Computer Science Journal of Moldova, vol.**23**, no.2(68), 2015

Solving Problem of Graph Isomorphism by Membrane-Quantum Hybrid Model

Artiom Alhazov Lyudmila Burtseva Svetlana Cojocaru
Alexandru Colesnicov Ludmila Malahov

Abstract

This work presents the application of new parallelization methods based on membrane-quantum hybrid computing to graph isomorphism problem solving. Applied membrane-quantum hybrid computational model was developed by authors. Massive parallelism of unconventional computing is used to implement classic brute force algorithm efficiently. This approach does not suppose any restrictions of considered graphs types. The estimated performance of the model is less then quadratic that makes a very good result for the problem of **NP** complexity.

1 Introduction

The present paper concerns application of new computational models based on hybrid of bio-inspired and quantum approaches. In computability theory, a model defines feasible computational operations with their execution time/space. There are many branches of bio-computation: evolution, DNA, swarm, etc. We took as our base the membrane computing formalism, also known as P systems [4].

A P system is a set of mutually inclusive membranes that contain multisets of objects (numbers, strings, or some abstract items) and evolve under some rules. All possible rules are applied in parallel to all possible membranes and objects. This is the membrane parallelism that makes this computation model very powerful.

In our model, membranes can additionally contain quantum machines that perform quantum computations (Fig. 1).

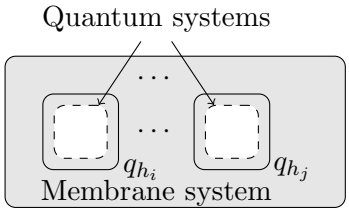

Figure 1. Structure of the hybrid model

The idea of such hybrid computation arises from needs of different domains delivering hard tasks, which are not always satisfactorily solved by existing high performance computational models.

To illustrate the proposed model, we present in this paper a solution of the graph isomorphism problem (GI). Due to its practical applications ranging from chemistry to social sciences, this problem has been solved by many algorithms, both classical and unconventional, still remaining under investigation. Unlike problems which are usually considered as suitable for unconventional computation, GI belongs to **NP**, but is not known to belong to its well-studied subsets like **P** or **NP**-complete. The best classical algorithm complexity is $O(c^{\sqrt{n}\log n})$, where n is the number of vertices in the graph.

Because of this uncertainty of general task, it is divided into several subtasks according to graph types (trees, planar graphs, poor-connected, etc.). The majority of mentioned subproblems are proved to be in the integer factorization class. So, existence of polynomial-time quantum algorithm for integer factoring makes GI a good candidate for speedup by a quantum computing [1]. Further developments of quantum computing solutions of GI mostly applied the quantum walk [6]. However, all issues attributable for quantum computation such as "probability" results or exponential growing of system size affect the proposed solutions.

As we said above, we choose the membrane computing formalism. The proposed hybrid model is the usual P system framework supplied by capability to perform quantum computations in its membranes.

Membranes, which are supposed to obtain quantum functionality, just have the specific marks in the P system description. In the marked membrane, the apparition of some specific objects (quantum data, or quantum triggers) starts quantum computation. Specified data become the initial state of the quantum registers. After finishing the quantum computation produces other specific objects (quantum results) in the external membrane.

To provide incorporated quantum functionality in the proposed hybrid model, standard scheme of quantum device [7] proves itself to be sufficient.

Both for membrane and quantum part, we use particular P system formalisms and quantum gates in dependence of the solved problem.

In this paper, the hybrid model solving GI is constructed over *decision P system with active membranes* implementing brute force algorithm. The quantum devices perform only comparison using CNOT, NOT and Toffoli gates.

2 Hybrid Computational Model

2.1 Membrane Subsystem

Membrane systems, or **P systems,** consist of a set of mutually inclusive membranes. The membrane structure μ is a rooted tree, traditionally represented by bracketed expression. For example, $[\ [\ [\ ]_4\ ]_2\ [\ ]_3\ ]_1$ denotes membranes 2 and 3 inside membrane 1, and membrane 4 inside membrane 2. There are many different variants of P systems. Some variants may use static membrane structure, others change it during calculations.

Membranes contain multisets of objects. Objects are numbers, strings, or some abstract items. Different operations over objects may be available. The initial state of a P system is always provided. The initial state is some membrane structure with some multisets inside.

The evolution of a P system is governed by a set of rules. Rules are applicable under certain conditions to change the objects in membranes. All possible rules are applied in parallel to all possible mem-

branes and objects (membrane parallelism). The calculation stops when no rules can be applied.

We will use a decision P system with active membranes. *Decision* means that the alphabet of objects contains symbols `yes` and `no` that represent two possible results of calculation. *P system with active membranes* is defined as a tuple:

$$\Pi = \{O, E, \mu, w_1, \cdots, w_m, e_1, \cdots, e_m, R\}.$$

Here O is the alphabet of objects. $E = \{0, 1, ..., k\}$ is a set of membrane electrical charges, or polarizations. μ is a membrane structure of m membranes labeled by integers; we will denote $H = \{1, \ldots, m\}$ a set of membrane labels. $w_i \in O^*$ and $e_i \in E$, $i \in H$, represent initial content and initial polarization of the i-th membrane. Strings w_i over alphabet O (possibly empty) represent multisets of objects from O. R is a set of rules of the form:

(a) $[a \to v]^i_h$, $a \in O$, $v \in O$, $h \in H$, $i \in E$ (evolution rules, used in parallel in the region of the h-th membrane, provided that the polarization of the membrane is i);

(b) $a[]^i_h \to [b]^j_h$, $a, b \in O$, $h \in H$, $i, j \in E$ (communication rules, sending an object into a membrane and possibly changing the polarization of the membrane);

(c) $[a]^i_h \to []^j_h b$, $a, b \in O$, $h \in H$, $i, j \in E$ (communication rules, sending an object out of a membrane, possibly changing the polarization of the membrane);

(d) $[a]^i_h \to b$, $a, b \in O$, $h \in H$, $i \in E$ (membrane dissolution rules; in reaction with an object, the membrane is dissolved);

(e) $[a]^i_h \to [b]^j_h [c]^k_h$, $a, b, c \in O$, $h \in H$, $i, j, k \in E$ (division rules for elementary membranes not containing other membranes inside; in reaction with an object, the membrane is divided into two membranes with the same label, possibly of different polarizations, and the object specified in the rule is replaced in the two new membranes by possibly new objects).

2.2 Quantum Subsystem

The investigated hybrid model supposes additionally that in any membrane the apparition of some specific objects (quantum data, or quantum triggers) starts a quantum calculation. The said data are available as initial state of the quantum registers. After its termination the quantum calculation produces another specific objects (quantum results) inside the membrane. From the P system point of view, the quantum calculation is a step of the membrane calculation.

Quantum device. We suppose a standard quantum device available for quantum calculations. The quantum device contains qubits organized in quantum registers. It works in three steps: non-quantum (classical) initialization of qubits when they are set in base states; quantum transformation when the qubits are non-observable; non-quantum (classical) measurement that produces the observable result.

Several restrictions are imposed over the quantum device. Each qubit contains 0, or 1, or (during quantum calculation) superposition of both. Therefore, the initial data and the result may be regarded as non-negative integers in binary notation. The quantum transformation is linear and reversible. The general rule is that arguments and results are kept in different quantum registers. Another general condition is that the ancillary qubits were not entangled with the argument and the result after the calculation.

The construction of a quantum computer shown in Fig. 2 guarantees this.

3 Interface between Membrane and Quantum Sub-systems

Communications between membrane and quantum sub-systems are performed through input/output signals and triggering (Fig. 3).

We define the hybrid system formally as a tuple

$$\beta = (\Pi, T, T', H_Q, Q_N, Q_M, Inp, Outp, t, q_{h_1}, \cdots, q_{h_m}).$$

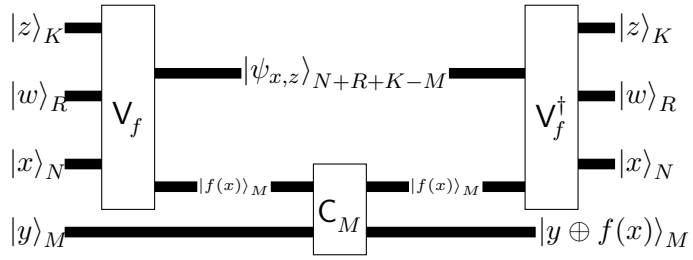

Figure 2. Quantum calculation; initialization and measurement are not shown

Here, Π is a P system, and $H_Q = \{h_1, \cdots, h_m\}$ is a subset of membrane labels in Π used for quantum calculations. T is a trigger and T' is the signal on obtaining the quantum result. Sub-systems $q_{h_1}, \cdots, q_{h_m}$ are the quantum sub-systems associated to the corresponding membranes from H_Q. The rest of the components of the tuple β specify the interaction between Π and q_{h_j}, $1 \leq j \leq m$ (Fig. 3).

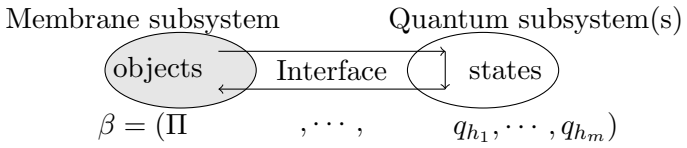

Figure 3. Subsystems and interface in the hybrid model

For simplicity, we assume that the running time of quantum subsystems of the same type is always the same. To keep this time general, we include a timing function $t : H_Q \to \mathbb{N}$: the quantum computation in a sub-system of type q_{h_j} takes $t(h_j)$ membrane steps. It is an open general question how to calculate the timing of quantum calculation with respect to the timing of membrane calculation. We could use as the first rough estimation that quantum calculation takes three steps of membrane calculation (initialization, quantum transformation, measurement).

The input size (in qubits) for quantum systems is given by Q_N :

$H_Q \to \mathbb{N}$. The output size (in bits) for quantum systems is given by $Q_M : H_Q \to \mathbb{N}$.

We would like to define the behavior of β in all possible situations, so we introduce the trigger $T \in O$, where O is the alphabet of Π. The work of a quantum sub-system of type q_{h_j} starts whenever T appears inside the corresponding membrane. Note that we said that q_{h_j} is a type of a quantum sub-system, because in general there may be multiple membranes with label h_j containing quantum sub-systems with the same functionality. The quantum state is initialized by objects from $Inp(h_j) = \{O_{k,h_j,b} \mid 1 \leq k \leq Q_N(h_j),\ b \in \{0,1\}\} \cup \{T\}$, so $Inp : H_Q \to 2^O$ is a function describing the input sub-alphabet for each type of quantum sub-system, the meaning of object $O_{k,h_j,b}$ being to initialize bit k of input by value b. We require that the set of rules satisfies the following condition: any object that may be sent into a membrane labeled h_j must be in $Inp(h_j)$.

The output of quantum sub-systems is returned to the membrane system in the form of objects from $Outp(h_j) = \{R_{k,h_j,b} \mid 1 \leq k \leq Q_M(h_j),\ b \in \{0,1\}\} \cup \{T'\}$, the meaning of object $R_{k,h_j,b}$ being that the output bit k has value b. In case of one-bit output, we often denote it `yes` and `no`.

The result of a quantum sub-system may be produced in the membrane together with object T'.

There are two possibilities to synchronize quantum and membrane levels. We can use a timing function and to wait for the quantum result by organizing the corresponding delay in membrane calculations, or we can wait for appearance of the resulting objects, or the trigger T'. For generality, our model provides both possibilities. The topic needs further investigations.

4 Graphs Isomorphism Problem

GI requires to decide whether two given graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are actually the same graph with relabeling of the vertices.

4.1 Graph Isomorphism: Hybrid Computation

The first graph is represented by objects $a^{(c)}_{i,j,0,0,0}$, where $c = 1$ if the graph has edge (i, j), and $c = 0$ if it does not, $0 \le i \le n-1$, $0 \le j \le n-1$. The second graph is similarly represented by objects $b^{(c)}_{i,j,0}$. Let $N = \lceil \log_2 n \rceil$ (hence, $n \le 2^N < 2n$). We construct the following hybrid system.

$$\begin{array}{rcl}
\beta & = & (\Pi, H_q = \{2\}, n, Inp, Outp = \{\texttt{yes}, \texttt{no}\}, q_2), \text{ where} \\
Inp & = & \{I_{k,b} \mid 0 \le k \le 2n^2 - 1,\ 0 \le b \le 1\}, \\
q_2 & & \text{is a quantum system comparing the first } n^2 \text{ bits with the} \\
 & & n^2 \text{ second bits in } 2N+1 \text{ steps, described later, and} \\
\Pi & = & (O, \Sigma, \mu = [\ [\ ]^0_2\ ]^0_1, w_1, w_2, R, 1)
\end{array}$$

is a decisional P system with active membranes, where

$$\begin{array}{rcl}
\Sigma & = & \{a^{(c)}_{i,j,0,0,0}, b^{(c)}_{i,j,0} \mid 0 \le i \le n-1,\ 0 \le j \le n-1, c \in \{0,1\}\}, \\
O & = & \{d_i \mid 1 \le i \le nN\} \cup \{p_i \mid 1 \le u \le (n+2)N+5\} \\
 & \cup & \{x_{i,t,k,s} \mid 0 \le i < n,\ 0 \le t < n,\ 0 \le k \le N, \\
 & & 0 \le s \le \max(2^{k-1}, 0)\} \cup \{\texttt{yes}, \texttt{no}, X\} \\
 & \cup & \{a^{(c)}_{i,j,t,k,s} \mid -2^N \le i < n,\ -2^N \le j < n,\ 0 \le t \le n, \\
 & & 0 \le k \le N,\ 0 \le s \le \max(2^{k-1}-1, 0),\ c \in \{0,1\}\} \cup Inp \\
 & \cup & \{b^{(c)}_{i,j,t} \mid 0 \le i < n,\ 0 \le j < n,\ 0 \le t \le nN+1,\ c \in \{0,1\}\}, \\
w_1 & = & p_1,\ w_2 = d_1 x_{0,0,0,0} \cdots x_{n-1,0,0,0},
\end{array}$$

and the set R is the union of the following rule groups (together with their explanations): generation, checking, processing the input, and result. Note that the all four groups start working in parallel.

Generation

$$\begin{array}{ll}
1: & [\ d_i\ ]^e_2 \to [\ d_{i+1}\ ]^0_2\ [\ d_{i+1}\ ]^1_2,\ e \in \{0,1\},\ 1 \le i \le nN, \\
2: & [\ d_{nN+1}\ ]^e_2 \to [\ ]^0_2 d_{nN+1},\ e \in \{0,1\},
\end{array}$$

1: creating 2^{nN} membranes and generating for each of them the corresponding nN bits defining the permutations candidates. Other

objects may check these bits as membrane polarizations during nN steps (not considering the initial step, where the polarization was 0).

2: After the generation phase, set the polarization to 0.

Checking

3 : $[\ x_{i,t,k,s} \to x_{i,t,k+1,2s+e}\ ]_2^e,\ 0 \le i < n,\ e \in \{0,1\},$
$0 \le t < n,\ 0 \le k \le N-1,\ 0 \le s \le \max(2^{k-1}-1, 0),$

4 : $[\ x_{2s+e,t,N,s} \to \lambda\ ]_2^e,$
$0 \le s \le 2^{N-1}-1,\ 0 \le t < n,\ e \in \{0,1\},$

5 : $[\ x_{i,t,N,s} \to x_{i,t+1,1,0}\ ]_2^e,\ 0 \le i < n,$
$0 \le t < n-1,\ 0 \le s \le 2^{N-1}-1,\ e \in \{0,1\},\ i \ne 2s+e,$

6 : $[\ x_{i,n-1,N,s} \to X\ ]_2^e,\ 0 \le i < n,$
$0 \le s \le 2^{N-1}-1,\ e \in \{0,1\},\ i \ne 2s+e,$

7 : $[\ X\ ]_2^0 \to [\ \ ]_2^1 X,$

3: Compute the value $\sigma(t)$ of permutation σ for node label t.

4: Erase the label $\sigma(t)$.

5: Continue matching the label.

6: Rename unmatched node labels into X (to make the next step deterministic).

7: Invalid permutation detected. Cancel isomorphism check by setting polarization to 1.

Processing the input

8 : $[\ a^{(c)}_{i,j,t,k,s} \to a^{(c)}_{i,j,t,k+1,2s+e}\ ]_2^e,$
$-2^N \le i < n,\ -2^N \le j < n,\ 0 \le t < n,\ 0 \le k \le N-1,$
$c \in \{0,1\},\ 0 \le s \le \max(2^{k-1}-1, 0),$

9 : $[\ a^{(c)}_{i,j,t,N,s} \to a^{(c)}_{i',j',t+1,1,0}\ ]_2^e,\ -2^N \le i < n,\ -2^N \le j < n,$
$0 \le t < n,\ c \in \{0,1\},\ 0 \le s \le 2^{N-1}-1,$
$i' = -2s-e-1,\ i = t,\ i' = i$ otherwise,
$j' = -2s-e-1,\ j = t,\ j' = j$ otherwise,

10 : $[\ a^{(c)}_{-i-1,-j-1,n,1,0} \to I_{ni+j,c}\ ]_2^0,$
$0 \le i < n,\ 0 \le j < n,\ c \in \{0,1\},$

11 : $[\ b^{(c)}_{i,j,t} \to b^{(c)}_{i,j,t+1}\ ]_2^e,\ 0 \le i < n,\ 0 \le j < n,$
$0 \le t \le nN,\ e \in \{0,1\},\ c \in \{0,1\},$

12 : $[\ b^{(c)}_{i,j,nN+1} \to I_{n^2+ni+j,c}\ ]_2^0,$
$0 \le i < n,\ 0 \le j < n,\ c \in \{0,1\},$

8: Compute $\sigma(t)$ for matrix elements.

9: Perform row/column substitution if row/column is t. If so, store the result as a negative index, minus one. In either case, proceed with the next node.

11: The input symbols for the second graph wait while the permutations for the first graph are being generated.

10,12: Initialize the quantum subsystem.

Result

13 : $[\ \texttt{yes}\ ]_2^0 \to [\ \ ]_2^1\texttt{yes}$,

14 : $[\ \texttt{yes}\ ]_1^0 \to [\ \ ]_1\texttt{yes}$,

15 : $[\ p_i \to p_{i+1}\ ]_1^0,\ 1 \le i \le (n+2)N+4$,

16 : $[\ p_{(n+2)N+5}\ ]_1^0 \to [\ \ ]_1^1\texttt{no},\ 1 \le i \le (n+2)N+4$.

13: If the quantum subsystem detected a match, send this signal out to the skin.

14: Send the final answer `yes` out, also halting the computation.

15: Wait for the possible answer `yes` to appear.

16: If it did not appear in time, send the final answer `no`.

4.2 Quantum Comparison

Quantum comparison is shown in Fig. 4. It uses CNOT, NOT and Toffoli gates. The result is produced on the qubit initialized by $|0\rangle$ (the lowest in the diagram).

4.3 Notes on Complexity

The classical general algorithm solves GI for graphs of n vertices in time $O(c^{\sqrt{n}\log n})$, were c is a constant [3].

Quantum computation has been widely employed at GI solving during last decade. Initially, users of classic algorithms just applied the Grover method for search between relabeled candidates. But for graphs with n nodes a naive application of Grover search means $O(\sqrt{n!})$ queries, so some other quantum methods have been proposed to improve the efficiency. The most popular of these methods seems to be

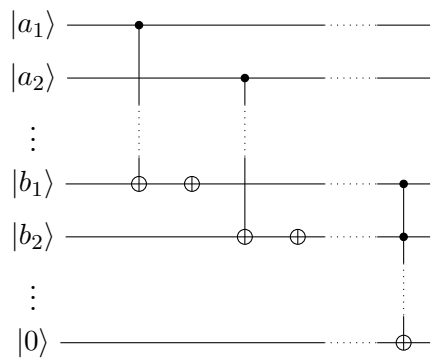

Figure 4. Quantum comparator

the quantum walk [6]. The computation complexity of GI solution applying quantum walk is declared for graph with n vertices as $O(n^7)$ for discrete quantum walk [2] and as $O(n^6)$ for continuous one [5].

In the presented GI solution the P system part of computation takes $2\lceil(\log_2 n)\rceil + 1$ steps. Supposing the pure P system computation the algorithm could execute the comparison of each pairs (candidate/pattern) by 2 steps. Totally the pure P system based comparison would take $2n^2$ steps.

We will count the quantum subsystem comparison as 3 steps. So, the whole work time is $(n+2)\lceil(\log_2 n)\rceil + 4$.

5 Conclusions

This paper concerns the application of membrane-quantum hybrid computational model to speed up the classical brute force algorithm solving the problem of graph isomorphism.

Membrane-quantum hybrid computational model is the P system framework with additional quantum functionalities. With this approach, we obtained computation time advancement against both pure membrane and pure quantum solutions, namely: $O(n \log_2 n)$ (hybrid) against $O(n^2)$ (pure membrane) and $O(n^6)$ (pure quantum).

References

[1] S. Dorn. *Quantum Algorithms for Graph and Algebra Problems.* VDM Verlag (2008).

[2] B.L. Douglas, J.B. Wang. *Classical approach to the graph isomorphism problem using quantum walks.* Journal of Physics A: Mathematical and Theoretical 41(7) (2008).

[3] J. Kobler, U. Schoning, J. Toran. *The graph isomorphism problem: its structural complexity.* Birkhauser Verlag (1994).

[4] Gh. Păun. *Membrane Computing. An Introduction.* Springer (2002).

[5] X. Qiang, X. Yang, J. Wu, X. Zhu. *An enhanced classical approach to graph isomorphism using continuous-time quantum walk.* Journal of Physics: A Mathematical and Theoretical 45(4) (2012).

[6] K. Rudinger, J.K. Gamble, E. Bach, M. Friesen, R. Joynt, S.N. Coppersmith. *Comparing algorithms for graph isomorphism using discrete- and continuous-time quantum random walks.* Journal of Computational and Theoretical Nanoscience 10(7), 1653–1661(9) (2013).

[7] C.P. Williams. *Explorations in Quantum Computing.* Springer (2008).

Artiom Alhazov, Lyudmila Burtseva,
Svetlana Cojocaru, Alexandru Colesnicov,
Ludmila Malahov

Received September 19, 2015

Institute of Mathematics and Computer Science
Str. Academiei 5,
Chişinău, MD-2028,
Moldova
Phone: +373 22 72 59 82
E–mail:
{artiom.alhazov,lyudmila.burtseva,svetlana.cojocaru,kae,mal}@math.md