INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 10(2):263-279, April, 2015.

QEAM: An Approximate Algorithm Using P Systems with Active Membranes

G. Zhang, J. Cheng, M. Gheorghe, F. Ipate, X. Wang

Gexiang Zhang*, Jixiang Cheng
School of Electrical Engineering, Southwest Jiaotong University
Chengdu, 610031, P.R. China
*Corresponding author: zhgxdylan@126.com

Marian Gheorghe
Faculty of Engineering and Informatics, University of Bradford,
Bradford, West Yorkshire BD7 1DP, UK,
m.gheorghe@bradford.ac.uk

Florentin Ipate
Faculty of Mathematics and Computer Science, University of Bucharest
Academiei 14, Bucharest, Romania
florentin.ipate@ifsoft.ro

Xueyuan Wang
School of Information Engineering
Southwest University of Science and Technology
MianYang 621010, P.R.China

Abstract: This paper proposes an approximate optimization approach, called QEAM, which combines a P system with active membranes and a quantum-inspired evolutionary algorithm. QEAM uses the hierarchical arrangement of the compartments and developmental rules of a P system with active membranes, and the objects consisting of quantum-inspired bit individuals, a probabilistic observation and the evolutionary rules designed with quantum-inspired gates to specify the membrane algorithms. A large number of experiments carried out on benchmark instances of satisfiability problem show that QEAM outperforms QEPS (quantum-inspired evolutionary algorithm based on P systems) and its counterpart quantum-inspired evolutionary algorithm.
Keywords: Membrane computing, active membranes, approximate optimization approach, quantum-inspired evolutionary algorithm; satisfiability problem.

1 Introduction

In the last decades, natural computing has been intensively studied and a wide range of applications in computer science and many other areas have been produced. As a well established branch of natural computing, membrane computing, using models called P systems, has made a significant impact on the development of various disciplines [22], such as theoretical computer science, biology, linguistics, etc. The first variants of P systems were proposed in 1998 by G. Păun [19]. They represent a new distributed-parallel framework for designing cell-like or tissue-like computing models, handling multisets of abstract objects in a compartmentalized arrangement of membranes. The membrane structure delimits compartments in a hierarchical or network manner. Objects are arranged as multisets and dispersed across these compartments. Rules are usually associated to the regions enclosed by membranes and control the evolution of objects inside in a maximally parallel way. The main characteristics of P systems are the hierarchical or network architecture of membranes, type of rules (transformation, communication etc.) and intrinsic parallelism, which are all very effective from a computational point of view

and attractive and suitable for modelling various problems. Until now, P systems have been developed principally from a mathematical and computational point of view, building a great variety of computing models and studied for their computational power, complexity aspects and potential solutions to NP-complete problems, and have been utilized for modelling real-world problems in graphics, linguistics, biology. However, the issue of adapting P systems for solving practical problems remains a fundamental aspect of the research in this field, and fortunately a burgeoning interest in this respect for many researchers has been noticeable in the last years [7]. The application of P systems to such problems is still in a developmental phase [22], as compared to evolutionary computation.

Inspired by the evolution in natural selection and molecular genetics, evolutionary algorithms (EAs) have become the most successful metaheuristic search techniques [2]. The great success of EAs in various applications, such as evolutionary optimization and machine learning, can be attributed to two outstanding characteristics: practicability and robustness. EAs are regarded as blind search methodologies without domain specific knowledge [23], suitable for a variety of complex problems in real-world applications. As population-based search tools, EAs usually sample multiple points of the search space in a single step and consequently are quite robust with respect to the objective function landscapes containing many peaks. Developing potential efficient solutions for specific problems is a challenging and attractive topic for researchers from a wide range of areas. Quantum-inspired evolutionary algorithms (QIEAs), one of the three main research areas related to the complex interaction between quantum computing and evolutionary algorithms, are receiving renewed attention [28]. A QIEA is a new evolutionary algorithm for a classical computer rather than for quantum hardware. QIEAs use quantum-inspired bits (Q-bits), quantum-inspired gates (Q-gates) and observation processes to specify their structure and steps. More specifically, Q-bits are applied to represent genotype individuals; Q-gates are employed to operate on Q-bits to generate offspring; and the genotypes and phenotypes are linked by a probabilistic observation process.

Even though P systems and EAs use different rules and computational strategies to handle different objects, both of them are paradigms of natural computing and employed to solve complex problems such as NP-complete problems [27, 34]. P systems represent a suitable formal framework for parallel-distributed computation and EAs are very effective for implementing different algorithms to solve many problems. Thus, the possible interplay between P systems and EAs is very promising for further exploration and represents a fertile research field.

Being the successful instances of this interaction, membrane algorithms can be regarded as a class of hybrid optimization algorithms using the concepts and principles of metaheuristic search methodologies and the hierarchical or network structures of membranes and, to some extent, rules of P systems. When a P system is considered as a parallel-distributed framework for metaheuristic search techniques, it is investigated in terms of optimization results and computation framework, instead of computing power and efficiency. According to the investigations in the literature, there are two main types of membrane algorithms in terms of membrane structures: hierarchical and network. In [30], a tissue membrane system with a network structure was used to appropriately organize five representative variants of differential evolution algorithms. Three principal categories, nested membrane structure (NMS), one-level membrane structure (OLMS) and hybrid membrane structure, were reported with respect to the membrane algorithms with hierarchical membrane structures. In [16], a membrane algorithm with NMS was proposed by using a genetic algorithm and a local search method to solve travelling salesman problems. This kind of membrane algorithms was also applied to solve the min storage problem [13], DNA sequence design problem [24, 25] and the proton exchange membrane fuel cell model parameter estimation problems [26]. In [27], a membrane algorithm integrating OLMS with a QIEA, called QEPS, was proposed to solve knapsack problems and the experiment-based comparisons

between OLMS and NMS were drawn, implying that the choice of the membrane structure is very important for membrane algorithms. This membrane structure was also combined with a QIEA and tabu search [32], differential evolution [3], ant colony optimization [29], particle swarm optimization [33] and multiple QIEA components to solve radar emitter signal time-frequency atom decomposition, numerical optimization problems, travelling salesman problems, broadcasting problems in P systems and image processing, respectively. In [11], a dynamic multi-objective optimization algorithm using a membrane system with a hybrid structure was developed to design a controller for a time-varying unstable plant. The dynamic behavior analysis in [31] indicates that the membrane algorithm, QEPS, has a stronger capability to balance exploration and exploitation than its counterpart approach, QIEA. It is worth pointing out that Păun has made a clear claim that membrane algorithms represent a research directions with a well-defined practical use [22], and therefore further studies are very necessary to prove the use of P systems for solving real-world applications.

P systems with active membranes can produce an exponential growth of membranes and consequently can solve a class of NP-complete problems, such as the satisfiability (SAT) problem [1, 20] and the knapsack problem [18], in a linear or polynomial time. The two types of complete problems were discussed in [1, 18, 20] and in many other places from a mathematical perspective. To the best of our knowledge, no evolutionary algorithm using P system with active membranes has been devised to approximately solve the two aforementioned kinds of problems.

This paper proposes an approximate algorithm combining a P system with active membranes model and a QIEA, called QEAM. This approach is based on the hierarchical arrangement of the compartments and developmental rules (e.g., membrane separation, merging, transformation/communication-like rules) of a P system with active membranes model, and the objects consisting of Q-bit individuals, a probabilistic observation and the evolutionary rules designed with Q-gates to specify the membrane algorithms. In the experiments, the application of QEPS to SAT problems is first discussed, and then QEAM is tested on 65 benchmark SAT problems. Extensive experiments show that QEAM achieves much better results than QEPS and its counterpart QIEA. Also, the parametric and non-parametric tests show significant differences.

2 QEAM

In this section, we start by introducing some concepts related to P systems with active membranes and QIEAs and then describe in detail the proposed QEAM algorithm.

2.1 P systems with active membranes

In this subsection, we give a brief description of P systems with active membranes without polarizations due to [20] and [17], where more details can also be found.

A *membrane structure* is a rooted tree represented by a Venn diagram and is identified by a string of correctly matching parentheses, with a unique external pair of parentheses; this external pair of parentheses corresponds to the external membrane, called *the skin*. A membrane without any another membrane inside (the leaves of the tree) is said to be *elementary*. For example, the structure in Fig. 1 contains 8 membranes; membranes 3, 5, 6 and 8 are elementary. The string of parentheses identifying this structure is $\mu = [_1[_2[_5]_5[_6]_6]_2[_3]_3[_4[_7[_8]_8]_7]_4]_1$.

All membranes are labelled; here we have used the numbers from 1 to 8. We say that the number of membranes is the *degree* of the membrane structure, while the height of the tree associated in the usual way with the structure is its *depth*. In the example above we have a membrane structure of degree 8 and of depth 4.

The membranes delimit *regions*, precisely identified by the membranes (the region of a membrane is delimited by the membrane and all membranes placed immediately inside it, if any such a membrane exists). In these regions we place *objects*, which are represented by symbols of an alphabet. Several copies of the same object can be present in a region, so we work with *multisets* of objects. A multiset over an alphabet V is represented by a string over V, together with all its permutations: the number of occurrences of a symbol $a \in V$ in a string $x \in V^*$ (V^* is the set of all strings over V; the empty string is denoted by λ) is denoted by $|x|_a$ and it represents the multiplicity of the object a in the multiset represented by x.

A *polarizationless P system with active membranes* is a construct

$$\Pi = (V, T, H, \mu, w_1, \ldots, w_m, R),$$

where

1. $m \geq 1$ (the initial *degree* of the system);
2. V is an alphabet (the *working alphabet* of the system);
3. $T \subseteq V$ (the *terminal* alphabet);
4. H is a finite set of *labels* for membranes;
5. μ is a *membrane structure* consisting of m membranes, labelled (not necessarily in a one-to-one manner) with elements of H;
6. $w_1, \ldots, w_m$, are strings over V, describing the *multisets of objects* placed in the m regions of μ;
7. R is a finite set of *developmental rules*, of the following forms:
   (a) $[_h a \to v]_h$, for $h \in H, a \in V, v \in V^*$; (*object evolution rules*, associated with membranes and depending on the label, but not directly involving the membranes, in the sense that the membranes are neither taking part in the application of these rules nor are they modified by them);
   (b) $a[_h]_h \to [_h b]_h$, for $h \in H, a, b \in V$; (*communication rules*; an object is introduced in the membrane, possibly modified during this process);
   (c) $[_h a]_h \to [_h]_h b$, for $h \in H, a, b \in V$; (*communication rules*; an object is sent out of the membrane, possibly modified during this process);
   (d) $[_h]_h[_h]_h \to [_h]_h$, for $h \in H$; (*merging rules for elementary membranes*; in reaction of two membranes, they are merged into a single membrane; the objects of the former membranes are put together in the new membrane);
   (e) $[_h W]_h \to [_h U]_h[_h W - U]_h$, for $h \in H, U \subset W$; (*separation rules for elementary membranes*; the membrane is separated into two membranes with the same labels; the objects from U are placed in the first membrane, those from $W - U$ are placed in the other membrane);

For a detailed description on how to use these rules, refer to [17,20]. It is worth pointing out that these rules are used in the non-deterministic maximally parallel manner, i.e., in any given step, one or more rules of type (a), such that no unallocated object to rules can be allocated to any rule, and/or at most one rule of types (b)-(e) can be applied to each membrane. In this way, we get transition from a configuration of the system to the next configuration. A sequence of transitions forms a computation. A computation is halting if no other rules can be employed in its last configuration.

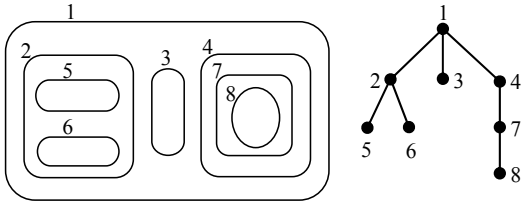

Figure 1: A membrane structure

```
Begin
    t ← 1
(i) Initialize Q(t)
    While (not termination condition) do
(ii)    Make P(t) by observing the states of Q(t)
(iii)   Evaluate P(t)
(iv)    Update Q(t) using Q-gates
(v)     Store the best solutions among P(t)
        t ← t + 1
    End
End
```

Figure 2: Pseudocode algorithm for QIEA [10]

2.2 Quantum-inspired evolutionary algorithms

The interaction of quantum computing and evolutionary algorithms has produced three research avenues: evolutionary-designed quantum algorithms using evolutionary algorithms to design new quantum algorithms, quantum evolutionary algorithms implementing evolutionary algorithms in a quantum computing environment and QIEAs [28]. QIEA is employed to describe the computational methods using concepts and principles of quantum computing for solving various problems in the context of a classical computer [14]. Based on the concepts and principles of quantum computing, such as quantum bit (qubit), quantum gate and superposition, a QIEA is developed as a novel evolutionary algorithm for a classical computer. Narayanan and Moore [15] introduced a preliminary idea of a QIEA and Han and Kim [10] proposed its practical algorithm. A QIEA is characterized by a Q-bit representation, a probabilistic observation and a Q-gate evolutionary rule. In recent years, QIEAs have become a promising and rapidly growing branch of evolutionary computation.

In QIEAs, a Q-bit is defined by a pair of complex numbers $(\alpha,\ \beta)$ as $[\alpha\ \beta]^T$, where $|\alpha|^2$ and $|\beta|^2$ are probabilities that the observation of the Q-bit will render a '0' or '1' state. Normalization requires that $|\alpha|^2 + |\beta|^2 = 1$. Note that QIEAs just need real numbers for amplitudes. Besides '0' and '1' states, a Q-bit can also be in a superposition of the two states. A Q-bit individual is represented as a string of l Q-bits

$$\begin{bmatrix} \alpha_1|\alpha_2|\cdots|\alpha_l \\ \beta_1|\beta_2|\cdots|\beta_l \end{bmatrix}, \tag{1}$$

where $|\alpha_i|^2 + |\beta_i|^2 = 1$ $(i = 1, 2, \cdots, l)$. A Q-gate in a QIEA is defined as a variation operator for updating the Q-bit individuals such as to guarantee that they also satisfy the normalization condition $|\alpha|^2 + |\beta|^2 = 1$ [10].

The basic pseudocode algorithm for a QIEA is shown in Fig. 2 and the description for each step is as follows.

1. In the "initialize $Q(t)$" step, a population $Q(1)$ with n Q-bit individuals is generated, $Q(t)$=$\{\boldsymbol{q}_1^t, \boldsymbol{q}_2^t, \cdots, \boldsymbol{q}_n^t\}$, at generation t, where $\boldsymbol{q}_i^t$ $(i = 1, 2, \cdots, n)$ is an arbitrary individual in $Q(t)$, which is represented as

$$\boldsymbol{q}_i^t = \begin{bmatrix} \alpha_{i1}^t|\alpha_{i2}^t|\cdots|\alpha_{il}^t \\ \beta_{i1}^t|\beta_{i2}^t|\cdots|\beta_{il}^t \end{bmatrix}, \tag{2}$$

where l is the number of Q-bits, i.e., the string length of the Q-bit individual. In the initial population, that is when $t = 1$, we have $\alpha_{ij}^t = \beta_{ij}^t = 1/\sqrt{2}$ for all $i = 1, 2, \cdots, n$

and $j = 1, 2, \cdots, l$. This means that all possible states are superposed with the same probability at the beginning.

2. By observing the states $Q(t)$, binary solutions in $P(t)$, where $P(t)$=$\{\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \cdots, \boldsymbol{x}_n^t\}$, are produced at step t. According to the current probability, either $|\alpha_{ij}^t|^2$ or $|\beta_{ij}^t|^2$ of $\boldsymbol{q}_i^t$, $i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, l$, a classical bit 0 or 1 is generated. Thus, l classical bits can construct a binary solution $\boldsymbol{x}_i^t$ $(i = 1, 2, \cdots, n)$.

3. The fitness value for each binary solution $\boldsymbol{x}_i^t$ $(i = 1, 2, \cdots, n)$ is calculated by using an evaluation function.

4. In this step, all the Q-bit individuals in $Q(t)$ are updated by applying Q-gates. To be specific, the jth Q-bit in the ith Q-bit individual $\boldsymbol{q}_i^t$, $j = 1, 2, \cdots, l$, $i = 1, 2, \cdots, n$, is updated by applying the current Q-gate $\boldsymbol{G}_{ij}^t(\theta)$. As usual, QIEAs use a quantum rotation gate as a Q-gate; this is given by

$$\boldsymbol{G}_{ij}^t(\theta) = \begin{bmatrix} \cos\theta_{ij}^t & -\sin\theta_{ij}^t \\ \sin\theta_{ij}^t & \cos\theta_{ij}^t \end{bmatrix}, \tag{3}$$

   where θ_{ij}^t is an adjustable Q-gate rotation angle.

5. The best solutions among $P(t)$ are selected and stored into $b(t)$.

In QIEAs, the Q-bit representation, which can describe simultaneously multiple genotype states using a linear superposition of states in a probabilistic way, makes the algorithm rather good with respect to population diversity. Q-gate evolutionary rules are executed in the Q-bit probability space to avoid the selection pressure problem of conventional genetic algorithms with selection, crossover and mutation operators. As compared with local search methods and conventional genetic algorithms, a QIEA has good balance between exploration and exploitation so as to obtain stronger global search capability and better convergence. Furthermore, a QIEA is able to exploit the search space for a global solution with a small number of individuals, even with one individual; Q-gate evolutionary rules, which are only related to searching the best solution, are easy to implement in a parallel distributed structure because little information needs to be transmitted and exchanged.

2.3 QEAM

This section will introduce the membrane algorithm, QEAM, combining P systems with active membranes and QIEAs. QEAM uses a dynamic P systems-like framework, which is initially randomly produced and then may be changed in the process of evolution. This framework directly uses some of the elements of a P system with active membranes, whereas others are slightly adapted for this evolutionary algorithm. The objects employed will be organized in multisets of special strings built either over the set of Q-bits or $\{0, 1\}$. The rules will be responsible to make the system evolve and select the best fit Q-bit individuals.

More precisely, the dynamic P system-like framework will consist of:

1. a dynamic structure $[_0[_1]_1, [_2]_2, \cdots, [_m]_m]_0$ with m regions contained in the skin membrane, denoted by 0, where m is a number varied during the evolution process;

2. an alphabet that consists of all possible Q-bits and the set $\{0, 1\}$;

3. a set of terminal symbols, $T = \{0, 1\}$;

4. initial multisets
   $w_0 = \lambda$, $w_1 = q_1 q_2 \cdots q_{n_1}$, $w_2 = q_{n_1+1} q_{n_1+2} \cdots q_{n_2}$, $\cdots$, $w_m = q_{n_{(m-1)}+1} q_{n_{(m-1)}+2} \cdots q_{n_m}$, where q_i, $1 \leq i \leq n$, is a Q-bit individual; n_j, $1 \leq j \leq m$, is the number of individuals in w_j; $\sum_{j=1}^{m} n_j = n$, where n is the total number of individuals in this computation;

5. rules include the types of (a)-(e) in P systems with active membranes and their use will be given in the following description.

In what follows, we summarize the steps of QEAM by using a pseudocode notation, shown in Fig. 3, to help presenting the membrane algorithm.

```
    Begin
      Input n, g_max, t_max
      t ← 1
(i)   Initialize the membrane structure (m_t elementary membranes) and objects;
(ii)  While (not termination condition) do
(iii)     Produce g(t);
(iv)      Perform object evolution rule (a) in elementary membranes;
(v)       Perform communication rule (c);
(vi)      Perform object evolution rule (a) in the skin membranes;
          t ← t + 1
(vii)     Determine the number m_{t+1} of elementary membranes;
          If (m_{t+1} < m_t)
(viii)      Perform membrane merging rule (d);
          Else if (m_{t+1} > m_t)
(ix)        Perform membrane separation rule (e);
          End
(x)       Perform communication rule (b);
      End
      Output: the best individual
    End
```

Figure 3: Pseudocode algorithm for QEAM.

1. In the initialization of QEAM, a one level membrane structure $[_0[_1]_1, [_2]_2, \cdots, [_m]_m]_0$ consisting of a skin membrane denoted by 0 and m elementary membranes delimiting m regions inside the skin membrane is constructed as the membrane structure at iteration $t = 1$, where m is a random number ranged from 1 to n, where n is the number of Q-bit individuals. Each Q-bit individual forms an object. Thus, n objects are randomly scattered across the m elementary membranes in a non-deterministic way to make sure that each elementary membrane contains at least one object. So the number of objects in each elementary membrane varies from 1 to $n - m + 1$.

2. The termination condition for QEAM could be a prescribed number of maximal iterations or the algorithm searches the optimal or close-to-optimal solution.

3. This step determines the numbers $\boldsymbol{g}(t) = (g_1, g_2 \ldots, g_m)$, of evolutionary generations for independently performing object evolution rule (a) in the m elementary membranes, where g_i $(i = 1, 2, \ldots, m)$ for the ith elementary membrane is generated randomly between 1 and a certain integer number g_{max}.

4. The steps (ii) to (v) of the QIEA shown in Fig. 2 are performed independently in each elementary membrane to evolve the objects inside. The termination condition for the ith

```
Begin
  If random[0,1) < |α|^2
  Then x ← 0
  Else x ← 1
End
```

Figure 4: Observation process in QIEA

```
Begin
  t ← 0;
  Initialize tabu search;
  While (t < t_max) do
    t ← t + 1;
    Search the neighborhood;
    Evaluate candidate solutions;
    Update tabu list.
  End
End
```

Figure 5: Pseudocode algorithm for tabu search

elementary membrane is the maximal number $g_i (i = 1, 2, \cdots, m)$ of evolutionary generations. It is worth pointing out that the observation process in QIEAs, illustrated in Fig. 4, is applied to build a connection between a Q-bit $[\alpha\ \beta]^T$ and a classical bit and hence to build a link between Q-bit individuals and binary solutions. The Q-gate update procedure

$$\begin{bmatrix} \alpha\prime \\ \beta\prime \end{bmatrix} = G(\theta) \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{4}$$

is used to transform a current Q-bit $[\alpha\ \beta]^T$ into the corresponding Q-bit $[\alpha\prime\ \beta\prime]^T$ at the next generation. The rotation angle θ in the Q-gate $G(\theta)$ in (4) is defined as $\theta = s(\alpha, \beta) \cdot \Delta\theta$, where $s(\alpha, \beta)$ and $\Delta\theta$ can be obtained from the lookup table in [10].

5. The communication rule is employed to send the best binary solution in each elementary membrane out to the skin membrane. This step is helpful to exchange information among the objects in the elementary membranes and the skin membrane because the QIEA employs Q-gates, which are related to only the best individual searched, to generate the offspring. After this step there are m binary solutions in total in the skin membrane.

6. In the skin membrane, a local search, tabu search [8, 12], is performed on the best binary solution selected from the m binary solutions, which are sent from the m elementary membranes (see Step (v)). The pseudocode algorithm for tabu search is shown in Fig. 5. In the "Initialize tabu search" step, an empty tabu list is constructed and tabu length is set to a value. At each iteration, the neighborhood of the best binary solution in the skin membrane is explored to obtain candidate solutions. Next, the candidate solutions are evaluated by using the fitness function and the best of them is selected to update the tabu list.

7. The number m_{t+1} of elementary membranes at iteration $t+1$ is produced randomly between 1 and n, which will directly determine the membrane structure at the next iteration.

8. If $m_{t+1} < m_t$, the $(m_t - m_{t+1})$ elementary membranes will be merged into the m_{t+1} elementary membranes. The merging process is shown in Fig. 6, where EM represents elementary membranes. In each merging operation, we first choose any two arbitary elementary membranes i and j from M elementary membranes, i.e., $1 \leq i, j \leq M$ and $i \neq j$; and then we merge the elementary membranes i and j into a single membrane and put the objects in the elementary membranes i and j into the merged membrane. The initial value of M is m_t. Thus, multiple membranes may be merged into a single membrane. So this rule is a multi-merging one.

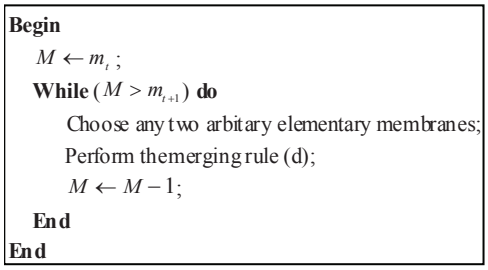

Figure 6: Merging process of EMs

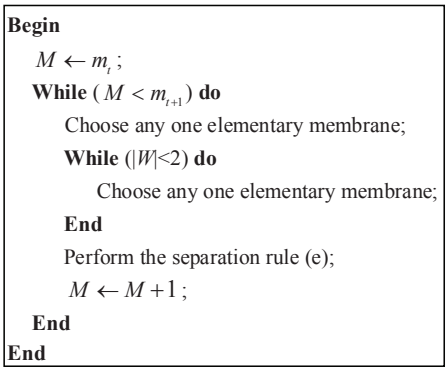

Figure 7: Separation process of EMs

9. If $m_{t+1} > m_t$, the $(m_{t+1} - m_t)$ elementary membranes will be separated into two membranes. The separation process is illustrated in Fig. 7, in which $|W|$ is the number of objects in the pre-separation membrane. We choose any one elementary membrane i which has at least two objects from M elementary membranes, i.e., $1 \leq i \leq M$. The initial value of M is m_t. When the separation rule is performed, $|U|$ ($|U| < |W|$) objects are placed in the first membrane and the $|W| - |U|$ objects are placed in the other membrane. Thus, a single membrane may be divided into several membranes. So, this rule is a multi-separation one.

10. By performing the communication rule (b) in the skin membrane, this step sends the fittest binary solution to each elementary membrane for the further evolution steps.

3 Experimental results

To test the performances of the presented algorithm, QEAM, we will use the satisfiability problem, which is a well-known NP-complete problem, to conduct the experiments. We start from the description of the satisfiability problem, and then turn to use QIEAs and QEPS as benchmark algorithms to solve 65 representative instances of the satisfiability problem. Finally, QEAM is tested on the same instance of the satisfiability problem to draw conclusions.

3.1 Satisfiability problem

The satisfiability problem (SAT) is a fundamentally paradigmatic problem in artificial intelligence applications, automated reasoning, mathematical logic, and related research areas [5]. SAT can be described as follows: given a Boolean formula in conjunctive normal form (CNF), determine whether or not it is satisfiable, that is, whether there exists an assignment to its variables on which it evaluates to true, i.e., a SAT instance is to search a variable assignment $\boldsymbol{x}$ so that a Boolean formula $f(\boldsymbol{x})$ becomes true, where $\boldsymbol{x}$ is a set of Boolean variables $x_1, x_2, \cdots, x_n$, i.e., $x_i \in \{0,1\}$, $i = 1, 2, \cdots, n$ and the propositional formula $f(\boldsymbol{x})$ is in a conjunctive normal form, i.e.,

$$f(\boldsymbol{x}) = c_1(\boldsymbol{x}) \wedge c_2(\boldsymbol{x}) \wedge \cdots \wedge c_m(\boldsymbol{x}), \tag{5}$$

where each clause $c_j(\boldsymbol{x})$, $j = 1, 2, \cdots, m$, is a disjunction of literals, and a literal is a variable or its negation [9]. A SAT instance is called *satisfiable* if such $\boldsymbol{x}$ exists, and *unsatisfiable* otherwise. In this paper only 3-SAT problems, in which each clause has exactly three literals, will be considered

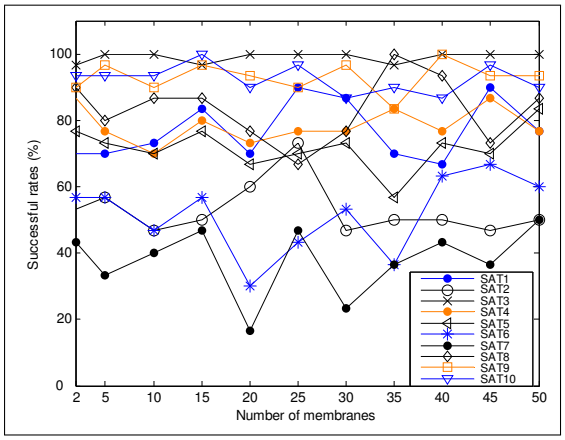

Figure 8: Successful rates

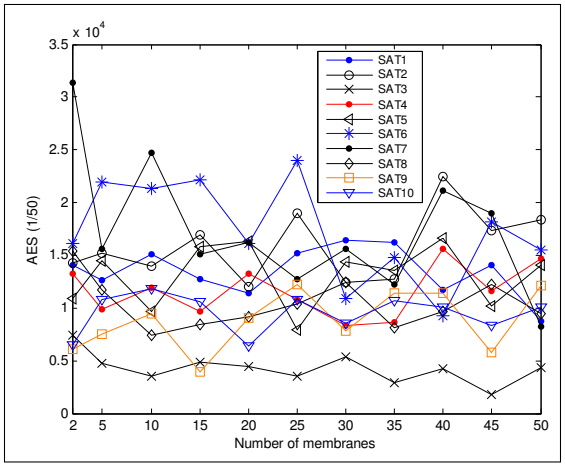

Figure 9: AES

because a number of other problems, such as the travelling salesman problem and the n-queens problem, can be reformulated with respect to 3-SAT problems. In [4] it is shown that 3-SAT problem is NP-complete.

In membrane computing, various types of P systems with membrane division are frequently investigated from a mathematical point of view to obtain an exponential working space in a linear time to solve the SAT problem [1, 20]. This paper will use an approximate algorithm to solve the SAT problem, in which the number of clauses that are not satisfied by the variable assignment $\boldsymbol{x}$ is considered as the evaluation function.

3.2 Results of QEPS and QIEA

According to the study in [27], the number of elementary membranes has a significant impact on the QEPS performances. So we first focus on how to set the number of elementary membranes in an empirical way. Ten benchmark 3-SAT problems[1], each of which has 20 Boolean variables and 91 clauses, are applied to conduct the experiments. The fitness function is the number of clauses that are not satisfied by the variable assignment. The population size n is set to 50. The values of 2, 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50, for the number m of elementary membranes, are used in the experiments. According to previous investigations regarding the effect of the number $g_i (i = 1, 2, \ldots, m)$ of iterations on the QEPS performances [27], the parameter $g_i (i = 1, 2, \ldots, m)$ is set to a uniformly random integer ranged from 1 to 10. The algorithm stops when either 2.75×10^6 evaluation steps are made or the SAT problem solution is found, i.e., the minimal fitness value 0 is attained. The performances of the above 11 cases are evaluated by using the successful rate of 30 independent runs (the percentage of the runs making the SAT problem satisfiable) and the average number of evaluations to solutions (AES) over the successful runs. The experimental results are listed in Fig. 8 and Fig. 9, which illustrate that the successful rates and the AES vary with the number of elementary membranes.

As shown in Fig. 8 and Fig. 9, the successful rates and the AES show a broad range of variability with respect to the number of different elementary membranes; this indicates that the number of elementary membranes has a significant impact on the QEPS performances. In order to obtain a balance between the successful rates and the AES, the number of elementary membranes could be fixed at 15.

[1]SATLIB - The Satisfiability Library, http://www.satlib.org/

QIEA is also applied to conduct the experiments on the 10 SAT problems. In these experiments, QIEA employs the same population size and stopping criteria as the QEPS. The statistical results of 30 independent runs for each problem are listed in Table 1. The best experimental results of QEPS are also shown in Table 1, where each of which has 20 Boolean variables and 91 clauses; SR and AES represent successful rates and average number of evaluations to solutions, respectively.

To perform convincing comparisons between QIEA, QEPS and QEAM, additional fifty-five 3-SAT benchmark problems are employed to carry out experiments. Both QEPS and QIEA use 50 individuals as a population, the prescribed number of 2.75×10^6 evaluations to solutions as the stopping criterion and the number of clauses that are not satisfied by the variable assignment as the fitness function. In QEPS, the parameter $g_i (i = 1, 2, \ldots, m)$ is set to a uniformly random integer ranged from 1 to 10, and the number of elementary membranes is assigned to 15. The performances of the two algorithms are evaluated by using the following criteria: the mean of the solutions over 15 runs and their standard deviations. It is worth pointing out that the experiments are very time-consuming and therefore only 15 independent runs are performed for each SAT problem. The number of Boolean variables, the number of clauses in each Boolean formula and the experimental results are provided in Table 2.

Table 1: Comparisons of QIEA, QEPS and QEAM on 10 SAT problems

	QIEA		QEPS		QEAM	
Problems	SR(%)	AES	SR(%)	AES	SR(%)	AES
SAT1	77	572750	100	528850	100	220804
SAT2	70	496700	90	701200	100	300468
SAT3	53	638250	73	949400	97	279174
SAT4	100	218250	100	90300	100	59978
SAT5	87	575900	87	582300	100	179445
SAT6	57	469650	83	701700	100	336728
SAT7	53	1137750	67	907300	93	527795
SAT8	27	1120300	50	410300	100	354903
SAT9	87	684800	100	405450	100	128633
SAT10	93	281050	100	572750	100	115876

3.3 Results of QEAM

In the experiments for testing QEAM performance, the population size and the prescribed number of evaluations of solutions as the stopping criterion are set to 50 and 2.75×10^6, respectively, which are the same as those in QIEA and QEPS. QEAM applies the same g_{max} as QEPS. Additionally, the tabu length and t_{max} in QEAM are 5 and 100, respectively. For each of the first 10 benchmark 3-SAT problems shown in Table 2, we performed 30 independent runs and recorded the successful rate and the AES over successful runs. The experimental results are provided in Table 1. The QEAM performance is further investigated by using the remaining 55 3-SAT benchmark problems. We record the average solution and the standard deviations over 15 runs for each of them. The experimental results are listed in Table 2, where each of the first ten, the second ten, the third ten, the fourth ten, the fifth ten and the last five problems has 50, 75, 100, 125, 150 and 250 Boolean variables and 218, 325, 430, 538, 645 and 1065 clauses, respectively; Mean and Std represent the mean of the best solutions and the standard deviation of the best solutions, respectively; (+) represents significant difference.

Table 2: Comparisons of QIEA, QEPS and QEAM using 55 instances of the SAT problem (to be continued).

	QIEA		QEPS		QEAM		QEAMvs.QIEA		QEAMvs.QEPS	
SAT	Mean	Std	Mean	Std	Mean	Std	t-test	Imp.(%)	t-test	Imp.(%)
1	7.67	0.82	6.60	1.18	0.93	0.26	5.23e-23(+)	+87.87	5.30e-17(+)	+85.91
2	8.27	2.12	7.40	1.12	1.53	0.64	2.32e-12(+)	+81.50	1.13e-16(+)	+79.32
3	7.40	1.40	6.27	0.88	1.00	0.53	5.90e-16(+)	+86.49	5.64e-18(+)	+84.05
4	7.87	1.19	6.33	0.98	1.00	0.38	7.31e-19(+)	+87.29	5.74e-18(+)	+84.20
5	7.13	1.41	6.20	0.86	0.07	0.26	1.30e-17(+)	+99.02	2.50e-21(+)	+98.87
6	7.27	0.80	5.93	0.80	0.20	0.41	5.39e-23(+)	+97.25	1.53e-20(+)	+96.63
7	7.73	1.16	6.40	1.18	1.07	0.46	1.73e-18(+)	+86.16	8.26e-16(+)	+83.28
8	8.73	0.70	7.67	1.18	1.53	0.83	5.99e-21(+)	+82.47	6.01e-16(+)	+80.05
9	7.87	1.13	6.87	1.19	1.47	0.64	1.27e-17(+)	+81.32	2.84e-15(+)	+78.60
10	8.33	0.82	6.93	0.88	1.07	0.59	5.74e-22(+)	+87.15	7.33e-19(+)	+84.56
11	16.67	1.11	15.40	0.99	2.00	0.93	5.05e-26(+)	+88.00	9.32e-26(+)	+87.01
12	15.07	1.49	14.60	0.74	1.47	0.74	1.76e-23(+)	+90.25	1.38e-28(+)	+89.93
13	16.33	0.82	14.80	2.01	2.07	0.88	6.61e-28(+)	+87.32	1.84e-19(+)	+86.01
14	14.00	1.20	12.87	1.60	1.87	0.64	1.53e-24(+)	+86.64	1.41e-20(+)	+85.47
15	15.07	1.03	14.87	0.92	1.60	1.06	9.13e-25(+)	+89.38	3.01e-25(+)	+89.24
16	16.13	1.06	14.87	1.19	2.20	0.68	4.29e-27(+)	+86.36	5.80e-25(+)	+85.21
17	15.33	1.40	14.53	1.64	1.67	0.90	1.54e-23(+)	+89.11	2.00e-21(+)	+88.51
18	15.73	1.44	14.53	1.51	2.20	0.86	2.54e-23(+)	+86.01	8.03e-22(+)	+84.86
19	14.93	1.49	13.80	1.97	1.80	0.41	6.02e-24(+)	+87.94	9.25e-20(+)	+86.96
20	14.40	1.45	13.93	1.22	2.00	0.76	1.48e-22(+)	+86.11	1.19e-23(+)	+85.64
21	24.53	1.81	22.93	1.39	3.0	0.93	1.44e-26(+)	+87.77	5.29e-28(+)	+86.92
22	24.20	1.21	22.80	2.14	3.33	0.62	4.78e-31(+)	+86.24	3.08e-24(+)	+85.39
23	24.27	1.28	22.93	1.79	4.20	0.94	1.15e-28(+)	+82.69	6.05e-25(+)	+81.68
24	23.40	1.68	22.80	1.08	3.53	0.92	2.63e-26(+)	+84.91	1.51e-29(+)	+84.52

Table 2 Comparisons of QIEA, QEPS and QEAM (continued)

	QIEA		QEPS		QEAM		QEAM vs. QIEA		QEAM vs. QEPS	
SAT	Mean	Std	Mean	Std	Mean	Std	t-test	Imp.(%)	t-test	Imp.(%)
25	24.27	1.22	22.80	1.47	3.73	0.88	1.45e-29(+)	+84.63	4.13e-27(+)	+83.64
26	24.00	1.51	22.47	1.60	3.60	0.74	3.53e-28(+)	+85.00	1.06e-26(+)	+83.98
27	24.13	1.06	23.40	1.35	3.87	0.74	2.99e-31(+)	+83.96	1.08e-28(+)	+83.46
28	24.00	1.85	23.40	1.30	3.73	0.80	6.33e-26(+)	+84.46	6.40e-29(+)	+84.06
29	25.13	1.68	24.13	2.00	4.27	1.28	1.06e-25(+)	+83.01	9.18e-24(+)	+82.30
30	22.93	2.15	22.27	1.98	3.40	0.74	4.81e-24(+)	+85.17	1.64e-24(+)	+84.73
31	33.53	1.96	32.87	1.51	5.67	0.90	6.07e-29(+)	+83.09	3.87e-31(+)	+82.75
32	33.80	1.90	32.07	2.12	5.40	1.06	6.07e-29(+)	+84.02	2.76e-27(+)	+83.16
33	34.47	1.81	33.73	1.33	6.07	1.33	4.37e-28(+)	+82.39	1.85e-30(+)	+82.00
34	34.06	1.84	33.27	2.34	5.13	0.74	1.13e-30(+)	+84.94	1.78e-27(+)	+84.58
35	34.67	1.76	33.60	2.20	5.20	1.32	2.25e-29(+)	+85.00	4.32e-27(+)	+84.52
36	34.80	2.21	33.93	1.44	6.20	1.47	9.51e-27(+)	+82.18	1.93e-29(+)	+81.73
37	32.80	2.86	32.93	1.33	5.27	1.28	2.49e-24(+)	+83.93	1.05e-30(+)	+84.00
38	32.80	1.97	32.47	1.19	5.27	1.10	3.02e-28(+)	+83.93	4.14e-32(+)	+83.77
39	34.20	2.08	33.40	1.76	5.67	0.98	1.78e-28(+)	+83.42	1.09e-29(+)	+83.02
40	33.93	1.67	33.20	2.34	5.67	0.98	1.96e-30(+)	+83.29	7.19e-27(+)	+82.92
41	42.60	1.50	41.20	2.01	6.87	1.13	1.29e-33(+)	+83.87	1.13e-30(+)	+83.33
42	43.07	1.67	40.00	2.67	5.40	0.91	4.16e-34(+)	+87.46	2.66e-28(+)	+86.50
43	41.73	1.71	41.53	1.06	6.27	1.75	2.55e-30(+)	+84.97	2.08e-32(+)	+84.90
44	42.80	3.28	41.07	1.94	6.47	1.25	2.73e-26(+)	+84.88	1.01e-30(+)	+84.25
45	43.93	1.67	42.60	2.64	7.13	1.06	2.38e-33(+)	+83.77	1.66e-28(+)	+83.26
46	43.00	3.23	42.07	1.62	7.60	1.30	4.55e-26(+)	+82.33	6.09e-32(+)	+81.93
47	43.20	2.04	42.73	1.87	7.87	1.30	2.12e-30(+)	+81.78	5.60e-31(+)	+81.58
48	44.27	2.19	43.60	2.32	7.47	1.06	7.50e-31(+)	+83.13	4.97e-30(+)	+82.87
49	44.67	2.26	43.53	2.10	8.93	1.67	9.21e-29(+)	+80.01	6.37e-29(+)	+79.49

Table 2 Comparisons of QIEA, QEPS and QEAM (continued)

	QIEA		QEPS		QEAM		QEAM vs. QIEA		QEAM vs. QEPS	
SAT	Mean	Std	Mean	Std	Mean	Std	t-test	Imp.(%)	t-test	Imp.(%)
50	43.13	1.55	41.47	2.50	6.53	0.99	3.87e-34(+)	+84.86	5.45e-29(+)	+84.25
51	83.07	3.45	81.27	2.91	12.93	1.33	1.48e-33(+)	+84.43	5.51e-35(+)	+84.09
52	83.40	3.44	82.73	2.96	15.27	1.67	8.05e-33(+)	+81.69	4.07e-34(+)	+81.54
53	83.00	2.73	81.60	2.50	12.67	1.50	1.05e-35(+)	+84.73	3.04e-36(+)	+84.47
54	85.13	3.23	83.60	3.14	15.80	1.66	1.15e-33(+)	+81.44	1.14e-33(+)	+81.10
55	84.87	2.83	80.80	2.31	14.20	1.52	2.22e-35(+)	+83.27	1.76e-36(+)	+82.43

According to these experimental results, we employ statistical techniques to analyze the behaviour of the three algorithms over the 55 instances of the SAT problem. There are two statistical methods: parametric and non-parametric [6]. The former, also called single-problem analysis, uses a parametric statistical analysis t-test to analyse whether there is a significant difference between the two algorithms solving the optimization problem. The latter, also called multiple-problem analysis, applies non-parametric statistical tests such as Wilcoxon's and Friedman's tests, to compare different algorithms whose results represent average values for each problem, regardless of the inexistence of relationships among them. Therefore, a 95% confidence Student t-test is first applied to check whether the number of false clauses of the two pairs of algorithms, QEAM vs. QIEA and QEAM vs. QEPS, are significantly different or not. Furthermore, the percentage of improvement (%) in the average number of false clauses due to the QEPS algorithm over QIEA and QEPS is also listed in Table 2. Then two non-parametric tests, Wilcoxon's and Friedman's tests, are employed to check whether there are significant differences between the two pairs of algorithms, QEAM vs. QIEA and QEAM vs. QEPS. The level of significance considered is 0.05. The results of Wilcoxon's and Friedman's tests are shown in Table 3. The symbols + and - in Tables 2–3 represent significant difference and no significant difference, respectively.

In the experiments carried out on the QEAM, the QEPS and the QIEA, we also record the average elapsed time for the first ten SAT problems over 30 independent runs and for the remaining 55 instances of the SAT problem over 15 independent runs. The comparisons of the three algorithms are illustrated in Fig. 10. The x-axis and y-axis represent the number of SAT problems and the elapsed time, respectively.

As shown in Table 1, the QEAM greatly outperforms the QIEA and the QEPS in terms of the successful rates and the average number of evaluations. Also, Table 1 shows that QEPS obtains higher successful rates and smaller average number of evaluations than QIEA. It can be seen from the experimental results of 55 SAT bench problems in Table 2 that the QEAM achieves much better results than the QIEA and the QEPS. The QEPS obtains better results than the QIEA in 54 out of 55 cases. The t-test results demonstrate that there are 55 significant differences between the two pairs of algorithms, QEAM vs. QIEA and QEAM vs. QEPS. The p-values of the two non-parametric tests in Table 3 are far smaller than the level of significance 0.05, which indicates that the QEAM really outperforms the QIEA and the QEPS by introducing the framework and some rules of P systems with active membranes. It is worth noting that the study in [6] shows that the non-parametric statistical tests are more appropriate than parametric statistical tests in the analysis of the behaviour of the evolutionary algorithms over multiple optimization problems.

The QEPS uses the framework and some evolution rules of P systems. Each elementary membrane evolves for a certain number of generations in a non-deterministic way, and then all elementary membranes communicate in the skin membrane. Thus, the QEPS has better population diversity and the capability to balance exploration and exploitation. Consequently the QEPS obtains better results and smaller elapsed time, shown in Fig. 10, than the QIEA. The QEAM goes further and applies the framework and some evolution rules of P systems with active membranes. The good performance of the QEAM is

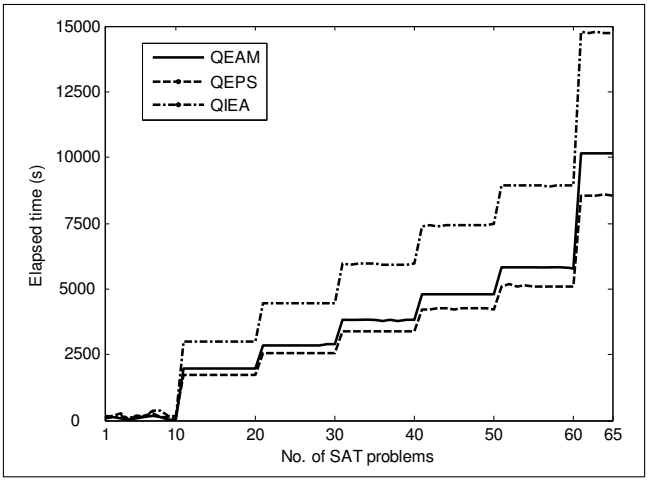

Figure 10: Comparisons of elapsed time.

due to the combination of independent evolution of each elementary membrane in a non-deterministic way, communication in the skin membrane, membrane separation and merging, and a local search in the skin membrane. Figure 10 shows that the QEAM and QEPS consumes less time than QIEA, which indicates that the use of evolution rules of P systems in the QEAM and QEPS has little effect on the overall computational load. Furthermore, the QEAM and QEPS may use a slightly smaller number of evaluations of the solutions than the QIEA because of the randomness of evolutionary generations for each elementary membrane. Additionally, as a result of the use of membrane separation and division, the QEAM consumes slightly more time than the QEPS, which is shown in Fig. 10.

Table 3: Results of non-parametric statistical tests for the two pairs of algorithms, QEAM vs. QEPS and QEAM vs. QIEA, in Table 2. The symbol + represents significant difference.

Tests	QEAM vs. QIEA	QEAM vs. QEPS
Wilcoxon test (p-value)	1.21e-13 (+)	1.21e-13 (+)
Friedman test (p-value)	1.11e-10 (+)	1.11e-10 (+)

4 Conclusions

Membrane algorithms, defined by carefully mixing selected ingredients of P systems and meta-heuristic search methodologies, and the interaction between P systems and quantum computing, are highly promising and give rise to challenging research issues, which are mentioned as open problems and research topics in [7, 21]. Benefiting from the cross-fertilization of ideas from P systems, evolutionary computation and quantum computing areas, this paper discussed a novel membrane algorithm combing P systems with active membranes and QIEA to solve satisfiability problem. A large number of experiments show that QEAM performs better than QEPS and QIEA. As further work, we aim to investigate other interactions between the three disciplines and their applications to specific problems.

Acknowledgments

The work of GZ is supported by the National Natural Science Foundation of China (61170016, 61373047), the Program for New Century Excellent Talents in University (NCET-11-0715) and SWJTU supported project (SWJTU12CX008). The work of MG and FI was partially supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-II-ID-PCE-2011-3-0688.

Bibliography

[1] Alhazov, A.; Martin-Vide, C.; Pan, L.Q. (2003); Solving a PSPACE- complete problem by recognizing P systems with restricted active membranes, *Fund Inform*, ISSN 0169-2968, (2): 67-77.

[2] Bonissone, P.P.; Subbu, R.; Eklund, N.; Kiehl, T.R. (2006); Evolutionary algorithms + domain knowledge = real-world evolutionary computation, *IEEE T Evolut Comput*, ISSN 1089-778X, 10(3):256-280.

[3] Cheng, J.; Zhang, G.; Zeng, X.(2011); A novel membrane algorithm based on differential evolution for numerical optimization, *Int J Unconv Comput*, ISSN 1548-7199, 7(3):159-183.

[4] Cook, S. (1971); The complexity of theorem-proving procedures, *Proc. of STOC*, 151-158.

[5] Folino, G.; Pizzuti, C.; Spezzano, G. (2001); Parallel hybrid method for SAT that couples genetic algorithms and local search, *IEEE T Evolut Comput*, ISSN 1089-778X, 5(4):323-334.

[6] Garcia, S.; Molina, D.; Lozano, M.; Herrera, F.(2009); A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC 2005 special session on real parameter optimization, *J Heuristics*, ISSN 1381-1231, 15(6):617-644.

[7] Gheorghe, M.; Păun, Gh.; Prez-Jimenez, M.J.; Rozenberg, G. (2013); Frontiers of membrane computing: Open problems and research topics, *Int J Found Comput Sci*, ISSN 129-0541, 24(5):547-623.

[8] Glover, F.; Taillard, E.; Werra de, D. (1993); A users guide to tabu search, *Ann Oper Res*, ISSN 0254-5330, 41(1):3-28.

[9] Gottlieb, J.; Marchiori, E.; Rossi, C. (2002); Evolutionary algorithms for the satisfiability problem, *Evolut Comput*, ISSN 1063-6560, 10(1):35-50.

[10] Han, K.H.; Kim, J.H.(2002); Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, *IEEE T Evolut Comput*, ISSN 1089-778X, 6(6):580-593.

[11] Huang, L.; Suh, I.H.; Abraham, A.(2011), Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants, *Inform Sciences*, ISSN 0020-0255, 181(11): 2370-2391.

[12] Hwang, G.J.; Yin, P.Y.; Yeh, S.H.(2006); A tabu search approach to generating test sheets for multiple assessment criteria, *IEEE T Educ*, ISSN 0018-9359, 49(1):88-97.

[13] Leporati, A.; Pagani, D. (2006); A membrane algorithm for the min storage problem, *Lect Notes Comput Sci*, ISSN 0302-9743, 4361:443-462.

[14] Moore, M.; Narayanan, A. (1995); *Quantum-inspired computing. Tech. rep.*, Department of Computer Science, University Exeter, Exeter, U.K.

[15] Narayanan, A.; Moore, M. (1996); Quantum-inspired genetic algorithms, *Proc of IEEE CEC*, 61-66.

[16] Nishida, T.Y.(1996); Membrane algorithm with brownian subalgorithm and genetic subalgorithm. *Int J Found Comput Sci*, ISSN 129-0541, 18(6):1353-1360.

[17] Pan, L.Q., Alhazov, A., Ishdorj, T.O. (2005); Further remarks on P systems with active membranes, separation, merging, and release rules. *Soft Comput*, ISSN 1432-7643, 9(9):686-690.

[18] Pan, L.Q.; Martín-Vide, C. (2005); Solving multidimensional 0-1 knapsack problem by P systems with input and active membranes, *J Parallel Distr Comput*, ISSN 0743-7315, 65(12):1578-1584.

[19] Păun, Gh. (2000); Computing with membranes, *J Comput Syst Sci*, ISSN 0022-0000, 61(1):108-143.

[20] Păun, Gh. (2001); P systems with active membranes: attacking NP-complete problems. *J Automata Lang Comb*, ISSN 1430-189X, 6(1):75-90.

[21] Păun, Gh. (2007); Tracing some open problems in membrane computing. *Rom J Inf Sci Tech*, ISSN 1453-8245, 10(4):303-314.

[22] Păun, Gh.; Rozenberg, G.; Salomaa, A., eds. (2010); *The Oxford Handbook of Membrane Computing*. Oxford University Press.

[23] Whitley, D. (2001); An overview of evolutionary algorithms: practical issues and common pitfalls. *Inform Software Tech*, ISSN 0950-5849, 43(14):817-831.

[24] Xiao, J.H.; Zhang, X.Y.; Xu, J.(2012); A membrane evolutionary algorithm for DNA sequence design in DNA computing. *Chinese Sci Bull*, ISSN 1001-6538, 57(6):698-706.

[25] Xiao, J.H.; Jiang, Y.; He, J.J.; Cheng, Z.(2013); A dynamic membrane evolutionary algorithm for solving DNA sequences design with minimum free energy. *MATCH-Commun Math Ch*, ISSN 0340-6253, 70(3):971-986.

[26] Yang, S.; Wang, N. (2012): A novel P systems based optimization algorithm for parameter estimation of proton exchange membrane fuel cell model. *Int J Hydrogen Energ*, ISSN 0360-3199, 37(10): 8465-8476.

[27] Zhang, G.; Gheorghe, M.; Wu, C. (2008); A quantum-inspired evolutionary algorithm based on P systems for knapsack problem, *Fund Inform*, ISSN 0169-2968, 87(1):93-116.

[28] Zhang, G. (2011); Quantum-inspired evolutionary algorithms: a survey and empirical study. *J Heuristics*, ISSN 1381-1231, 17(3): 303-351.

[29] Zhang, G.; Cheng, J.; Gheorghe, M. (2011); A membrane-inspired approximate algorithm for traveling salesman problems, *Rom J Inf Sci Tech*, ISSN 1453-8245, 14(1):3-19.

[30] Zhang, G.; Cheng, J.; Gheorghe, M.; Meng, Q. (2013); A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems, *Appl Soft Comput*, ISSN 1568-4946, 13(3):1528-1542.

[31] Zhang, G.X.; Cheng, J.X; Gheorghe, M. (2014); Dynamic behavior analysis of membrane-inspired evolutionary algorithms, *Int J Comput Commun Control*, ISSN 1841-9836, 9(2):227-242.

[32] Zhang, G.; Liu, C.; Rong, H. (2010); Analyzing radar emitter signals with membrane algorithms. *Math Comput Model*, ISSN 0895-7177, 52(11-12):1997-2010.

[33] Zhang, G.; Zhou, F.; Huang, X.; Cheng, J.; Gheorghe, M.; Ipate, F.; Lefticaru, R. (2012); A novel membrane algorithm based on particle swarm optimization for solving broadcasting problems, *J Univers Comput Sci*, ISSN 0948-695x, 18(13):1821-1841.

[34] Zhang, X.; Zeng, X.; Luo, B.; Zhang, Z.(2012); A uniform solution to the independent set problem through tissue P systems with cell separation, *Front Comput Sci*, ISSN 2095-2228, 6(4):477-488.