Solving NP-complete Problems Using Quantum Weightless Neuron Nodes

Conference Paper · November 2015

DOI: 10.1109/BRACIS.2015.22

4 authors:

Fernando Maciano De Paula Neto
Federal University of Pernambuco
14 PUBLICATIONS **47** CITATIONS
SEE PROFILE

Teresa Bernarda Ludermir
Federal University of Pernambuco
369 PUBLICATIONS **3,136** CITATIONS
SEE PROFILE

Wilson Rosa de Oliveira
Universidade Federal Rural de Pernambuco
59 PUBLICATIONS **296** CITATIONS
SEE PROFILE

Adenilton José da Silva
Federal University of Pernambuco
44 PUBLICATIONS **228** CITATIONS
SEE PROFILE

Solving NP-complete Problems using Quantum Weightless Neuron Nodes

Fernando M. de Paula Neto, Teresa B. Ludermir
Centro de Informática
Universidade Federal de Pernambuco
Recife, Pernambuco
Email: {fmpn2, tbl}@cin.ufpe.br

Wilson R. de Oliveira, Adenilton J. da Silva
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Pernambuco
Email: wilson.rosa@gmail.com, ajs@deinfo.ufrpe.br

***Abstract*—Despite neural networks have super-Turing computing power, there is no known algorithm for obtaining a classical neural networks that solves NP-complete problems in polynomial time. However this paper shows that a quantum neural networks model coupled with a non-unitary operator can solve 3-SAT in polynomial time. The proposed method uses a network circuit to represent a Boolean logic function and a non-unitary operator to decide the satisfiability. The parameters of the network is set deterministically and manually, accordingly to the problem at hand with neither quantum nor classical learning.**

***Keywords*-Quantum computing; Weightless neuron; NP-complete problem; Non unitary;**

I. Introduction

Classical computers apparently have limitations to efficiently solve complex problems such as those in the NP-complete class. The travelling salesman, the graph coloring, the knapsack and the satisfiability (SAT) problems, are examples that are in NP-complete class and are so hard that solve one them in polynomial time implies we can also solve all problems in NP [18]. These are technically called NP-complete problems.

Neural networks have been used to solve problems inductively, using a training set to adjust their parameters. In the supervised training there are expected targets in each training example so the network has its parameters adjusted for minimising the measured error in relation of the expected value. That inductive learning gives an approximate solution not necessarily the best one. Several neural network models were proposed to solve NP-complete problems but none always gives a precise correct solution due to its stochastic behaviour and limitation of heuristic method of updating the network parameters [11][10].

Quantum neural networks were firstly proposed in the nineties and several works propose concrete models of quantum neural networks [3][6]. Some advantages of quantum neural networks are the capacity of a single neuron to solve non-linearly separable patterns, learning algorithms with polynomial cost in relation to the number of patterns in training set and an exponential gain in memory capacity [7].

Non-unitary and non-linear operators [1][19][20] combined with quantum gates solve analytically NP-complete problems in polynomial time [13][15]. In this paper, we show how to solve 3-SAT problem using a quantum weightless neuron circuit [6] and a non-unitary gate [12].

This paper is organized as follows. In Section II and Section III we discuss quantum computing and quantum weightless neuron nodes, respectively. 3-SAT Problem is defined in Section IV and the 3-SAT proposed solution using qRAM circuit representation for Boolean logic functions is presented in Section V, followed by an example. The conclusions are presented in Section VI.

II. Quantum Computing

A quantum bit, *qubit*, is a special complex bidimensional unitary vector. The computational basis for the vector space, $\mathbb{C}^2$, of all qubits is composed of the vectors written in the Dirac or bra-ket nottion: $|0\rangle = [1,0]^T$ and $|1\rangle = [0,1]^T$. An arbitrary qubit $|\psi\rangle$ can be written as linear combination (*superposition*) of the computational basis as shown in Equation (1), where α and β are complex numbers and subject to the restriction that $|\alpha|^2 + |\beta|^2 = 1$. Qubits represents the states of a single quantum system. Composition of quantum systems are obtained with tensor products: $|ij\rangle = |i\rangle \otimes |j\rangle$.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

Quantum system evolves by a unitary operators that change the state amplitude values. A quantum operator $\mathbf{U}$ over n qubits is a $2^n \times 2^n$ complex unitary matrix. Some main operators over one qubit are the identity operator $\mathbf{I}$, the flip operator $\mathbf{X}$ and Hadamard $\mathbf{H}$ operator, described in Equation (2) and Equation (3). A quantum circuit is a combination of unitary operators applied to one or some set of qubits.

$$\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{array}{l} \mathbf{I}|0\rangle = |0\rangle \\ \mathbf{I}|1\rangle = |1\rangle \end{array} \quad \mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{array}{l} \mathbf{X}|0\rangle = |1\rangle \\ \mathbf{X}|1\rangle = |0\rangle \end{array} \tag{2}$$

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \begin{array}{l} \mathbf{H}|0\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle) \\ \mathbf{H}|1\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle) \end{array} \tag{3}$$

The identity operator **I** outputs the input; the flip operator **X** behaves as the classical **NOT** on the computational basis; Hadamard transformation **H** generates an even superposition of states. The **CNOT** operator has two input qubits and two output qubits and flips the second qubit if the first one is 1 as show in Figure 1.

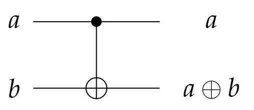

Figure 1. The **CNOT** operator

III. Quantum Neurons

Many quantum neurons have been proposed generalising the (classical) artificial neural networks [7][16]. Some of them are only quantum inspired but their working is not intrinsically quantum once their learning algorithms interfere with network parameters violating some quantum postulates [4]. The qRAM neuron node [6][5] studied here is a generalisation of RAM based nodes, used in many applications of pattern recognition and classification tasks [14]. The RAM based nodes have good generalization capabilities [9] and computational power [8].

A. qRAM - Quantum Neuron RAM-based Node

In [6] and [5], the quantum RAM based neuron was defined as quantization of the weightless neural networks proposed in [2].

The RAM node stores in its memory one bit addressed by an input bit string. The qRAM represents that bit storage by the gate A, as showed below. The gate A is the classical behavior of a memory addressed by one bit:

$$A = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix} \quad \begin{array}{l} \text{where} \\ A|00\rangle = |0\rangle I|0\rangle \\ A|10\rangle = |1\rangle X|0\rangle \end{array} \tag{4}$$

If the first input bit is zero, A matrix outputs $|0\rangle$ in the second position. Otherwise, A matrix outputs $|1\rangle$ in the second position. Then, the first bit called selector is changed to modify the content loaded. The second bit is always zero and is considered an output register because is only accessed in the final. A qRAM of input

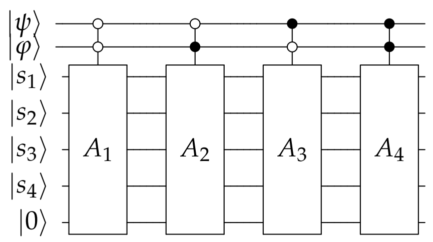

Figure 2. qRAM node with two inputs, four selectors and one output representation.

n-qubits has a collection of 2^n A's. Each operator A has its selector to change its content loaded. qRAM circuit representation is shown in Figure 2. To train a qRAM circuit one needs to change the selectors values to answer correctly with the input.

In general, given an input qubit, the qRAM model chooses the selectors that will be applied to the output. If the chosen selector value is one, the output qubit will be one. Otherwise, the output qubit will be zero. If the input qubit is in superposition state, the selectors will be chosen in superposition, resulting in an output qubit in superposition.

One can represent the qRAM with the implicit selector representation as shown in Figure 3. This notation is introduced in [6] as q-ROM, quantum read-only memory, since the addressed content cannot be modified and the quantum operators U that implicitly represent the selectors are fixed. Training a q-ROM, shown in Figure 3, is to choose the operator U_i for each i positions.

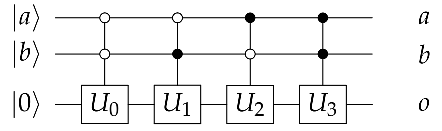

Figure 3. q-PLN node, where U_i, $i = 0, 1, ..., 2^n - 1$, can be I or X gates.

IV. 3-SAT Problem

The Boolean Satisfiability Problem (SAT) is, given a propositional logic expression $\phi(x_1, x_2..., x_n)$, to decide if there are logical values for each x_i that turn the expression ϕ true.

The 3-SAT problem restricts the SAT logic expression format. A literal is a Boolean variable or a negated Boolean variable. A clause is a disjunction of literals and a 3-clause is a disjunction of exactly 3 literals. A 3-SAT expression is formed by conjunctions of 3-clauses. Since each clause contains exactly three literals, the expression will be at most $(2n)^3 = 8n^3$ clauses, where n is the number of variables.

A problem p is NP-complete if satisfies two conditions (a) it is in NP, and (b) every problem in NP is polynomial time reducible to p. These problems equally difficult to solve and have to date only be solved by algorithms with an exponential number of steps as function of the input size. In this class of problems, there is no known polynomial time solution. 3-SAT problem is NP-complete [17].

V. Solving 3-SAT using qRAM

In this Section, we use the quantum node qRAM in the solution of the 3-SAT using a non-unitary operator. The qRAM nodes are used to represent the logic expressions. The non-unitary operator applied to the qRAM circuit provides a null vector as output only if the logic expression is not satisfiable and provides a superposition vector in case the expression is satisfiable.

A. Logic expression as qRAM circuits

To set a logic expression in a qRAM circuit, we configure the selectors $|\mathbf{s}\rangle$ of each qRAM node to represent a logic operation of each clause. As example, for a logic expression of two variables a e b formed by the disjunction $a \vee b$, the selectors $|s_1\rangle$, $|s_2\rangle$ and $|s_3\rangle$ are set to $|1\rangle$ and the selector $|s_0\rangle$ is set to $|0\rangle$. The disjunction operation qRAM representation is presented in Figure 4. For a conjunction representation, the selector $|s_3\rangle$ is set to $|1\rangle$ and the others selectors are set to $|0\rangle$. That procedure is extended for qRAMs with more than two inputs.

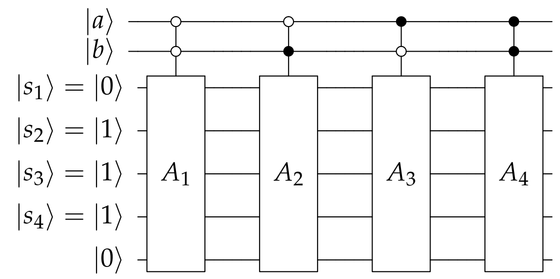
$|a\rangle$
$|b\rangle$
$|s_1\rangle = |0\rangle$
$|s_2\rangle = |1\rangle$
$|s_3\rangle = |1\rangle$
$|s_4\rangle = |1\rangle$
$|0\rangle$
A_1
A_2
A_3
A_4

Figure 4. qRAM representing disjunction operation of two inputs $|a\rangle$ and $|b\rangle$.

A logic expression is formed by a circuit with qRAMs where each one represents a clause. Quantum ancillary registers load partial results for each operation. The last qRAM node calculates the conjunction of the partial outputs giving the final output. In a conjunctive normal form which has c clauses, it is necessary $c+1$ output registers $|o\rangle$.

In a general construction, a clause with n variables, i.e. a logic expression which executes a disjunction operator in the n inputs, the qRAM selector $|s_0\rangle$ is set to $|0\rangle$, and all the others ones $|s_1, s_2..., s_{2^n}\rangle$ are set to $|1, ..., 1\rangle$. In the case the operation is a conjunction, only the selector s_{2^n} is set to $|1\rangle$, and the others are set to $|0\rangle$.

Some Boolean expressions have negated variables $|\hat{x}\rangle$. For representing these variables, it is proposed to use one additional input in the qRAM. The quantum circuit designer is responsible to feed in that input the negated representation of the respective variable $|x\rangle$. This is done simply by the negation of the variable by the NOT gate X. In this way, expressions of n free variables, qRAM circuits will have $2n$ inputs.

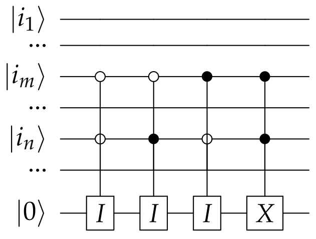

Figure 5. Example of qROM prepared to a conjunctive operation with no selectors representation. The gates I e X do the configuration role. This example considers the m and n-th register as input.

To process a clause in a qRAM circuit, a qRAM node needs to choose what variables will be used as input in the operation. It is possible the neuron does not need use them alls. A qRAM $^{OR}_{S|k}$ applies its disjunction operation in the set of positions $S \subseteq \{t | 1 < t < p\}$, where n is the quantity of input variables, m is the clauses quantity, and $p = 2^n + m + 1$ and the output is load in the k-th output register. Figure 5 represents a qROM setting for a conjunction operation and Figure 6 show the same circuit but in this representation the internal configuration is hidden for better displaying the image.

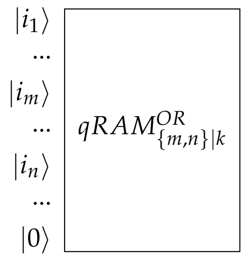

Figure 6. Representation of qRAM node. The selectors and U matrices are hidden in the circuit representation to simplify the notation. This example considers the m and n-th registers of input and put the result of the OR operation in the k-th register

In Figure 7, it is shown an example of qRAM circuit of the Boolean logic that express $\phi(x_1, x_2) = (x_1 \vee x_2) \wedge (\bar{x_1} \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_2})$ using quantum ROMs. For simplify our notation, we use qRAMs notation even in the qROMs configurations, i.e. when the selectors are hidden.

1) Solving 3-SAT: Then, to solve a 3-SAT problem, a qRAM circuit is created to represent a Boolean expression. To calculate all possibles values in a single shot operation, the inputs are prepared in superposition. The output registers are set to zero. The neuron circuit calculates the logic expression and the non-unitary operator verifies if any input configuration turns the expression true [12].

As some inputs of the circuits are in negated representation of some variables, we cannot put all inputs in the superposition as $|x, \hat{x}\rangle$ would be wrongly set to $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. In the case which negated inputs are used, a Bell state for this couple qubit is created, as described in Figure 8: $|x, \hat{x}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$.

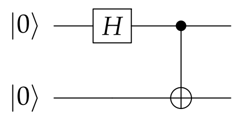

Figure 8. To solve 3-SAT problems, the qRAM circuit receives all inputs in superposition. In the case in which variables and their negated representation are input, a Bell state is set to input these couple of qubits, since the possible values are $|01\rangle$ e $|10\rangle$ and not all values.

After applying the inputs in the qRAM circuit, the next step is apply the non-unitary operator [12] in the last output qubit:

$$O = 2^n \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{5}$$

That operator does a check if some final output qubit was flipped to $|1\rangle$, i.e. if some input value turns the logic expression true. To a circuit with n variables and k clauses, we have $2n$ registers of input, $k+1$ output registers, then $m = 2n + k + 1$ registers in total. To apply the non-unitary operator described in Equation 5 only in the last total output register, the operator above is used:

$$O^{(m)} = (\otimes^{m-1} I_1) \otimes O \tag{6}$$

where I is identity gate. Finally, the operator O plays the role for checking the output. This is done by applying the operator O and verifying if the result is a null vector or not. Since $O|0\rangle = 2^n|1\rangle\langle 1||0\rangle = \mathbf{0}$, then if our final quantum vector is null, no input turns the final output register to $|1\rangle$, i.e. the logic expression is not satisfiable.

Otherwise, if there is one solution, the final output qubit is $|1\rangle$, even in superposition state, our final quantum vector is not a null vector. For example, given a superposition state, applying the operator O results in $O(\alpha_0|0\rangle + \beta_0|1\rangle) = \alpha_0 2^n|1\rangle\langle 1||0\rangle + \alpha_1 2^n|1\rangle\langle 1||1\rangle = \mathbf{0} + \alpha_1 2^n|1\rangle = \alpha_1 2^n|1\rangle$ [12].

As an example, given an expression

$$\begin{aligned}\phi(x_1, x_2, x_3, x_4) = (\hat{x_1} \vee x_2 \vee x_4) \wedge (\hat{x_2} \vee x_3 \vee x_4) \wedge \\ (x_1 \vee \hat{x_3} \vee x_4) \wedge (x_1 \vee \hat{x_2} \vee \hat{x_4}) \wedge (x_2 \vee \hat{x_3} \vee \hat{x_4}) \wedge \\ (\hat{x_1} \vee x_3 \vee \hat{x_4}) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\hat{x_1} \vee \hat{x_2} \vee \hat{x_3})\end{aligned} \tag{7}$$

We need six qRAMs, five ones are used to do a disjunction in each clause and one qRAM does a conjunctive composition of the partial outputs.

Analytically, the input in superposition to apply in the qRAM circuit:

$$\begin{aligned}|\psi\rangle &= |x_1, \hat{x_1}, x_2, \hat{x_2}, x_3, \hat{x_3}, x_4, \hat{x_4}, o_1, o_2, o_3, o_4, o_5, o_6\rangle \\ &= (\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \otimes \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \otimes \\ &\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \otimes \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)) \otimes (|000000\rangle)\end{aligned} \tag{8}$$

Distributively, the input vector is also equal to:

$$\begin{aligned}|\psi\rangle &= |x_1, \hat{x_1}, x_2, \hat{x_2}, x_3, \hat{x_3}, x_4, \hat{x_4}, o_1, o_2, o_3, o_4, o_5, o_6\rangle \\ &= |01010101\rangle|000000\rangle + |01010110\rangle|000000\rangle + \\ &|01011001\rangle|000000\rangle + |01011010\rangle|000000\rangle + \\ &|01100101\rangle|000000\rangle + |01100110\rangle|000000\rangle + \\ &|01101001\rangle|000000\rangle + |01101010\rangle|000000\rangle + \\ &|10010101\rangle|000000\rangle + |10010110\rangle|000000\rangle + \\ &|10011001\rangle|000000\rangle + |10011010\rangle|000000\rangle + \\ &|10100101\rangle|000000\rangle + |10100110\rangle|000000\rangle + \\ &|10101001\rangle|000000\rangle + |10101010\rangle|000000\rangle\end{aligned} \tag{9}$$

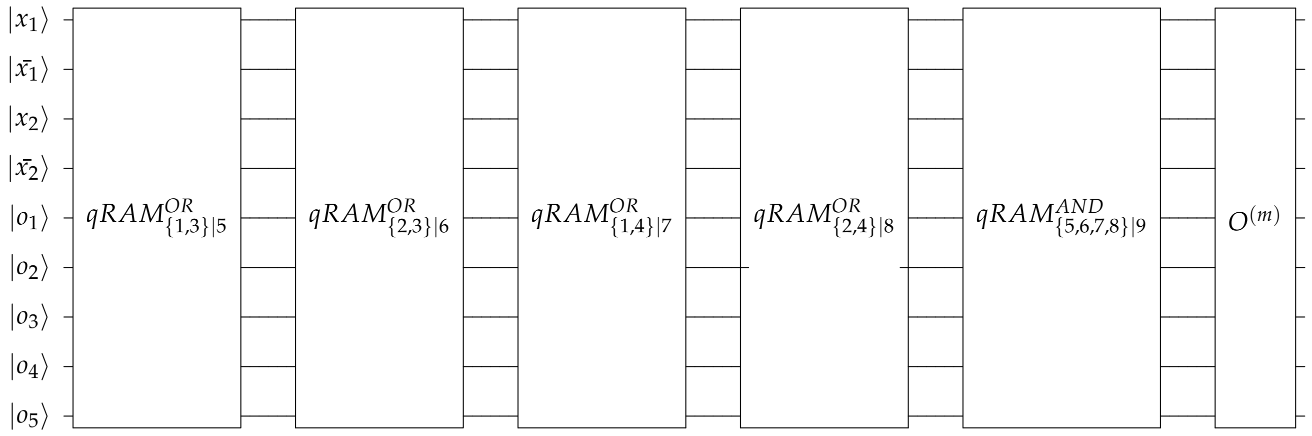

Figure 7. qRAM Circuit representing the logic function $\phi(x_1, x_2) = (x_1 \vee x_2) \wedge (\bar{x_1} \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_2})$. Each qRAM node $qRAM_{S|k}$ has its indexes input set and define the k-th output to load its result. The register o_i loads the result of the i-th clause and the register o_5 loads the final output. The non-unitary operator $O^{(m)}$ verifies if the input values satisfies the equation. If the final state is a null vector the input vector does not satisfy the equation. Otherwise, it satisfies.

When the first qRAM processes the first clause, $(\hat{x_1} \vee x_2 \vee x_4)$ the first register $|o_1\rangle$ is updated and the input vector is:

$$\begin{aligned}
&|x_1, \hat{x_1}, x_2, \hat{x_2}, x_3, \hat{x_3}, x_4, \hat{x_4}, o_1, o_2, o_3, o_4, o_5, o_6\rangle = \\
&|01010101\rangle|100000000\rangle + |01010110\rangle|100000000\rangle + \\
&|01011001\rangle|100000000\rangle + |01011010\rangle|100000000\rangle + \\
&|01100101\rangle|100000000\rangle + |01100110\rangle|100000000\rangle + \\
&|01101001\rangle|100000000\rangle + |01101010\rangle|100000000\rangle + \\
&|10010101\rangle|100000000\rangle + |10010110\rangle|100000000\rangle + \\
&|10011001\rangle|100000000\rangle + |10011010\rangle|100000000\rangle + \\
&|10100101\rangle|000000000\rangle + |10100110\rangle|000000000\rangle + \\
&|10101001\rangle|000000000\rangle + |10101010\rangle|000000000\rangle
\end{aligned} \tag{10}$$

When the second qRAM processes the second clause, $(\hat{x_2} \vee x_3 \vee x_4)$, the second output register $|o_2\rangle$ is updated and the input vector is:

$$\begin{aligned}
&|x_1, \hat{x_1}, x_2, \hat{x_2}, x_3, \hat{x_3}, x_4, \hat{x_4}, o_1, o_2, o_3, o_4, o_5, o_6\rangle = \\
&|01010101\rangle|110000000\rangle + |01010110\rangle|110000000\rangle + \\
&|01011001\rangle|110000000\rangle + |01011010\rangle|110000000\rangle + \\
&|01100101\rangle|110000000\rangle + |01100110\rangle|110000000\rangle + \\
&|01101001\rangle|110000000\rangle + |01101010\rangle|110000000\rangle + \\
&|10010101\rangle|110000000\rangle + |10010110\rangle|110000000\rangle + \\
&|10011001\rangle|110000000\rangle + |10011010\rangle|110000000\rangle + \\
&|10100101\rangle|010000000\rangle + |10100110\rangle|010000000\rangle + \\
&|10101001\rangle|010000000\rangle + |10101010\rangle|010000000\rangle
\end{aligned} \tag{11}$$

When the third qRAM processes the third clause, $(x_1 \vee \hat{x_3} \vee x_4)$, the third register $|o_3\rangle$ is updated and the input vector is:

$$\begin{aligned}
&|x_1, \hat{x_1}, x_2, \hat{x_2}, x_3, \hat{x_3}, x_4, \hat{x_4}, o_1, o_2, o_3, o_4, o_5, o_6\rangle = \\
&|01010101\rangle|111000000\rangle + |01010110\rangle|111000000\rangle + \\
&|01011001\rangle|111000000\rangle + |01011010\rangle|111000000\rangle + \\
&|01100101\rangle|110000000\rangle + |01100110\rangle|110000000\rangle + \\
&|01101001\rangle|110000000\rangle + |01101010\rangle|110000000\rangle + \\
&|10010101\rangle|111000000\rangle + |10010110\rangle|111000000\rangle + \\
&|10011001\rangle|111000000\rangle + |10011010\rangle|111000000\rangle + \\
&|10100101\rangle|011000000\rangle + |10100110\rangle|011000000\rangle + \\
&|10101001\rangle|011000000\rangle + |10101010\rangle|011000000\rangle
\end{aligned} \tag{12}$$

Finally, when the sixth first qRAMs process the disjunction of the clauses, the input vector is:

$$\begin{aligned}
&|x_1, \hat{x_1}, x_2, \hat{x_2}, x_3, \hat{x_3}, x_4, \hat{x_4}, o_1, o_2, o_3, o_4, o_5, o_6\rangle = \\
&|\psi\rangle = \\
&|01010101\rangle|111011110\rangle + |01010110\rangle|111011110\rangle + \\
&|01011001\rangle|111011110\rangle + |01011010\rangle|111011110\rangle + \\
&|01100101\rangle|110111110\rangle + |01100110\rangle|110111110\rangle + \\
&|01101001\rangle|110111110\rangle + |01101010\rangle|110111110\rangle + \\
&|10010101\rangle|111110110\rangle + |10010110\rangle|111110110\rangle + \\
&|10011001\rangle|111110110\rangle + |10011010\rangle|111110110\rangle + \\
&|10100101\rangle|011111110\rangle + |10100110\rangle|011111110\rangle + \\
&|10101001\rangle|011111110\rangle + |10101010\rangle|011111110\rangle
\end{aligned} \tag{13}$$

As can be seen, in each state there is at least one register set to zero in some output register $|o_1, ..., o_8\rangle$. Then, the last qRAM in the circuit will not alter the final output register $|o_9\rangle$ by the conjunction operation.

Applying Equation 5 in the output quantum state, the state $|\psi\rangle$ transforms to $|\psi\rangle'$:

$$O^{(m)} = (\otimes^{m-1} I_1) \otimes O \quad (14)$$

$$|\psi\rangle' = O^{(m)}|\psi\rangle \quad (15)$$

Verifying the value of $|\psi\rangle'$, we check that is a null vector $\mathbf{0}$, as expected, since the expression is not satisfiable: $|\psi'\rangle = \mathbf{0}$.

VI. Conclusion

In this paper, we demonstrate that it is possible to solve the 3-SAT problem using a non-unitary operator quantum gate and a quantum weightless neural network in polynomial time. Each logic function clause is represented by a qRAM node, setting the selectors according its operation. The non-unitary operator verifies weather the function is satisfiable or not. No one knows if there is a configuration of a classical weightless neuron networks to solve 3-SAT, this would implies that $P = NP$, but using a quantization of weightless neuron node, 3-SAT problem is solved in polynomial time. There are controversies about physical implementation of non-unitary operators, notwithstanding there are some papers showing its working [19] [20].

The logic function representation can be studied further improved as regarding the circuit size, although it grows polynomially in the input size. Further studies in quantum approximated heuristic solutions for NP-complete problems are envisioned [20].

Acknowledgment

The authors would like to thank CNPq and FACEPE (Brazilian Research Agencies) for their financial support.

References

[1] D. S. Abrams and S. Lloyd. Nonlinear Quantum Mechanics Implies Polynomial-Time Solution for NP-Complete and P Problems. *Phys. Rev. Lett.*, 81(18):3992–3995, 1998.

[2] I. Aleksander. Self-adaptive universal logic circuits. *Electronics Letters 2*, 8:321–322, 1966.

[3] M. V. Altaisky. Quantum neural network. Technical report, Joint Institute for Nuclear Research, Russia, 2001.

[4] A. J. da Silva, W. R. de Oliveira, and T. B. Ludermir. Comments on "quantum m-p neural network". *International Journal of Theoretical Physics*, 54(6):1878–1881, 2015.

[5] W. R. de Oliveira. Quantum RAM Based Neural Networks. In *ESANN*, pages 22–24, 2009.

[6] W. R. de Oliveira, A. J. da Silva, T. B. Ludermir, A. Leonel, W. R. Galindo, and J. C. Pereira. Quantum Logical Neural Networks. In *Brazilian Symposium on Neural Networks*, pages 147–152, 2008.

[7] F. M. de Paula, A. J. da Silva, T. B. Ludermir, and W. R. de Oliveira. Analysis of Quantum Neural Models. *XI Brazilian Congress of Computational Intelligence*, 1:1–6, 2013.

[8] M. C. P. de Souto, T. B. Ludermir, and W. R. de Oliveira. Equivalence between ram-based neural networks and probabilistic automata. *Neural Networks, IEEE Transactions on*, 16(4):996–999, 2005.

[9] A. F. De Souza, F. Pedroni, E. Oliveira, P. M. Ciarelli, W. F. Henrique, L. Veronese, and C. Badue. Automated multi-label text categorization with vg-ram weightless neural networks. *Neurocomput.*, 72(10-12):2209–2217, June 2009.

[10] J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

[11] J.-S. Lai, S.-Y. Kuo, and I.-Y. Chen. Neural networks for optimization problems in graph theory. In *Circuits and Systems, 1994. ISCAS '94., 1994 IEEE International Symposium on*, volume 6, pages 269–272 vol.6, May 1994.

[12] A. Leporati and S. Felloni. Three "quantum" algorithms to solve 3-sat. *Theoretical Computer Science*, 372(2–3):218 – 241, 2007.

[13] A. Leporati, C. Zandron, and G. Mauri. Simulating the fredkin gate with energy-based p systems. *J. UCS*, 10(5):600–619, 2004.

[14] T. B. Ludermir, A. de Carvalho, A. P. Braga, and M. C. P. de Souto. Weightless neural models: a review of current and past works. *Neural Computing Surveys*, 2:41–61, 1989.

[15] M. Ohya and I. Volovich. Quantum Computing, NP-complete Problems and Chaotic Dynamics. *eprint arXiv:quant-ph/9912100*, Dec. 1999.

[16] M. Schuld, I. Sinayskiy, and F. Petruccione. The quest for a quantum neural network. *Quantum Information Processing*, 13(11):2567–2586, 2014.

[17] M. Sipser. *Introduction to the Theory of Computation*. 2006.

[18] N. Srinivasan. Tutorial: Current state of p vs np problem. In *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, pages 1–1, June 2011.

[19] H. Terashima and M. Ueda. Nonunitary quantum circuit. *eprint arXiv:quant-ph/0304061*, Apr. 2003.

[20] C. Williams and R. Gingrich. Non-unitary probabilistic quantum computing circuit and method, Aug. 4 2005. US Patent App. 11/007,792.