global phase*, however, only the difference between the two, *i.e.* the relative phase, matters from a physical or logical point of view.

When considering multiple-qubits, the value of the various qubits may not be independent from each other : this phenomenon is known as quantum *entanglement*. Consequently, an entangled system of n qubits cannot be viewed as a sum of qubits. It should be considered as a single entity, and its state is a superposition of 2^n values. For example, the state of a 3-qubit system is a superposition of $2^3 = 8$ values, noted $a|000\rangle + b|001\rangle + c|010\rangle + d|011\rangle + e|100\rangle + f|101\rangle + g|110\rangle + h|111\rangle$ and corresponds to a matrix with 1 column and 8 rows.

Some multiple-qubit states are not *entangled* but *separable*. Those states can be factored in a *tensor product* composed of several states involving fewer qubits. For example, the state $\frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|011\rangle$ can be factored in $|0\rangle \otimes \left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right)$. Here, the first qubit is not entangled, while the last two qubits are. To conclude, the quantity of information stored in multiple qubits grows exponentially with the number of qubits. However, as previously with a single qubit, only one classical bit can be measured for each qubit. Therefore, when read, the 3-qubit system above would returns only 3 classical bits, *e.g.* 011. Relative phases also exist in multiple-qubit states.

Several approaches have been proposed for quantum computing, and quantum circuit is currently the main one. Similarly to a classical logical circuit, gates are applied on the qubits. These gates act on a superposition of values. A gate is defined by an unitary 2^k square matrix, where k is the number of qubits the gate is applied to. The gate is applied to a state vector by multiplying the gate matrix by the vector matrix. For example, NOT (also called X) is a single-qubit gate; it will transform the one-qubit state $\sqrt{\frac{1}{4}}|0\rangle + \sqrt{\frac{3}{4}}|1\rangle$ into $\sqrt{\frac{3}{4}}|0\rangle + \sqrt{\frac{1}{4}}|1\rangle$. Other typical gates include: the Z gate that flips the phase of one qubit, the Hadamard gate (H) that creates superposition from a non-superposed state of one qubit, the CNOT gate (Conditional NOT) that considers two qubits and performs a NOT gate on the second qubit whenever the first qubit is $|1\rangle$, and the CZ gate (Conditional Z) that flips the phase of the second qubit whenever the first qubit is $|1\rangle$. These gates are defined as follows:

$$\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \text{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \text{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \text{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

The following example applies a NOT gate to the state vector $\sqrt{\frac{1}{4}}|0\rangle + \sqrt{\frac{3}{4}}|1\rangle$:

$$\text{NOT} \times \left(\sqrt{\tfrac{1}{4}}|0\rangle + \sqrt{\tfrac{3}{4}}|1\rangle\right) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} \sqrt{\frac{1}{4}} \\ \sqrt{\frac{3}{4}} \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{3}{4}} \\ \sqrt{\frac{1}{4}} \end{pmatrix} = \sqrt{\tfrac{3}{4}}|0\rangle + \sqrt{\tfrac{1}{4}}|1\rangle$$

[1]Matrix rows and columns are understood to be in bit order, *e.g.* 0, 1 for one qubit, 00, 01, 10, 11 for two, *etc.*

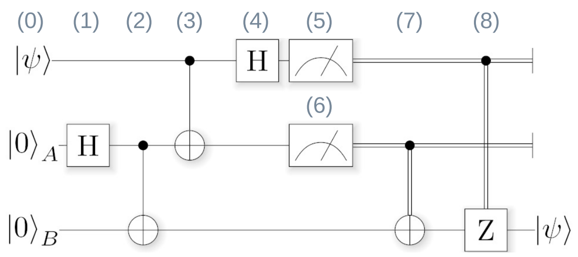

Figure 1: Block diagram of a quantum circuit performing quantum teleportation. Each horizontal line represent a qbit.

2.2 Visual Approaches for Quantum Computing

Few visual approaches have been proposed for quantum computing. First, block circuit diagram representations of quantum circuits have been proposed, *e.g.* for the IBM Q Experience (Cross et al., 2017). They show the qubits as horizontal lines, and the gates and measurements of a given program as boxes on the lines of the qubits they are applied to. Gates and measurements are applied from left to right. For example, Figure 1 shows an example of block diagram for a quantum circuit, with 3 qubits and 8 gates (labeled (1) to (8) and corresponding to the 8 steps of the progam). This circuit corresponds to quantum teleportation, a well-known quantum algorithm that allows to "teleport" the state of one qubit (labelled $|\psi\rangle$) to another qubit. The destination qubit may be physically in a distant place, thanks to quantum entanglement. Hence the name "teleportation".

Block diagrams provide an interesting visualization of a quantum program. However, they do not indicate the qubit states after each gate. Moreover, they can be error-prone because, due to entanglement, applying a gate on a qubit may modify *other* qubits. For example, in Figure 1, the gate (3) is a CNOT gate applied on qubit 1 and 2 (represented by the two top-most horizontal lines). One may abusively consider that qubit 3 (the bottom-most line) is not modified by gate (3). Actually, because the preceeding gate (2) entangled qubit 2 and 3, the states of those two qubits can no longer be separated. Therefore, gate (3) *does* modify qubit 3.

The state of a single qubit $|\psi\rangle = a|0\rangle + b|1\rangle$ is usually visualized by a point at the surface of the Bloch sphere (Figure 2). The point can be identified by two angles θ and φ. The probability of obtaining 0 or 1 when measuring the qubit depends on the position of the point on the vertical axis z and thus on angle θ. Angle φ corresponds to the *relative phase* of the state. It has no impact on the measured value, but it could impact the results of future operations, depending on

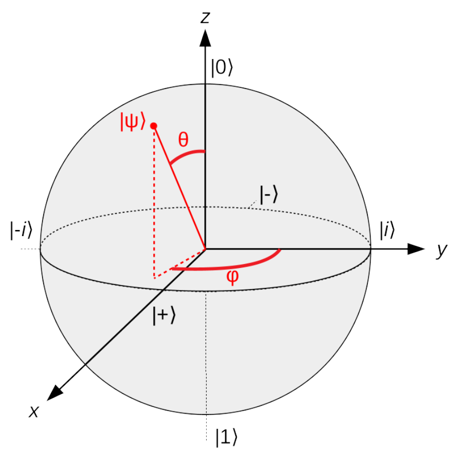

Figure 2: The Bloch sphere.

the gates applied. Single-qubit gates correspond to rotation on the Bloch sphere, for example the NOT gate corresponds to a rotation of 180° around the x axis (thus this gate is also called the X gate). The surface of the Bloch sphere is very practical and visual, but misleading: $|0\rangle$ and $|1\rangle$ are not two opposite directions of the same axis (as shown in Figure 2) but two *perpendicular dimensions* (as the matrix forms of $|0\rangle$ and $|1\rangle$ suggest). Moreover, this approach cannot visualize multiple-qubit states.

The extended complex plane $\mathbf{CP} \cup \{\infty\}$ can also be used for visualizing a single-qubit state, but it is even less intuitive than the surface of the Bloch sphere, because $|0\rangle$ and $|1\rangle$ are represented differently ($|0\rangle$ is the origin while $|1\rangle$ is the "infinity point" noted ∞).

Finally, bar charts are commonly used to represent the probability of the various superposed states (Francik J, 2002), in particular for teaching the Grover algorithm.

2.3 Set Visualization

Set visualization is a classical task in information visualization. Alsakallah *et al.* (Alsallakh et al., 2016) reviewed the techniques for overlapping set visualization. They distinguished 6 approaches: (1) Euler and Venn diagrams and their variants, (2) overlays on a map, (3) node-link diagrams, (4) matrix-based techniques, (5) aggregation-based techniques, and (6) scatter plot-based techniques.

In this work, we chose rainbow boxes, a technique we recently introduced (Lamy et al., 2016; Lamy et al., 2017) for visualizing 2-25 elements and 5-100 sets. Later, a proportional version of rainbow boxes (Lamy and Tsopra, 2017) was proposed for representing artificial neurons. In the present work, rainbow boxes were chosen because (a) they can be gener-

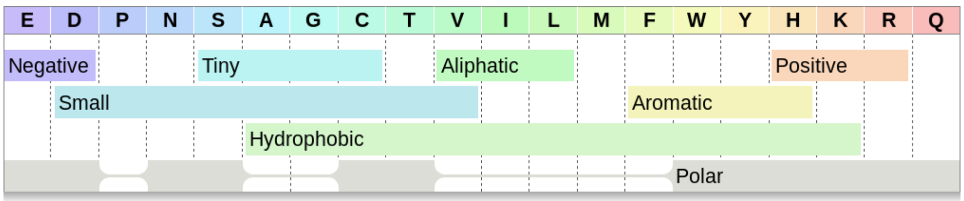

Figure 3: Example of rainbow boxes displaying the properties of amino acids.

ated automatically (contrary to proportional Venn diagrams, which can be generated exactly only up to 4 sets (Rodgers P, 2014)), and (b) they offer several additional visual variables, including box color, box height and box texture (contrary to other diagrams such as Leibniz's linear diagrams (Bellucci et al., 2014)).

Figure 3 shows an example of rainbow boxes displaying amino acid properties (Lamy JB, 2018). The elements (*i.e.* the amino acids, identified by their one-letter code) are shown in columns, and the sets (*i.e.* their shared properties) are represented by rectangular boxes placed below column headers. Each box covers the columns corresponding to the elements belonging to the box's set. The column order is computed using a heuristic optimization algorithm. It tries to order the columns so as the elements belonging to each set are contiguous. When it is not possible to have them contiguous for a given set, "holes" are present in the set's box (in Figure 3, the polar box has 3 holes). Boxes are stacked vertically, with the largest at the bottom. Two boxes can be next to each other, as long as they do not occupy the same columns.

3 METHODS AND DESIGN

3.1 Unique Description of a Multiple-qubit State

Multiple-qubits states are usually formalized by vectors, as in section 2.1. These vectors are useful for computing the effects of gates, but they are not intuitive, due to their complex coordinates. Moreover, they do not exactly represent the physical or logical state of a quantum system. In particular, the vector representation is not *unique*: because of the *global phase* phenomenon, several vectors correspond to the same quantum state. Although mathematically different, any two vectors $|\psi\rangle$ and $e^{i\phi} \times |\psi\rangle$ actually represent the same state, for any value of ϕ (i being the imaginary number). This is denoted $|\psi\rangle \sim e^{i\phi}|\psi\rangle$. For example, $\sqrt{\frac{1}{4}}|0\rangle + \sqrt{\frac{3}{4}}i|1\rangle$ and $\left(\frac{1}{2\sqrt{2}} + \frac{1}{2\sqrt{2}}i\right)|0\rangle + \left(\frac{\sqrt{3}}{2\sqrt{2}}i - \frac{\sqrt{3}}{2\sqrt{2}}\right)|1\rangle$ are two different vectors representing the same state. Thus, we searched for a formal description of a multiple-qubit state that is both unique and as intuitive as possible.

Let us consider the vector form of an entangled multiple-qubit state $|\psi\rangle = a_1|0...0\rangle + ... + a_{2^n}|1...1\rangle = \sum_{j=1}^{2^n} a_j|B_j\rangle$, where $|B_j\rangle$ are the base vectors of the state space. Each term of this sum includes a possible measured base state $|B_j\rangle$ and a complex coefficient $a_j = \alpha_j + \beta_j i$. A (relatively) intuitive notion is the probability of obtaining $|B_j\rangle$ when $|\psi\rangle$ is measured. This probability is $p_j = |a_j|^2 = |\alpha_j|^2 + |\beta_j|^2$. The remnant piece of information describing the state is the phase $\varphi_j = \mathrm{Re}\left(\frac{1}{i} \times \log\left(\frac{a_j}{|a_j|}\right)\right)$ in radians, where Re() returns the real part of a complex number. φ_j is equivalent to angle φ on the Bloch sphere (section 2.2), but for mutiple qubit states. In order to avoid the global phase phenomenon described above, and to have a unique representation of the state, a reference phase must be chosen. We arbitrarily chose as reference the phase φ^0 of the lowest bit-value base state $|B_j\rangle$ with a non-null coefficient a_j (*e.g.* for a 2-qubit state, the phase of $|00\rangle$, or if the corresponding coefficient is null, the phase of $|01\rangle$, then the one of $|10\rangle$, *etc*). Then, the relative phase is $\varphi'_j = \left(\varphi_j - \varphi^0\right) \bmod 2\pi$. Consequently, a pair $(p_j, \varphi'_j) \in \mathbb{R}^2$ (with $0 \leq p_j \leq 1$ and $0 \leq \varphi'_j < 2\pi$, or $\varphi'_j = 0$ for reference phase) allows a unique and more intuitive representation of one coefficient a_j.

Now, let us consider a set of qubits $q = \{q_1, ..., q_n\}$ and their quantum state $|\psi\rangle$, separable in a tensor product of m factors: $|\psi\rangle = |\psi_1\rangle \otimes ... \otimes |\psi_m\rangle$ with $1 \leq m \leq n$. Each factor $|\psi_k\rangle$ can be defined by $q_k \subseteq q$, the subset of qubits it involves, and a sum of terms $|\psi_k\rangle = \sum a_{kj}|B_{kj}\rangle$. As explained above, each coefficient a_{kj} can be fully defined by a pair (p_{kj}, φ'_{kj}), using as reference phase the phase of the lowest bit-value term with a non-null coefficient (since factor are separable, one reference is required for each). Therefore, the state $|\psi\rangle$ can be uniquely defined by a set of quadruplet $\left\{(q_k, |B_{kj}\rangle, p_{kj}, \varphi'_{kj})\right\}$, each quadruplet corresponding to one term of one separable factor.

For example, the previously mentioned 3-qubit

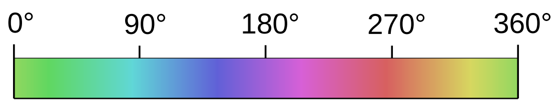

Figure 4: Hue color scale for relative phase.

state $|0\rangle \otimes \left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right)$ is uniquely represented by the following quadruplets:

- $(\{q_1\}, |0\rangle, 1, 0)$
- $(\{q_2, q_3\}, |00\rangle, 0.5, 0)$
- $(\{q_2, q_3\}, |11\rangle, 0.5, 0)$

They are read as follows: "qubit q_1 takes value 0 with probability 100%", "qubits q_1 and q_2 take values 00 with probability 50%", and "qubits q_1 and q_2 take values 11 with probability 50% and a relative phase of 0".

This representation requires to distinguish separable states from entangled ones, in order to determine the reference phases appropriately (*i.e.* one reference per separable factor). We achieved this by tracing entanglement through the program: at the beginning, all qubits are separated. Single-qubit gates do not create entanglement. On the contrary, multiple-qubit gates, like CNOT or CZ, entangle the two qubits they are applied on. Finally, measurements destroy the superposition with regards to the measured qubit, and thus unentangle it.

3.2 Visual Representation

The visualization of a multiple-qubit quantum state defined by a set of quadruplets $\left\{(q_k, |B_{kj}\rangle, p_{kj}, \varphi'_{kj})\right\}$ is a set visualization problem, if we consider the first member of the quadruplets: the qubits q_1, q_2,... are the elements and each quadruplet contains a set made of these elements. The last three members of the quadruplets can be seen as three "attributes" of the set.

We represent those sets with rainbow boxes (Lamy et al., 2016; Lamy et al., 2017), with one column per qubit and one box per quadruplet, *i.e.* one box for each term of each separable factor of the quantum state. In the rainbow boxes, set membership (*i.e.* which qubits are involved by a given term) is visually encoded by the horizontal position of the box. We place the boxes with the lower bit-value at the bottom.

We used three additional visual variables for visualizing the rest of the quadruplets. Probability p_{kj} is represented by the height of the box. Phase φ'_{kj} is represented by the box color. The color is gray when there is no relative phase, *i.e.* when the state of the qubits is certain and there is no superposition. Otherwise, the color hue indicates the relative phase, ranging through the entire rainbow spectrum (see color scale in Figure 4). We choose green to represent a phase of 0 (*i.e.* equal to the reference phase). Finally, the binary value of $|B_{kj}\rangle$ is given in the box label, and it is also represented by the box texture, by adding hatches in the columns corresponding to qubits measured at 1.

We add detail-on-demand interactivity: when the mouse is over a box, a popup label displays the term $a_j|B_j\rangle$ represented by the box, as well as the probability in percentage and the phase angle in degree.

In order to visualize the evolution of the qubit state through the various steps of an algorithm, we juxtapose several rainbow boxes, one per each step.

3.3 Implementation Details

We implemented our system in Python 3. We used ProjectQ (Steiger et al., 2018; Häner et al., 2018), a Python module for quantum computing. ProjectQ can compile quantum circuits for various quantum computers, but it can also *simulate* a quantum computer on a classical computer. In simulation mode, one can access to the "inner state" of the qubits (through the cheat() method) and obtain the complex coefficients a_j that characterize the current quantum state. On the contrary, on a real quantum computer, the inner state cannot be obtained: as explained previously, measuring qubits breaks the superposition. We therefore worked in simulation mode.

4 APPLICATION EXAMPLES

In this section, we illustrate our approach through a simple example on the Bell pair, and the visualization of a quantum teleportation.

4.1 Bell Pair

Figure 5 shows the state of two qubits during the creation and the measurement of a Bell pair. A Bell pair is a pair of two qubits having the maximal level of quantum entanglement. For each step, a rainbow boxes visualization of the current state is given, and labeled with the quantum gate applied at this step. In each visualization, a qubit is represented by a column and a term of a separable factor of the quantum state by a box. Consequently, entangled states are represented by boxes that span across several columns, indicating which qubits are entangled. On the contrary, non-entangled (*i.e.* separable) states are represented by 1-column boxes. For each step, the label uses the ProjectQ syntax, for example "H | q1" means "apply the Hadamard (H) gate to the first qubit".

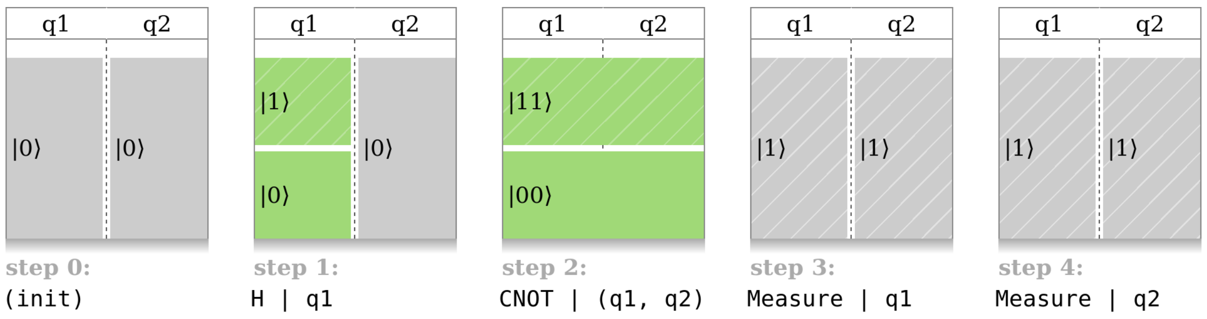

Figure 5: Visualization of the creation and the measurement of a Bell pair, using the proposed approach.

In Figure 5, step 0 corresponds to the initial state of the system. The two qubits q_1 and q_2 have the $|0\rangle$ value and are not entangled. In step 1, we apply the Hadamard (H) gate to q_1. It creates a superposition of $|0\rangle$ and $|1\rangle$. We can see at a glimpse that $|0\rangle$ and $|1\rangle$ would be measured with the same probability (same box size), and that q_1 and q_2 are still not entangled. In addition, the two boxes are green: it indicates that, now, the relative phase does matter. Since the two boxes have the same color, they have the same phase (*i.e.* the relative phase is null). In step 2, we apply the CNOT gate to q_1 and q_2. This gate performs a NOT gate on q_2 whenever $q_1 = |1\rangle$. It results in an entangled state (the boxes span across the two qubits), corresponding to one of the four possible *Bell pairs*.

In step 3, we measure q_1, and the obtained value is 1 (notice that we could also have obtained 0, with a 50-50% ratio). Due to entanglement, measuring q_1 also determine the value of q_2. We can see in the figure that q_2 is now 1, and that the entanglement is now broken. In step 4, we measure q_2 and we obtain 1 (with certainty). Notice that measuring q_2 does not modify the quantum state, contrary to step 3, because the value of q_2 was already determined at the previous step. Consequently, the rainbow boxes are the same after step 3 and 4.

4.2 Quantum Teleportation

Figure 6 shows the state of three qubits during the execution of quantum teleportation. Quantum teleportation is an algorithm with huge potential applications in cryptography. It allows to "teleport" the value $|\psi\rangle$ of a qubit through a shared Bell pair. Figure 1 presented the circuit for quantum teleportation (the step numbers correspond in the two figure).

Step 0 shows the initial step: $q_1 = |\psi\rangle$ contains an interesting value that Alice want to send to Bob, in a secured manner. We arbitrarily chose $|\psi\rangle = (-0.195 + 0.6i)|0\rangle + 0.776|1\rangle$, a 1-qubit state that has a non-null relative phase and a probability of about 40% of being measured at 1. Notice how the colors indicate the relative phase: the $|0\rangle$ box is green (it is the reference phase) and the $|1\rangle$ box has a different color. The other qubits q_2 and q_3 starts at $|0\rangle$, as previously.

Steps 1 and 2 create an entangled Bell pair in q_2 and q_3, as previously. After this step, q_3 can be send to a distant person: Bob, the one to whom we want to "teleport" the value of q_1. Entanglement must be preserved during the sending (this is physically possible, although it remains technically difficult). Sending q_3 is safe, because it currently contains no information about $|\psi\rangle$.

In step 3, we apply the CNOT gate on q_1 and q_2. It entangles all the three qubits, because q_2 and q_3 were already entangled together. However, notice that, at this step, the probability of measuring 0 or 1 for any of the qubit have not changed. By looking at the hatches, we can verify that about 40% of the entire q_1 column have hatches.

In step 4, we apply the H gate on q_1. Due to entanglement, it also modifies q_2 and q_3. Notice that a new color appeared, orange. It is the color opposite to blue, and thus corresponds to the opposite phase. Notice also how "balanced" are the 8 boxes: the probability of measuring 0 or 1 for any qubit, including q_1, is now 50%-50%, and relative phases are also balanced between the blue and the opposite orange phase. However, the initial value $|\psi\rangle$ of q_1 is still encoded in the system. If we look at the two bottom-most boxes, we see that they correspond to $|\psi\rangle$ in terms of phase and of probability (relatively, *i.e.* if we consider only those two boxes). Similarly, the next two boxes just above correspond to $|\psi\rangle$, but are vertically swapped. The next two correspond to $|\psi\rangle$, but with a phase flip: the relative phase is opposed. The last two correspond to $|\psi\rangle$, but both vertically swapped and phase-flipped.

In step 5, we measure q_1. Due to entanglement, it reduces the possible states for the two remnant qubits. Here, we measured 1, thus only states in which the value of q_1 is 1 remain.

In step 6, we measure q_2. Here, we measured

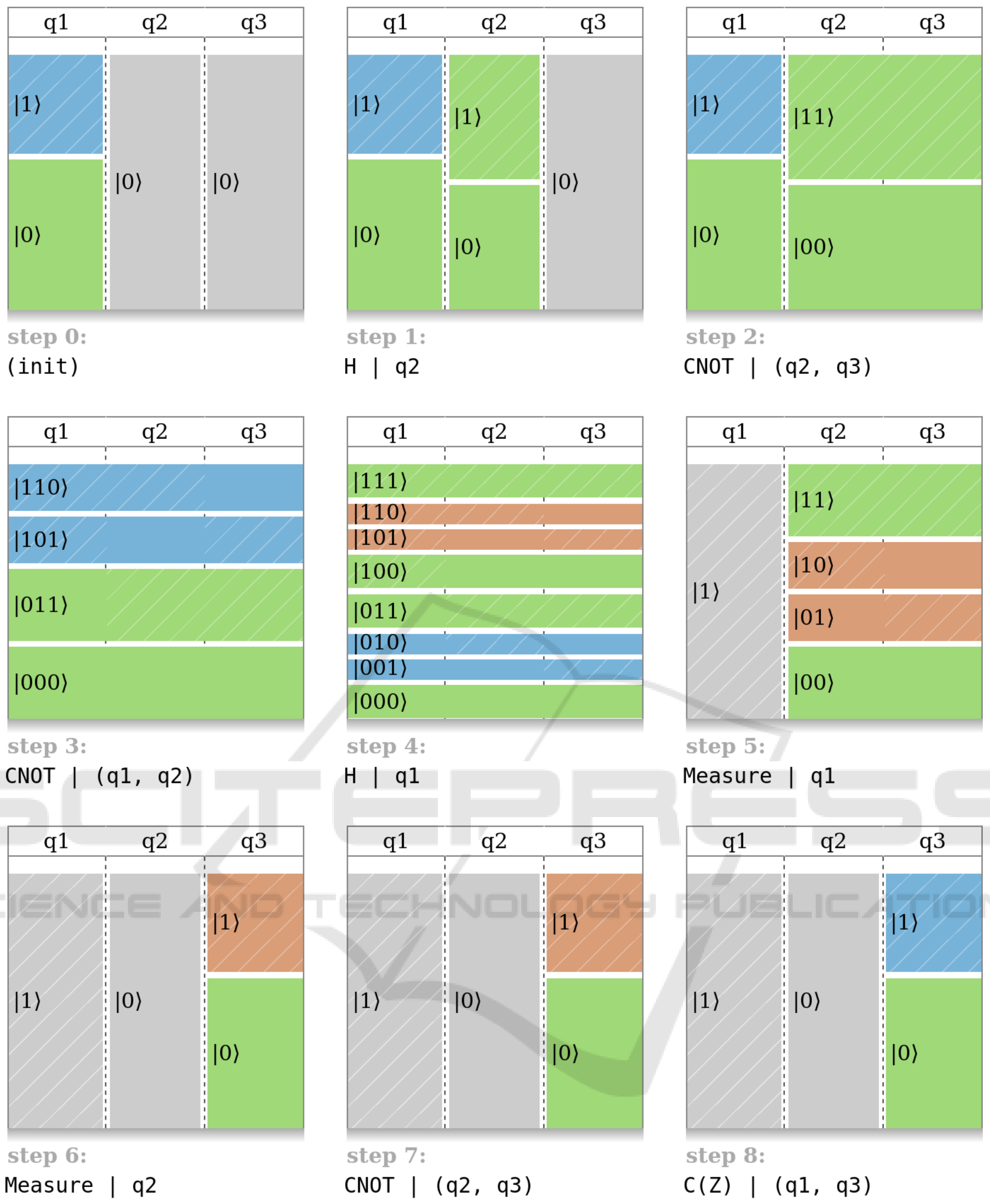

Figure 6: Visualization of the state of the system during the execution of quantum teleportation.

0. Since we measured $q_1 = 1$ and $q_2 = 0$, only the states starting with $|01...\rangle$ remains, compared to step 4. Now, Alice must send to Bob the two binary values she measured, *e.g.* $q_1 = 1$ and $q_2 = 0$. Sending these values is safe, because they are not related to $|\psi\rangle$.

In step 7 and 8, Bob reconstructs the value $|\psi\rangle$, using the binary values measured for q_1 and q_2. If $q_1 = 0$ and $q_2 = 0$, the two superposed states remaining in q_3 correspond to the two bottom-most boxes of step 4. They directly correspond to $|\psi\rangle$. If $q_2 = 1$, the two boxes are vertically swapped, *i.e.* $|0\rangle$ and $|1\rangle$ are swapped. A NOT gate can be used to swap them again, reverting to $|\psi\rangle$. Since this NOT gate must be applied only if $q_2 = 1$, we apply a CNOT gate on q_2 and q_3 (step 7). Then, if $q_1 = 1$, we seen at step 4 that the phase is flipped. Thus we apply a CZ (conditional phase flip) gate on q_1 and q_3 (step 8).

Here, we measured $q_1 = 1$ and $q_2 = 0$. The CNOT gate (step 7) does nothing, because its conditional qubit (q_2) is 0. The CZ gate (step 8) applies the Z gate, because $q_1 = 1$, and flips the phase: the orange box of step 7 is now blue. We can verify visually that the initial state of q_1 is the same that the final state of q_3: we successfully teleported the value $|\psi\rangle$.

5 DISCUSSION AND CONCLUSION

Mathematically speaking, the visualization of an entangled n-qubit state corresponds to the visualization of a point in the complex plane $\mathbf{CP}^{2^n-1}$. In this paper, we showed that it can be reduced to a set visualization problem. We visualized those sets using rainbow boxes, and we demonstrated the proposed approach on the well-known algorithm for quantum teleportation. Our visualization of a quantum state is unique, *i.e.* a given state can be represented in a single manner, contrary to the vectors commonly used in quantum computing.

Our visual approach could be very interesting for teaching quantum computing. Many programmers may be interested in quantum programming in the future, but many of them do not necessarily have the mathematical background required for reading most books on that topic. Moreover, it is well known that quantum theory is not intuitive at all and almost impossible to understand, although it can be used to predict the evolution of a system accurately and it has been widely validated experimentally. Since we still do not understand fully how quantum algorithms work, being able to visualize it during runtime is a step forward. In particular, it greatly helped the author to better apprehend quantum computing.

This visual tool allows an empirical and experimental approach to quantum computing, through the "trial and error" method. It permit testing an algorithm with various initial values or algorithmic variants, and observe how the states of the system are changed. For example, one may visualize quantum teleportation of diverse states $|\psi\rangle$ and observe how changing $|\psi\rangle$ impacts the state of the system at each step. One can also observe how the system states differ when different values are measured for q_1 and q_2. Finally, one can test variants, in order to answer to questions such as "What about performing the two measurements in step 5 and 6 in reverse order?" or "What about performing step 7 and 8 in reverse order?". Through experiments, one would conclude that measurements can be performed in any order, while step 7 and 8 must be performed in order.

In rainbow boxes, finding the optimal column order is usually a complex problem. On the contrary, in this particular application of rainbow boxes, it was not, because of the small number of qubits. Most quantum algorithms use only a few qubits, or a variable number of qubits but can be demonstrated with few of them. Here, we presented the qubits in their order in the registry, without performing optimization at all. It worked well, because, in many algorithm, qubits are already "sorted" in order to put next to each other qubits that are entangled with multiple-qubit gates such as CNOT. If entangled qubits were not next to each other (*e.g.* if we swap $q1$ and $q2$ in Figure 6), holes would appear in the boxes.

To determine which quantum states are separable, we traced entanglement during the program execution. This method works well for simple algorithms, including quantum teleportation. However, it cannot identify all separable states. For example, a second CNOT gate may be used to unentangle an entangled state. A more robust method would consist in performing tensor product factorization from the state vector. But deciding whether a state is separable (*i.e.* the separability problem) is, in the general case, *NP*-hard (Gharibian S, 2010).

In order to represent the evolution of a quantum state, we juxtaposed several rainbow boxes views. A similar approach was proposed by Masoodian *et al.* (Masoodian and Koivunen, 2018) with linear diagrams for the temporal visualization of sets.

Perspectives of this work include (a) the use of proposed visualization with students in computer science and its evaluation in this context, (b) its improvement by implementing automatic tensor product factorization and by integrating it more deeply with ProjectQ to provide a visual platform for quantum computing, and (c) its extension to other quantum computing paradigms beyond quantum circuits, such as adiabatic quantum computation and one-way quantum computation.

REFERENCES

Alsallakh, B., Micallef, L., Aigner, W., Hauser, H., Miksch, S., and Rodgers, P. (2016). The State-of-the-Art of Set Visualization. *Computer Graphics Forum*, 35(1):234–260.

Bassil, S. and Keller, R. K. (2001). Software Visualization Tools: Survey and Analysis. In *International Workshop on Program Comprehension (IWPC)*, pages 7–17.

Bellucci, F., Moktefi, A., and Pietarinen, A. V. (2014). Diagrammatic autarchy: linear diagrams in the 17th and 18th centuries.

Cross, A. W., Bishop, L. S., Smolin, J. A., and Gambetta, J. M. (2017). Open quantum assembly language. *arXiv preprint*, arXiv:1707.03429.

De Pauw, W., Jensen, E., Mitchell, N., Sevitsky, G., Vlissides, J., and Yang, J. (2001). Visualizing the execution of Java programs. In *Revised Lectures on Software Visualization*, pages 151–162 .

Francik J (2002). Quantum software. *Studia informatica*, 23(2A):48.

Gharibian S (2010). Strong NP-hardness of the quantum separability problem. *Quantum Information and Computation*, 10(3&4):343–360.

Häner, T., Steiger, D. S., Svore, K. M., and Troyer, M. (2018). A software methodology for compiling quantum programs. *Quantum Science and Technology*, 3(2):020501.

Lamy, J. B., Berthelot, H., Capron, C., and Favre, M. (2017). Rainbow boxes: a new technique for overlapping set visualization and two applications in the biomedical domain. *Journal of Visual Language and Computing*, 43:71–82.

Lamy, J. B., Berthelot, H., and Favre, M. (2016). Rainbow boxes: a technique for visualizing overlapping sets and an application to the comparison of drugs properties. In *International Conference Information Visualisation (iV)*, pages 253–260, Lisboa, Portugal.

Lamy, J. B. and Tsopra, R. (2017). Translating visually the reasoning of a perceptron: the weighted rainbow boxes technique and an application in antibiotherapy. In *International Conference Information Visualisation (iV)*, pages 256–261, London, United Kingdom.

Lamy JB (2018). A new diagram for amino acids: User study comparing rainbow boxes to Venn/Euler diagram. In *International Conference Information Visualisation (iV)*, pages 361–366, Salerno, Italy.

Masoodian, M. and Koivunen, L. (2018). Temporal visualization of sets and their relationships using Time-sets. In *International Conference Information Visualisation (iV)*, pages 85–90.

Price, B. A., Baecker, R. M., and Small, I. S. (1993). A principled taxonomy of software visualization.

Rebello, N. S. and Zollman, D. (1999). Conceptual understanding of quantum mechanics after using hands-on and visualization instructional materials. In *Annual meeting national association for research in science teaching*, volume 2.

Rieffel, E. and Polak, W. (2014). *Quantum computing : A gentle introduction*. The MIT Press, Cambridge, Massachussetts, USA.

Rodgers P (2014). A survey of Euler diagrams. *Journal of Visual Languages and Computing*, 25(3):134–155.

Steiger, D. S., Häner, T., and Troyer, M. (2018). ProjectQ: An open source software framework for quantum computing. *Quantum*, 2:49.