Tema
Tendências em Matemática Aplicada e Computacional, **14**, N. 3 (2013), 399-414
© 2013 Sociedade Brasileira de Matemática Aplicada e Computacional
www.scielo.br/tema
doi: 10.5540/tema.2013.014.03.0399

Quantum Processes: A Novel Optimization for Quantum Simulation[*]

A.K. MARON[1,**], R.H.S. REISER[2], M.L. PILLA[2] and A.C. YAMIN[2]

Received September 30, 2012 / Accepted September 9, 2013

ABSTRACT. The simulation of quantum algorithms in classical computers demands high processing and storing capabilities. However, optimizations to reduce temporal and spatial complexities are promising and capable of improving the overall performance of simulators. The main contribution of this work consists in designing optimizations to describe quantum transformations using Quantum Processes and Partial Quantum Processes, as conceived in the qGM theoretical model. These processes, when computed on the VPE-qGM execution environment, reduce the execution time of the simulation. The performance evaluation of this proposal was carried out by benchmarks that include sequential simulation of quantum algorithms up to 24 qubits and instances of Grover's Algorithm. The results show improvements in the simulation of general, controled transformations since their execution time was significantly low, even for systems with several qubits. Furthermore, a solution based on GPU computing for dealing with transformations that still have a high simulation cost in the VPE-qGM is also discussed.

Keywords: Quantum Simulation, VPE-qGM, Quantum Processes.

1 INTRODUCTION

Quantum Computing (QC) predicts the development of quantum algorithms that, in various scenarios, are much faster than their classical versions [1, 2]. However, such algorithms can only be efficiently executed on quantum computers, which are currently unavailable for general purpose use. In this context, quantum simulation softwares, such as [3, 4, 5, 6] and [7], were proposed so researchers can anticipate the behaviors of the algorithms when executed on quantum hardware. Despite all the work already done, several approaches for simulation can still be explored.

The *VPE-qGM* (*Visual Programming Environment for the Quantum Geometric Machine Model*) [8] is a quantum simulator under development including both characterizations, visual

[*]Work presented in the XXXIV Congresso Nacional de Matemática Aplicada e Computacional.
[**]Corresponding author: Adriano K. Maron
[1]Graduate Program in Computer Science, University of Pittsburgh, Pittsburgh PA, USA. E-mail: akk48@cs.pitt.edu
[2]Graduate Program in Computer Science, PPGC/UFPEL, Technological Development Center, UFPel – Federal University of Pelotas, 96010-610 Pelotas, RS, Brazil.
E-mails: reiser@inf.ufpel.edu.br; pilla@inf.ufpel.edu.br; adenauer@inf.ufpel.edu.br

modeling, and distributed simulation of quantum algorithms, showing the application and evolution of quantum computing through integrated graphical interfaces. The current focus of this project is related to the exponential growth in the matrices associated to multi-qubit transformations, where the efforts are towards the reduction of the temporal complexities associated to the execution of a muti-qubit quantum transformation.

In this context, the main contribution of this work is the **extension of the *VPE-qGM* simulation capabilities through the implementation of two concepts: Quantum Process (QP) and Quantum Partial Process (QPP)**. According to the specifications of the *qGM* model, these new concepts can be explored for modeling quantum transformations and reducing the computations in a simulation. They are the mathematical structure underling the modeling of quantum parallelism in massive parallel architectures, such as GPUs (*Graphic Processing Units*), as described in [9].

This article is structured as follows: Section 2 comprehends the conceptual background related to this work. The theory and implementation regarding the *QPs* and *QPPs* are presented in Section 3. Section 4 contains the performance analysis of the simulation of *QPs* and *QPPs*. Discussions concerning the results and main contributions of this work are presented in Section 5.

2 PRELIMINARY

Some concepts of QC are necessary to understand the contribution proposed in this work. Thus, an introduction of quantum computing and the qGM (*Quantum Geometric Machine*) model [10] are presented in the following subsections.

2.1 Quantum Computing

In QC, the qubit is the basic information unit, being the simplest quantum system, defined by a unitary and bi-dimensional state vector. Qubits are generally described in Dirac's notation [11], by $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. The coefficients α and β are complex numbers for the amplitudes of the corresponding states in the computational basis (space states). These coefficients must respect the condition $|\alpha|^2 + |\beta|^2 = 1$, which guarantees the unitarity of the state vector of the quantum system represented by $(\alpha, \beta)^t$.

The state space of a quantum system with multiple *qubits* is obtained by the tensor product of the space states of its subsystems. Considering a quantum system with two *qubits*, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\varphi\rangle = \gamma|0\rangle + \delta|1\rangle$, the state space comprehends the tensor product $|\psi\rangle \otimes |\varphi\rangle$, described by $\alpha \cdot \gamma|00\rangle + \alpha \cdot \delta|01\rangle + \beta \cdot \gamma|10\rangle + \beta \cdot \delta|11\rangle$.

Transition states in a N-dimensional quantum system is performed by unitary quantum transformations, defined by square matrices of order N (2^N components since N is the number of *qubits* in the system). The matrix notation of *Hadamard* and its application over a one-qubit system are, respectively, given as

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \text{ and } H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \alpha+\beta \\ \alpha-\beta \end{pmatrix}. \quad (2.1)$$

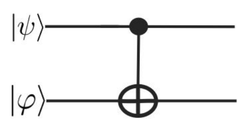
Quantum transformations simultaneously applied to different *qubits* are obtained by the application of the tensor product between the corresponding matrices, as in the following example:

$$H^{\otimes 2} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}. \tag{2.2}$$

Besides the multi-dimensional transformations obtained by the tensor product, controlled transformations can also be used in quantum systems. The *CNOT* transformation acts over two qubits $|\psi\rangle$ and $|\varphi\rangle$, applying the *NOT* (*Pauli X*) transformation to one of them (target qubit), considering the current state of the other (control). Figure 1(a) shows the matrix notation of the *CNOT* transformation and its application to a generic two-qubit quantum state. The corresponding representation (quantum gate) in the quantum circuit model is presented in Figure 1(b).

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \delta \\ \gamma \end{pmatrix}$$

(a) Evolution of quantum state



(b) Quantum circuit

Figure 1: Representations of the CNOT gate.

By the composition and synchronization of quantum transformations, computations exploring the potentialities of quantum parallelism are created. However, the exponential increase of memory usually arises in such computations. As a consequence, there is a loss of performance in the simulation of multidimensional quantum systems. Therefore, optimizations for efficient representation of multi-qubit quantum transformations are necessary.

2.2 qGM Model

The *qGM* model follows the concepts of the domain theory closely related to the Girard's coherent spaces [12]. The objects of the processes domain $\mathbb{D}_\infty$, as introduced in [13] and [14], define coherent sets which provide interpretation for possibly infinite quantum processes. The processes and states are labeled by points in a geometric space, which characterizes the computational basis as an n-dimensional subspace of the Hilbert ($\mathbb{H}$) space.

In the *qGM* model, an elementary process (*EP*) may read data from many memory positions (state space), but can only write in one. For example, in the application $H|\psi\rangle$, described in (2.1), two classical operations are executed. These operations correspond to the computation defined by each component vector of the matrix H and generate the new amplitudes of the state vector. The QP for the H transformation is obtained by the synchronization of two *EPs* associated to each one of these operations. During the simulation, both *EPs* are simultaneously executed, modifying the data in the memory positions. The first memory position, labeled as the state $|0\rangle$

of the computational basis, stores $\frac{\alpha+\beta}{\sqrt{2}}$. Similarly, the second memory position, labeled as $|1\rangle$, receives the $\frac{\alpha-\beta}{\sqrt{2}}$ amplitude. Such execution is performed accordingly to the behavior of the transformation, simulating the evolution of the quantum system.

The interpretation of *QPPs* is obtained from the partial application of a quantum gate due to the existence of uncertainties related to some sets of vectors.

Consider the gate $H^{\otimes 2}$ defined in (2.2). Each single subset in this construction interprets a QPP corresponding to a matrix with only one defined line, and all the others being unknown (indicated by the bottom element $\perp$). Considering as context the elements of the computational basis ($|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$), it is possible to obtain the final global state $|\Phi_1\rangle$ by the union (interpreting the amalgamated sum on the process domain of *qGM* model) of states. By this statement, it is possible to define partial states as in the following matrix-notation:

$$|\Phi_1^{0.x}\rangle_\perp = H_\perp \otimes H|\Phi_0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \perp & \perp \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{-1}{2} \\ \perp \\ \perp \end{pmatrix}$$

$$|\Phi_1^{1.x}\rangle^\perp = H_\perp \otimes H|\Phi_0\rangle = \begin{pmatrix} \perp & \perp \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \perp \\ \perp \\ \frac{1}{2} \\ \frac{-1}{2} \end{pmatrix}$$

Both states $|\Phi_1^{0.x}\rangle_\perp$ and $|\Phi_1^{1.x}\rangle^\perp$ are approximations of $|\Phi_1\rangle = \frac{1}{\sqrt{2}}(1, -1, 1, -1)^t$.

Although it is not the focus of this work, the *qGM* model provides interpretation for other quantum transformations, such as projections for measure operations.

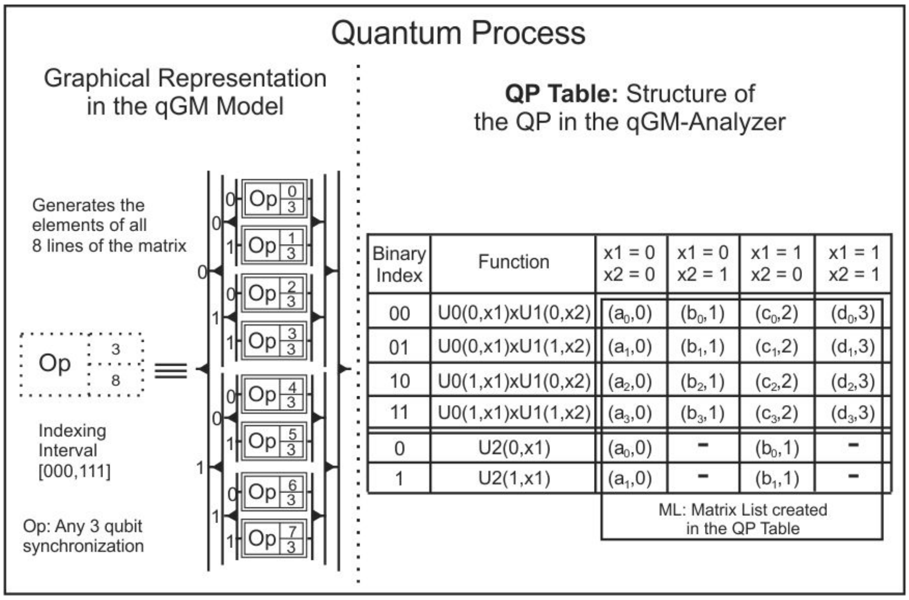
3 QPPS: A PROPOSAL FOR OPTIMIZATION OF QUANTUM SIMULATION IN THE VPE-QGM

The *VPE-qGM* environment is being developed aiming at the support for modeling and distributed simulation of algorithms from *QC*, considering abstraction of the *qGM* model. By following such abstractions, the concept of Quantum Process was implemented in the execution library of the *VPE-qGM*, called *qGM-Analyzer*.

The main extensions consider the representation of controlled and non-controlled transformations, and related possible synchronization. The specifications of these and other new features are described in the following subsections.

3.1 Non-Controlled Quantum Gates

A component QP is able to model a quantum gate. Figure 2 shows a QP associated to a three-dimensional quantum system, including its representation using *EPs* and the structure of such component in the *qGM-Analyzer*.


Quantum Process
Graphical Representation in the qGM Model
QP Table: Structure of the QP in the qGM-Analyzer
Generates the elements of all 8 lines of the matrix
Op
3
8
Indexing Interval [000,111]
Op: Any 3 qubit synchronization
Op 0/3
Op 1/3
Op 2/3
Op 3/3
Op 4/3
Op 5/3
Op 6/3
Op 7/3
ML: Matrix List created in the QP Table

Binary Index	Function	x1 = 0 x2 = 0	x1 = 0 x2 = 1	x1 = 1 x2 = 0	x1 = 1 x2 = 1
00	U0(0,x1)xU1(0,x2)	$(a_0,0)$	$(b_0,1)$	$(c_0,2)$	$(d_0,3)$
01	U0(0,x1)xU1(1,x2)	$(a_1,0)$	$(b_1,1)$	$(c_1,2)$	$(d_1,3)$
10	U0(1,x1)xU1(0,x2)	$(a_2,0)$	$(b_2,1)$	$(c_2,2)$	$(d_2,3)$
11	U0(1,x1)xU1(1,x2)	$(a_3,0)$	$(b_3,1)$	$(c_3,2)$	$(d_3,3)$
0	U2(0,x1)	$(a_0,0)$	-	$(b_0,1)$	-
1	U2(1,x1)	$(a_1,0)$	-	$(b_1,1)$	-

Figure 2: QP and its representation by applying EPs.

In Figure 2, *ML* stores the matrices associated with quantum transformations. Each line in *ML* is generated by functions, which are indicated in the second column of the QP_Table. These functions ($U0$, $U1$ e $U2$) describe the corresponding quantum transformation of the application modeled in the VPE-qGM.

The tuples of each line are obtained by changing the values of the parameters $x1$ and $x2$. The first tuple corresponds to the value obtained by the scalar product between the corresponding functions. The second indicates the column in which the value will be stored.

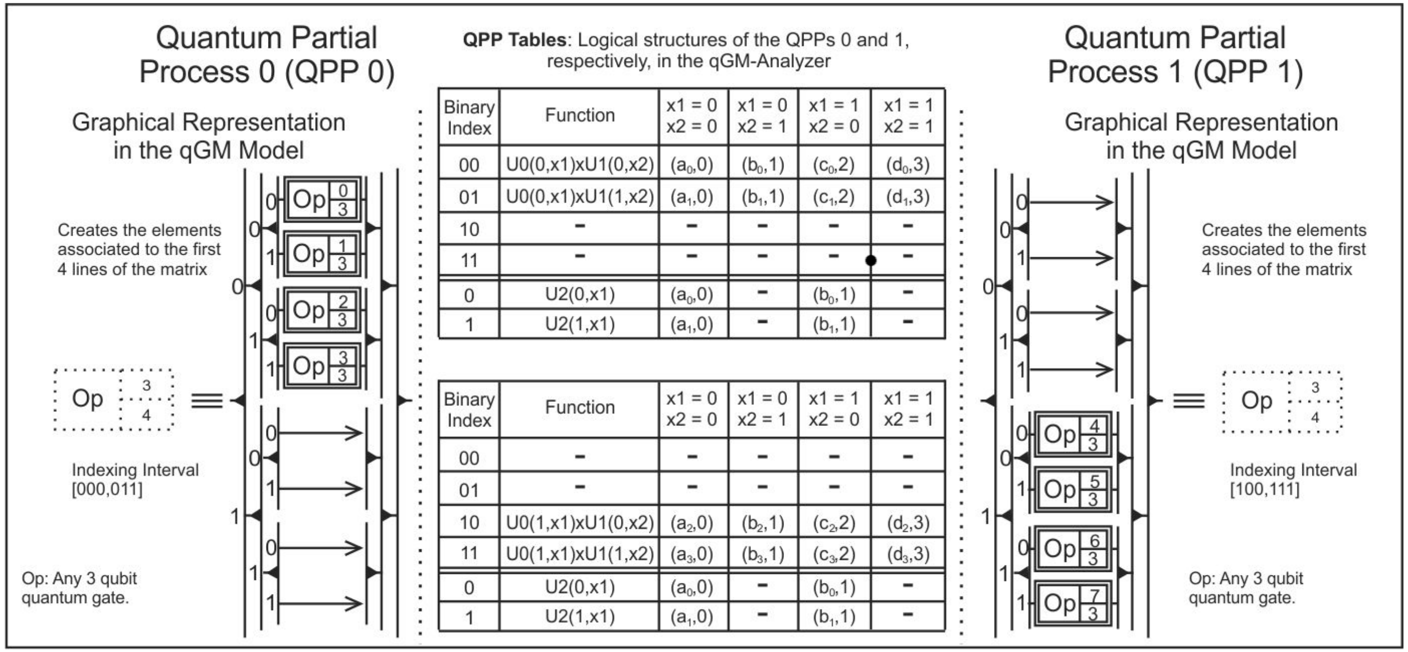
The matrix-order in *ML* is defined from the number of functions (n) grouped together. In Figure 2, the first matrix in *ML*, indicated by M_1, has $n = 2$. Similarly, M_2 has $n = 1$.

It is interesting that the order of each matrix in *ML* can be arbitrarily determined. Although, there is an exponential growth in memory consumption. Hence, a balance between the order and the number of matrices in *ML* ($|ML|$) interferes directly in the performance of an application.

Besides the *ML*, it is necessary to create a list (see in (3.1)) containing auxiliary values for indexing the amplitudes of the state space, which must be multiplied by each value of the matrices in *ML*. In such list, q indicates the total number of qubits in the quantum application.

$$sizesList = \left[2^{q-n}, 2^{q-(2*n)}, \ldots, 2^{q-(|ML|*n)}\right] \tag{3.1}$$

Based on the concept of partial processes defined as partial objects in the *qGM* model, it is possible to split the QP described in Figure 2 in two *QPPs*. Figure 3 contains the description of the QPP_0, which is responsible for the computation of all new amplitudes of the states in the subset of memory positions $M_{QPP_0} = \{0, 1, 2, 3\}$. Similarly, the QPP_1 is responsible for computing the amplitudes in the complement set of M_{QPP_0}, indicated as $M_{QPP_1} = \{4, 5, 6, 7\}$, which is performed independently from the execution of the QPP_0.


Quantum Partial Process 0 (QPP 0)
Graphical Representation in the qGM Model
Creates the elements associated to the first 4 lines of the matrix
Op
Indexing Interval [000,011]
Op: Any 3 qubit quantum gate.
Quantum Partial Process 1 (QPP 1)
Graphical Representation in the qGM Model
Creates the elements associated to the first 4 lines of the matrix
Indexing Interval [100,111]
Op: Any 3 qubit quantum gate.

QPP Tables: Logical structures of the QPPs 0 and 1, respectively, in the qGM-Analyzer

Binary Index	Function	x1 = 0 x2 = 0	x1 = 0 x2 = 1	x1 = 1 x2 = 0	x1 = 1 x2 = 1
00	U0(0,x1)xU1(0,x2)	$(a_0,0)$	$(b_0,1)$	$(c_0,2)$	$(d_0,3)$
01	U0(0,x1)xU1(1,x2)	$(a_1,0)$	$(b_1,1)$	$(c_1,2)$	$(d_1,3)$
10	–	–	–	–	–
11	–	–	–	–	–
0	U2(0,x1)	$(a_0,0)$	–	$(b_0,1)$	–
1	U2(1,x1)	$(a_1,0)$	–	$(b_1,1)$	–

Binary Index	Function	x1 = 0 x2 = 0	x1 = 0 x2 = 1	x1 = 1 x2 = 0	x1 = 1 x2 = 1
00	–	–	–	–	–
01	–	–	–	–	–
10	U0(1,x1)xU1(0,x2)	$(a_2,0)$	$(b_2,1)$	$(c_2,2)$	$(d_2,3)$
11	U0(1,x1)xU1(1,x2)	$(a_3,0)$	$(b_3,1)$	$(c_3,2)$	$(d_3,3)$
0	U2(0,x1)	$(a_0,0)$	–	$(b_0,1)$	–
1	U2(1,x1)	$(a_1,0)$	–	$(b_1,1)$	–

Figure 3: Possible *QPPs* for a 3 qubit gate.

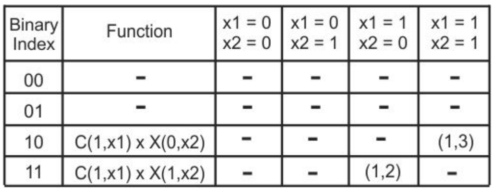
The *QPPs* contribute with the possibility of establishing partial interpretations of a quantum transformation. Complementary *QPPs* (that interpret distinct line sets) can be synchronized and executed independently (in different processing nodes of a multiprocessor system). The bigger the number of *QPPs* synchronized, the smaller is the computation executed by each one, resulting in a low-cost of execution.

3.2 Definition of Controlled Quantum Gates

For **non-controlled** quantum gates, it is possible to model all the evolution of the global state of a quantum system with **a single** QP. However, this possibility can not be applied to controlled quantum gates.

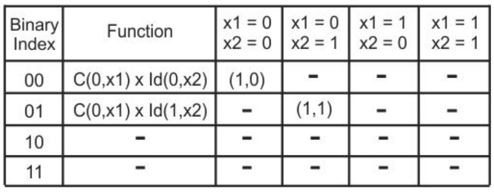
The complete description of *CNOT* transformation is obtained through the expressions in Eq. (3.2), which defines a set of *QPPs*, called *QPP_Set*. *QPPs* for the *CNOT* transformation have their structures illustrated in Figure 4. The QPP_1, in Figure 4(a) and associated to Exp_1, describes the evolution of the states in which the state of the control qubit is $|1\rangle$ (requiring the application of the Pauli *X* transformation to the target qubit). The evolution of the states in which the control qubit is $|0\rangle$ is modeled by Exp_2 generating the QPP_2, illustrated in Figure 4(b). As these states are not modified, the execution of the QPP_2 is not mandatory.

$$Exp_1 = C(1), X \qquad Exp_2 = C(0), Id \tag{3.2}$$



Binary Index	Function	x1 = 0 x2 = 0	x1 = 0 x2 = 1	x1 = 1 x2 = 0	x1 = 1 x2 = 1
00	-	-	-	-	-
01	-	-	-	-	-
10	C(1,x1) x X(0,x2)	-	-	-	(1,3)
11	C(1,x1) x X(1,x2)	-	-	(1,2)	-

(a) QPP_1: change amplitudes



Binary Index	Function	x1 = 0 x2 = 0	x1 = 0 x2 = 1	x1 = 1 x2 = 0	x1 = 1 x2 = 1
00	C(0,x1) x Id(0,x2)	(1,0)	-	-	-
01	C(0,x1) x Id(1,x2)	-	(1,1)	-	-
10	-	-	-	-	-
11	-	-	-	-	-

(b) QPP_2: do not change amplitudes

Figure 4: QPPs for the modeling of the CNOT gate.

In general, $|QPP\_Set| = |Exp| = 2^{nC}$, where nC is the total number of control qubits in all gates applied. However, it is only necessary the creation/execution of the *QPPs* in a subset (*QPP_Subset*) of *QPP_Set*. If only one controlled gate is applied, $|QPP\_Subset| = 1$. When nC controlled qubits are considered in a synchronization of controlled gates, $|QPP\_Subset| = 2^{nC} - 1$.

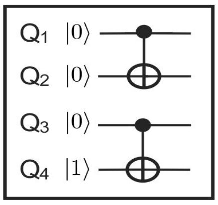
Now, consider the synchronization of *CNOT* transformation, as shown in Figure 5(a). By *VPE-qGM* environment, this configuration is modeled using the expressions in (3.3). Hence, $|QPP\_Set| = 4$. However, the QPP_4, associated to the expression Exp_4, does not change any amplitude and should not be created/executed.

$$\begin{aligned} Exp_1 &= C(1), X, C(1), X & \qquad Exp_2 &= C(1), X, C(0), Id \\ Exp_3 &= C(0), Id, C(1), X & \qquad Exp_4 &= C(0), Id, C(0), Id \end{aligned} \tag{3.3}$$

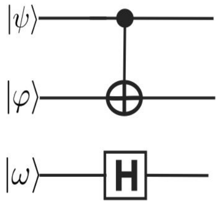
In a synchronization mixing controlled and non-controlled gates (different from *Id*), all the amplitudes are modified. Hence, $QPP\_Subset = QPP\_Set$. The configuration illustrated in the Figure 5(b) is modeled by the expressions in (3.4).

$$Exp_1 = C(1), X, H \quad \text{and} \quad Exp_2 = C(0), Id, H \tag{3.4}$$

Thus, it means that two *QPPs*, identified by QPP_1 and QPP_2, are associated to the expressions in Exp_1 and Exp_2, respectively. However, it is not possible to discard the execution of the QPP_2, once it modifies the amplitudes of some states. Those changes are due to the *H* transformation, which is **always** applied to the last qubit, despite the control state of the *CNOT* transformation.



(a) Two CNOT's



(b) CNOT and H gates

Figure 5: Modeling synchronization of controlled gates.

3.3 Recursive Function

After building the *QPPs*, a recursive operator is applied to the matrices in *ML* for computing the amplitudes of the new global state of the quantum system. This operator dynamically generates all values associated to the resulting matrix obtained by the tensor product of the transformations, defining the quantum application. Besides, a value indexing the amplitude is also generated. The algorithmic description of this procedure with some optimizations is shown in Figure 6.

The execution time of this algorithm grows exponentially when new qubits are added. When analyzing the use of *QPs* and *QPPs* exclusively for the representation of quantum gates, there is a high-cost related to temporal complexity, specially when *Hadamard* gates are applied. Such cost reflects directly in the execution time. However, this approach presents a low-cost related to spatial complexity, once the matrices stored during the execution have maximum size of 32×32.

4 PERFORMANCE ANALYSIS OF THE OPTIMIZATIONS

For validation and performance analysis of the simulation with *QPs* and *QPPs*, the following three study-cases were considered:

C1: Reversible circuit benchmarks from [15];

C2: *Hadamard* gates up to 14 qubits;

C3: Instances of Grover's algorithm up to 14 qubits.

```
if $matrixIndex = numMatrices - 1$ then
    for $l = 0$ to $size(Matrices[matrixIndex])$ do
        $res \leftarrow 0$;
        $line \leftarrow Matrices[matrixIndex][l][0]$;
        $linePos \leftarrow Matrices[matrixIndex][l][0]$;
        for $column = 0$ to $size(line)$ do
            $pos \leftarrow basePos + line[column][1]$;
            $res \leftarrow res + (partialValue \times line[column][0] \times memory[1][pos])$;
        end for
        $writePos \leftarrow memPos + (linePos \times sizesList[matrixIndex])$;
        $res \leftarrow res + memory[0][writePos]$;
        $memory[0][writePos] \leftarrow res$;
    end for
else
    for $l = 0$ to $size(Matrices[matrixIndex])$ do
        $line \leftarrow Matrices[matrixIndex][l][1]$;
        $linePos \leftarrow Matrices[matrixIndex][l][0]$;
        for $column = 0$ to $size(line)$ do
            $next\_basePos \leftarrow basePos + (line[column][1] \times sizesList[matrixIndex])$;
            $next\_partialValue \leftarrow partialValue \times line[column][0]$;
            $ApplyValues(Matrices, numMatrices, sizesList, memory,$
            $next\_partialValue, matrixIndex + 1, next\_basePos,$
            $memPos + (linePos \times sizesList[matrixIndex]))$;
        end for
    end for
end if
```

Figure 6: Algorithm for the execution of the computations over QPs and QPPs.

The evaluation with benchmarks $C1$ and $C2$ considered 10 simulations for each study-case, and the average of execution time and memory consumption were measured. For $C3$, only one execution of each instance of Grover's algorithm was performed, however each step of the simulation was monitored and therefore several samples of the simulation time for each step were collected. From those, the average for the simulation time associated with each step was obtained. The hardware considered for each scenario is characterized as follows:

C1 and **C2:** Core i5-2410M, 4 GB RAM, Python 2.7 and Ubuntu 11.10 64 bits;

C3: Core i7-3770, 8 GB RAM, Python 2.7 and Ubuntu 12.04 64 bits.

4.1 Reversible Circuits and Hadamard Gates

Execution time and memory usage were monitored. The main performance comparison was made against the previous version of the *qGM-Analyzer*, which supports the simulation of quantum algorithms using *EPs*, considering the optimizations described in [16]. The main features of each algorithm and the results obtained are presented in Tables 1 and 2.

Table 1: Quantum algorithms simulated using QPPs.

Algorithm	Qubits	Gates	Simulation w/QPPs		Simulation w/EPs	
			Time (s)	Mem (MB)	Time (s)	Mem (MB)
9symd2	12	28	0.400	12	38.805	12
$gf2^4$	12	19	0.236	12	26.352	12
$gf2^5$	15	29	1.328	13	372.488	13
$gf2^6$	18	41	11.264	24	5081.553	22
$gf2^7$	21	55	135.013	92	NS[1]	NS
$gf2^8$	24	85	1532.015	524	NS	NS
ham15_1	15	132	6.872	13	1778.170	13
ham15_2	15	70	4.708	13	925.319	13
ham15_3	15	109	8.436	13	1400.632	13
mod1024adder	20	55	44.099	60	NS	NS
rc_adder	16	19	2.358	15	549.079	14

NS: Not Supported. Simulation time over 4 hours.

Table 2: Quantum algorithms simulated using QPs.

Algorithm	Qubits	Gates	Simulation w/QPs		Simulation w/EPs	
			Time (s)	Mem (MB)	Time (s)	Mem (MB)
$H^{\otimes 11}$	11	11	6.816	12	7.882	12
$H^{\otimes 12}$	12	12	25.292	12	28.281	12
$H^{\otimes 13}$	13	13	97.401	12	111.572	12
$H^{\otimes 14}$	14	14	348.923	12	496.934	12

Quantum algorithms up to 24 qubits were simulated. The memory consumption was higher for the new proposal due to slightly more complex structures necessary to represent the new components. However, the trade off between memory usage and execution time is positive for the new approach.

As the optimizations regarding *QPs* and *QPPs* only affect quantum gates, the high memory cost is due to the storage of the amplitudes of the quantum system. This structure limits the *VPE-qGM* to the simulation of algorithms with approximately 25 qubits in a $4GB\ RAM$ machine. The improvement for controlled operations is due to the optimization focused on the identification of *QPPs* that change amplitudes in the spaces state, being different of the *EPs*, which recomputes even the amplitudes that remain unchanged. For the *Hadamard* gates, an inferior improvement was obtained since all the amplitudes are modified. However, the reduction of 29% in simulation time was due to the generation of all elements in the same QP, reducing the simulation overhead.

Despite the different sizes of the state vectors for $H^{\otimes 11}$, $H^{\otimes 12}$, $H^{\otimes 13}$ and $H^{\otimes 14}$, the memory usage remained the same (12 MB). This behavior is explained by the memory management of the Python interpreter. For all Python processes, an initial 12 MB memory space is allocated, even if the process execution requires less. As the data referred to memory usage presented in the Tables 1 and 2 was obtained using the *top* software, only the total amount of memory allocated to the process was exhibited instead of the actual space occupied by allocation calls. For algorithms with more than 12 qubits, which generate bigger state vectors, the Python interpreter dynamically allocates more memory when necessary.

4.2 Grover's Algorithm Simulation

The simulation of the Grover's algorithm follows the circuit described in [17]. Herein, the focus is in the simulation time since the memory consumption follows the values previously presented in Table 2.

Table 3 describes the number of iterations of the Grover's (G) operator, the total number of simulation steps generated and the total simulation time. The G operator is comprised by:

- U_f: oracle that applies a controlled transformation to all qubits;
- $2|\psi\rangle\langle\psi| - I$: amplitude amplification operator with five steps as defined in [17].

Table 3: Grover's Algorithm Simulation.

Algorithm	Iterations	Number of Steps	Simulation Time (s)
Grover 10 qubits	17	103	21.635
Grover 11 qubits	25	151	116.226
Grover 12 qubits	35	211	630.203
Grover 13 qubits	50	301	3437.802
Grover 14 qubits	71	427	20914.590

The highest standard deviation for these simulations was 0.29% of the average, measured for the *Grover 10 qubits*. As it can be seen, due to the many steps necessary, the simulation time suffers an exponential increase when systems with more qubits are simulated.

A more detailed analysis of each step of the Grover's algorithm reveals what generates this exponential increase. Figure 7 presents the amount of computation time required by each step of Grover's algorithm. The *Initialization* step, comprised by Hadamard gates, accounts for at most 5% of the total simulation time, for the system with 12 qubits. As stated in the previous sections, the Hadamard gate has the highest computational cost in the VPE-qGM. However, in this case the influence over the total simulation time is not higher since the *Initialization* step is executed only once.

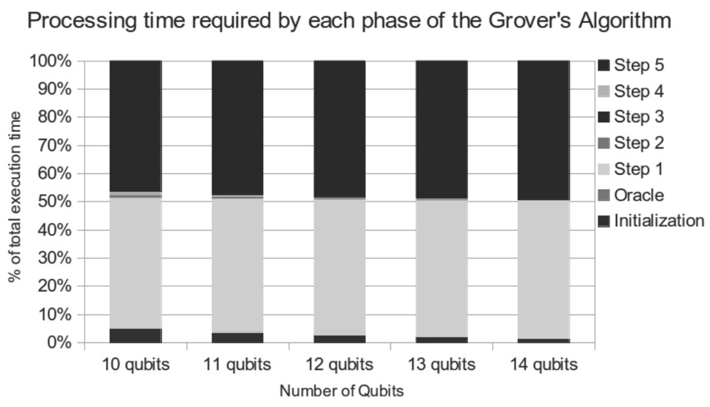


Figure 7: Computation time required by each step of Grover's algorithm

The *Oracle* is applied at each iteration of G, but it does not significantly affect the total time due to partial execution with *QPPs*.

In particular, one can observe that:

(i) Steps from 1 to 5 refer to the amplitude amplification operator;

(ii) Steps 2 and 4 yield a more significant computation than controlled operators, since in the global context of the simulation they must be fully executed;

(iii) Step 3 is slightly more complex than the *Oracle*, but as it is also described in terms of *QPPs*, an efficient execution is obtained;

(iv) Steps 1 and 5 account for the greatest shares of the total simulation time. Being described by many Hadamard gates and executed at each iteration of the G operator, they lead to a scenario where a transformation defined by $H^{\otimes 13} \otimes Id$ is applied 854 times during the simulation (427 times for each step).

4.3 Related Work Results

The state-of-art in sequential simulation of quantum algorithms, characterized by the works of [7] and [6], represents the best performance reference for our work. As we do not have access to these simulators yet, simulations of these alternative software in our own hardware was not performed and therefore a direct performance comparison is not possible. However, it is possible to perform an approximated performance comparison of the *VPE-qGM* with the *QuIDDPro* simulator according to the results presented in [7]. The simulated circuits include the following: $ham15\_1$, $ham15\_2$, $ham15\_3$, rc_adder, and $9symd2$. It is important to note that those results were obtained using a different hardware configuration (Athlon 1.2 GHz processor with 1GB of RAM).

The simulation time and memory usage in the *QuIDDPro* simulator, described in [7], are better than those obtained in the *VPE-qGM*. Such results can be justified due two characteristics of the *VPE-qGM*:

- The former is related to Python language, which is interpreted and, consequently, slower than C, used in the *QuIDDPro*;
- The latter consists in the absence of optimizations for the storage of the state space in the *VPE-qGM*, which is responsible for the high memory usage.

The results presented in [6] included some stabilizer circuits and factorization algorithms, which yet are not fully supported in the *VPE-qGM* environment. Thus, no comparison with *PVLIB* was performed.

4.4 Expectations for GPU Computing

As discussed in Section 4.2, the simulation time of Grover's algorithm is highly affected by the *Hadamard* gates applied during the amplitude amplification operator. As our solution does not consider gate-by-gate simulation, currently the problem of high computational cost is being treated by the highly parallel architecture of GPUs.

Our first results in this regard, presented in [9], comprehends the simulation of *Hadamard* gates up to 21 qubits. Since this implementation is in its initial stages, algorithms such as Grover are not supported yet. However, as its computational cost is directly affected by the Hadamard gates, it is possible to estimate how it should perform on the parallel simulation on the GPU.

Table 4 shows results from simulations of Hadamard gates using the Python sequential approach, presented in this work, and a GPU simulation described in [9]. The speedups reflect the efficiency of the parallel simulation and provide an approximation of the simulation time for an instance of the Grover's algorithm. As $\approx$ 97% of the Grover's simulation time is spent on execution of Hadamard gates, it is feasible to state that a similar speedup may be obtained in the Grover's parallel simulation in the GPU.

Table 4: Simulations of Hadamard gates with sequential and parallel approaches.

Transformation	Sequential (s)	GPU (s)	Speedup
$H^{\otimes 10}$	1.142	0.001	1142
$H^{\otimes 11}$	4.383	0.001	4383
$H^{\otimes 12}$	17.402	0.008	2175
$H^{\otimes 13}$	68.956	0.020	3447
$H^{\otimes 14}$	285.241	0.085	3355
$H^{\otimes 15}$	1140.114	0.299	3813

5 CONCLUSION

The *VPE-qGM* environment introduces a novel approach for the simulation of quantum algorithms in classical computers. Besides the availability of graphical interfaces for modeling and simulation of the algorithms, this environment supports the simulation of algorithms up to 24 qubits. This limit is established by the memory consumption due to the storage of the state vector of the algorithm.

For *Hadamard* gates, the limitation is related to the exponential growth in the simulation time. In our current hardware, the limit for *Hadamard* gates is appropriately 16 qubits. The simulation of controlled quantum transformations has the benefit of a reduced number of operations in order to simulate state evolution. Hence, such simulation up to 24 qubits is possible.

Considering the state-of-art in quantum simulation, even after the optimizations described in this work, the best simulators available still outperforms the *VPE-qGM*. However, new improvements can be developed in the *VPE-qGM* to handle memory consumption and execution time. By exploring the visual tools provided by the *VPE-qGM* and its integration with optimized libraries, this environment becomes an intuitive platform for the development and study of quantum algorithms.

With the recent development of *GPUs*, several research areas are working with massive amounts of data and dealing with heavy calculations. The exploration of this approach in benefit of QC is a novel research field and by optimizing algorithms and the computing power of *GPUs*, new breakthroughs can be achieved.

The contribution of this work is the first of three steps towards a solution for quantum simulation not yet consolidated: the use of HPC's (High Performance Computing) resources, such as *GPUs* and clusters, coupled with optimizations that are capable of exploring patterns and mathematical properties intrinsic to quantum computing. Steps two and three of our project are respectively comprised by: (i) support for GPU/cluster simulation; and (ii) optimizations for efficient representation/storage of the state vector.

ACKNOWLEDGMENTS

This work is supported by the Brazilian funding agencies CAPES, FAPERGS (PqG 06/2010, process number 11/1520-1), and by the CNPq/PRONEX/FAPERGS Green Grid project.

RESUMO. A simulação de algoritmos quânticos em computadores clássicos exige alta capacidade de processamento e armazenamento. Entretanto, otimizações voltadas à redução das complexidades espacial e temporal são promissoras e capazes de melhorar o desempenho dos simuladores. A principal contribuição deste trabalho consiste no desenvolvimento de otimizações para descrição de transformações quânticas utilizando Processos Quânticos e Processos Quânticos Parciais, seguindo as concepções do modelo teórico qGM. Esses processos, quando computados no ambiente de execução VPE-qGM, reduzem o tempo de execução das simulações. A avaliação de performance desta proposta foi efetuada utilizando benchmarks que incluem a simulação sequencial de algoritmos quânticos com até 24 qubits e instâncias do Algoritmo de Grover. Os resultados mostram uma melhora na simulação de transformações básicas e controladas, dado que seus correspondentes tempo de execução foram significantemente reduzidos, mesmo quando utilizados sistemas com muitos qubits. Ainda, uma solução baseada em GPUs voltada à transformações que ainda possuem alto custo de simulação no VPE-qGM é discutida.

Palavras-chave: Simulação Quântica, VPE-qGM, Processos Quânticos.

REFERENCES

[1] L. Grover. "A fast quantum mechanical algorithm for database search", *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 212–219, 1996, available at <http://doi.acm.org/10.1145/237814.237866> (dec.2011).

[2] P. Shor. "Polynomial-time algoritms for prime factorization and discrete logarithms on a quantum computer". *SIAM Journal on Computing* (1997).

[3] J. Agudelo & W. Carnielli. "Paraconsistent machines and their relation to quantum computing". *J. Log. Comput.*, **20**(2) (2010), 573–595.

[4] A. Barbosa. "Um simulador simbólico de circuitos quânticos". Master's thesis, Universidade Federal de Campina Grande (2007).

[5] H. Watanabe. "Qcad: Gui environment for quantum computer simulator", 2002, available at <http://apollon.cc.u-tokyo.ac.jp/watanabe/qcad/> (dec.2011).

[6] V. Samoladas. "Improved bdd algorithms for the simulation of quantum circuits", in *Annual European Symposium on Algorithms*. Berlin, Heidelberg: Springer-Verlag, (2008), 720–731.

[7] G. Viamontes. "Efficient quantum circuit simulation". Phd Thesis, The University of Michigan, (2007).

[8] A. Maron, A. Pinheiro, R. Reiser & M. Pilla. "Consolidando uma infraestrutura para simulação quântica distribuída", in *Anais da ERAD 2011*. SBC/Instituto de Informática UFRGS, (2011), 213–216.

[9] A. Maron, R.H.S. Reiser & M.L. Pilla. "High-performance quantum computing simulation for the quantum geometric machine model", in *CCGRID 2013 IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. NY: IEEE Conference Publishing Services, May 2013, pp. 1–8.

[10] R. Reiser & R. Amaral. "The quantum states space in the qgm model", in *Anais/III WECIQ*. Petrópolis/RJ: Editora do LNCC, (2010), 92–101.

[11] M.A. Nielsen & I.L. Chuang. *Computação Quântica e Informação Quântica*. Bookman (2003).

[12] J.-Y. Girard. "Between logic and quantic: a tract", in *Linear logic in computer science*, P.R. Thomas Ehrhard, Jean-Yves Girard and P. Scott, Eds. Cambridge University Press, 2004, pp. 466–471. [Online]. Available: http://iml.univ-mrs.fr/∼girard/Articles.html

[13] R. Reiser, R. Amaral & A. Costa. "Quantum computing: Computation in coherence spaces", in *Proceedings of WECIQ 2007*. UFCG – Universidade Federal de Campina Grande, (2007), 1–10.

[14] R. Reiser, R. Amaral & A. Costa. "Leading to quantum semantic interpretations based on coherence spaces", in *NaNoBio 2007*. Lab. de II – ICA/DDE – PUC-RJ, (2007), 1–6.

[15] G.D.D. Maslov & N. Scott. "Reversible logic synthesis benchmarks page", 2011, available at <http://www.cs.uvic.ca/dmaslov> (apr.2012).

[16] A. Maron, A. Ávila, R. Reiser & M. Pilla. "Introduzindo uma nova abordagem para simulação quântica com baixa complexidade espacial", in *Anais do DINCON 2011*. SBMAC, (2011), 1–6.

[17] A. Prokopenya. "Wolfram demonstrations project – quantum circuit implementing grover's search algorithm", 2009, http://demonstrations.wolfram.com/QuantumCircuitImplementingGroversSearch Algorithm/ (may 2013).