# Solving Satisfiability Problems with Membrane Algorithms

Gexiang Zhang [#1], Chunxiu Liu [#2], Marian Gheorghe [*3], Florentin Ipate [$4]

[#] *School of Electrical Engineering, Southwest Jiaotong University*
*Chengdu, 610031, P.R. China*
[1] zhgxdylan@126.com
[2] liucx2007@163.com

[*] *Department of Computer Science, The University of Sheffield*
*Regent Court, Portobello Street, Sheffield, S1 4DP, UK*
[3] M.Gheorghe@dcs.shef.ac.uk

[$] *Department of Computer Science and Mathematics, University of Pitesti*
*Str. Targu din Vale 1, 110040 Pitesti, Romania*
[4] florentin.ipate@ifsoft.ro

***Abstract*—This paper presents the application of membrane algorithms to satisfiability problems which are well-known NP-hard combinatorial optimization problems. The membrane algorithm, called QEPS, is a combination of P system approaches and quantum-inspired evolutionary algorithms. QEPS employs the hierarchical structure of the compartments of P systems, the objects consisting of quantum-inspired bit individuals, the rules composed of quantum-inspired gate evolutionary rules and transformation/communication-like rules in P systems to specify the membrane algorithms. A large number of experiments carried out on bench satisfiability problems show that QEPS performs better than its counterpart quantum-inspired evolutionary algorithm.**

## I. Introduction

In the last decades, natural computing is a significant research area in computer science. As a young and vigorous branch of natural computing, membrane computing, using models called P systems, has produced an important impact on the development of various disciplines, such as theoretical computer science [1], biology [2], linguistics [3], etc. P systems, the first version was proposed in 1998 by Gh. Păun [4], are new distributed-parallel frameworks for designing cell-like or tissue-like computing models which handle multisets of abstract objects in a compartmentalized arrangement of membranes [5]. A P system is organized by using a membrane structure, objects and rules. The membrane structure delimits compartments in a hierarchical or network manner. Objects are arranged as multisets and dispersed across these compartments. Rules are provided for specified membranes to control the evolution of objects in a maximally parallel way [6]. 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 for modelling various problems [7]. Until now, P systems have been developed principally from a mathematical point of view, building a variety of computing models for different classes of problems, and have been investigated using experimental methods to solve real-world issues. However the issue of adapting P systems for solving practical problems remains a fundamental aspect and fortunately a burgeoning interest in this respect for many researchers. The application investigation on P systems is still in an extremely preliminary phase, as compared to evolutionary computation [8].

Inspired by the evolution in natural selection and molecular genetics, evolutionary algorithms (EAs) have become the most successful metaheuristic search techniques on the usual computers in natural computing [8],[9]. 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 [10], 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 in the objective function landscapes containing many peaks [11]. Developing efficient EAs for specific problems is a challenging and unfailing topic for the researchers in computer science [12].

Albeit 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 [13], [14]. P systems represent a suitable formal framework for parallel-distributed computation [15] and EAs are very effective for implementing different algorithms to solve lots of problems [9]. Thus, the possible interplay between P systems and EAs, also mentioned by the list of thirty-nine open problems and research topics in membrane computing [6], represents a fertile research field. Being the successful instances of this interaction, membrane algorithms, firstly introduced in [13], can be regarded as a class of hybrid optimization algorithms using the concepts and principles of metaheuristic search

**978-1-4244-3867-9/09/$25.00 ©2009 IEEE**

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 computational load. In [13], [16] and [17], a membrane algorithm using a nested membrane structure (NMS) and a local search method was proposed to solve travelling salesman problems. This membrane algorithm was also applied to solve the min storage problem [18]. In [19] and [20], a membrane algorithm combining NMS and conventional genetic algorithms was presented to solve single-objective and multi-objective numerical optimization problems. In [21], the similarities between distributed EAs and P systems were analyzed and new variants of distributed EAs are suggested and applied for some continuous optimization problems. In our previous work [7], a membrane algorithm integrating one level membrane structure (OLMS) with quantum-inspired evolutionary algorithms (QIEA) was proposed to solve knapsack problems. In [7], the experiment-based comparisons between OLMS and NMS were drawn and suggested that the choice of the membrane structure is very important for membrane algorithms. It is worth pointing out that Gh. Păun and M. J. Pérez-Jiménez [22] made a clear claim that membrane algorithms are rather new research directions with a well-defined practical interest, and therefore further studies are very necessary to prove the use of P systems for real-world applications.

In this paper, the application of membrane algorithms to satisfiability problems, which are well-known NP-hard combinatorial optimization problems, are discussed for the first time. The membrane algorithm uses the hierarchical structure of the compartments of P systems, the objects consisting of quantum-inspired bit individuals, the rules composed of quantum-inspired gate evolutionary rules and transformation / communication-like rules in P systems to specify the membrane algorithms. Extensive experiments carried out on 65 bench satisfiability problems show that the membrane algorithm outperforms its counterpart quantum-inspired evolutionary algorithm. Also, this paper discussed the selection of the number of elementary membranes inside the skin membrane and the use of parametric and non-parametric tests for analysing the membrane's behaviour.

This paper is organized as follows. Section II describes the use of evolutionary computation to solve satisfiability problems. Section III first gives a brief introduction to P systems and QIEA, and then discusses the membrane algorithm in detail. Subsequently, experiments are conducted on various satisfiability problems. Some discussions involving membrane algorithm are addressed in Section IV. Finally, conclusions are drawn in Section 5.

## II. Satisfiability Problems

The satisfiability problem (SAT) is a fundamentally paradigmatic problem in artificial intelligence applications, automated reasoning, mathematical logic, and related research areas [23]. 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., $f(\boldsymbol{x}) = c_1(\boldsymbol{x}) \wedge c_2(\boldsymbol{x}) \wedge \cdots \wedge c_m(\boldsymbol{x})$, 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 [24]. 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 because a number of other problems, such as the travelling salesman problem and the $n$-queens problem, can be reformulated into 3-SAT problems. Cook [25] shows that 3-SAT problem is NP-hard.

In membrane computing, various types of P systems with membrane division are frequently investigated to obtain an exponential working space in a linear time to solve the SAT problem [14],[26],[27], from a mathematical point of view. 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.

## III. Membrane Algorithms

In this section, before the membrane algorithm is described in detail, some concepts related to P systems and QIEA are briefly introduced.

### A. P Systems

At present, there are three main types of P systems: cell-like P systems, tissue-like P systems and neural-like P systems [6]. A cell-like P system is made up of one membrane cell. A tissue-like P system consists of several one-membrane cells in a common environment. Neural-like P systems use neurons as objects. A cell-like P system, considered in this paper, is principally characterized by three ingredients: the membrane structure delimiting compartments, the multisets of abstract objects placed in compartments, and the evolution rules associated to objects or membranes. The membrane structure of a cell-like P system, shown in Fig. 1, is a hierarchical arrangement of membranes [4]. A skin membrane separates the system from its environment. Several membranes, each of which defines a region, are placed inside the skin membrane. An elementary membrane is the one without any membrane inside. Each region forms a different compartment of the membrane structure and contains a multiset of objects, membranes and a set of evolution rules.

A cell-like P system with an output set of objects and using transformation and communication rules is formally defined as follows [4] [5]

$$\Pi = (V, T, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_0),$$

where

1) $V$ is an alphabet; its elements are called objects;
2) $T \subseteq V$ (the output alphabet);

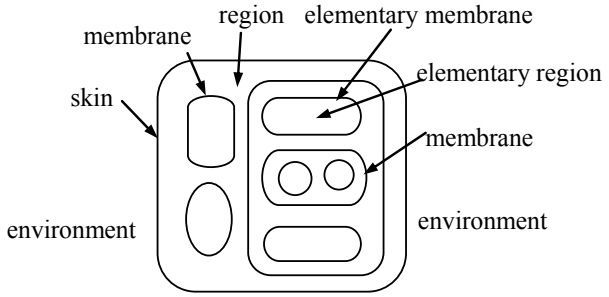


Fig. 1. The membrane structure of a cell-like P system [4]

3) $\mu$ is a membrane structure consisting of $m$ membranes, with the membranes and the regions labelled in a one-to-one manner with elements of a given set $\Lambda$ – usually the set $\{1, 2, \ldots, m\}$; $m$ is called the degree of $\Pi$;
4) $w_i$, $1 \leq i \leq m$, are strings which represent multisets over $V$ associated with the regions 1, 2, $\cdots$, $m$ of $\mu$;
5) $R_i$, $1 \leq i \leq m$, are sets of rules associated to the regions $1, 2, \cdots, m$ of $\mu$;
6) $i_0$ is a number between 1 and $m$ which specifies the output membrane of $\Pi$.

The rules of $R_i$, $1 \leq i \leq m$, have the form $a \rightarrow v$, where $a \in V$ and $v \in (V \times \{here, out, in\})^*$. The multiset $v$ consists of pairs $(b, t)$, $b \in V$ and $t \in \{here, out, in\}$, where $here$ means that $b$ will stay in the region where the rule is applied; $out$ is used to show that $b$ exits the region and $in$ means that $b$ will be communicated to one of the membranes contained in the current region which is chosen in a non-deterministic way.

A P system, regarded as a model of computation, provides a suitable framework for distributed parallel computation that develops in steps. The computation starts by processing the initial multisets, $w_i$, $1 \leq i \leq m$. The initial configuration of a P system is specified by the specific membrane structure and the multisets of objects enclosed in regions. And then the system will go from one configuration to a new one by applying the rules associated to regions in a non-deterministic and maximally parallel manner, i.e., all the objects that may be transformed or communicated must be used. A computation is a sequence of configurations as it is described above, where the final configuration is produced when the system halts under the condition that in any region no more rules are applicable. The result of the computation, a multiset of objects, is obtained in region $i_0$. For more details about P systems definition see [5]. We notice that the rules presented above combine both transformation and communication, but these operations may be separated and then the transformation rules are responsible for evolving the objects and the communication rules will transfer objects among regions according to some targets. The initial multisets of objects may be replaced by strings or multisets of strings, the multiset rewriting by string rewriting and in the output region consists of a set or multiset of strings.

### *B. Quantum-Inspired Evolutionary Algorithms*

The interaction of quantum computing and evolutionary algorithms spouts three branches: evolutionary-designed quantum algorithms (EDQAs) using evolutionary algorithms to design new quantum algorithms [28], quantum evolutionary algorithms (QEAs) implementing evolutionary algorithms in a quantum computing environment [29] and quantum-inspired evolutionary algorithms (QIEAs) [30]. Quantum-inspired is employed to describe the computational methods using concepts and principles of quantum mechanics for solving various problems in the context of a classical computer [31]. Based on the concepts and principles of quantum computing, such as quantum bit (qubit), quantum gate and superposition, QIEA is developed as a novel evolutionary algorithm for a classical computer. Narayanan and Moore [32] introduced QIEA and Han and Kim [30] proposed its practical algorithm. QIEA is characterized by a Q-bit representation, an observation process and a Q-gate evolutionary rule. In recent years, QIEA has become a promising and rapidly growing branch of evolutionary computation. The pseudocode algorithm for QIEA is shown in Fig. 2 and its brief description is as follows.

```
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
```

Fig. 2. Pseudocode algorithm for QIEA [30]

1) In the "initialize $Q(t)$" step, a population $Q(t)$ 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

$$\boldsymbol{q}_i^t = \begin{bmatrix} \alpha_{i1}^t | \alpha_{i2}^t | \cdots | \alpha_{il}^t \\ \beta_{i1}^t | \beta_{i2}^t | \cdots | \beta_{il}^t \end{bmatrix}, \tag{1}$$

where $l$ is the number of Q-bits, i.e., the string length of the Q-bit individual, and $|\alpha_{ij}^t|^2 + |\beta_{ij}^t|^2 = 1$ $(j = 1, 2, \cdots, l)$. In this step, $t = 0$, $\alpha_{ij}^t = \beta_{ij}^t = 1/\sqrt{2}$, which 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 binary bit 0 or 1 is generated. Thus, $l$ binary 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, the Q-bit individuals in $Q(t)$are updated by applying the current Q-gate. In QIEA, the quantum rotation gate is used as a Q-gate; this is given by

$$\boldsymbol{G} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{2}$$

where $\theta$ is the Q-gate rotation angle.
5) The best solutions among $P(t)$ are selected and stored.

In QIEA, 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 (CGAs) with selection, crossover and mutation operators. As compared with local search methods [16]-[18] and CGAs [19]-[21], QIEA has good balance between exploration and exploitation so as to obtain stronger global search capability and better convergence. Furthermore, 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 need to be transmitted and exchanged.

### C. The Membrane Algorithm

In this subsection, the ingredients of the membrane algorithm, QEPS, will be elaborated by using the concepts and mechanism of both P systems and QIEA. In QEPS, a P systems-like framework is introduced to arrange objects and evolution rules; Q-bits, organized as a Q-bit individual which is a special string of Q-bits, are dealt with as multisets of objects; the set of rules responsible to evolve the system and select the best fit Q-bit individuals, consists of evolution rules, observation rules and communication rules.

The pseudocode algorithm for QEPS is shown in Fig. 3 and its detailed description is as follows.

1) In this step, a membrane structure $[_0[_1]_1, [_2]_2, \cdots, [_m]_m]_0$ with $m$ regions contained in the skin membrane denoted by 0 is constructed. The membrane structure is called a one level membrane structure (OLMS) and is shown in Fig. 4.
2) Randomly allocate the $n$ Q-bit individuals of a population $Q(t)$ into the $m$ elementary membranes so that there is at least one individual in each elementary membrane. Thus, the number of individuals in an elementary membrane varies from 1 to $n - m + 1$.
3) This step determines the number of iterations for each elementary membrane to independently perform QIEA. To be specific, the number $g_i$ $(i = 1, 2, \ldots, m)$ of iterations for the $i$th elementary membrane is generated randomly between 1 and a certain integer number such as 10.
4) In each elementary membrane, the evolutionary process shown in Fig. 2 is performed independently. It is worth pointing out that the observation rule occurring in step (2) of QIEA is applied to build a connection between genotypes and phenotypes, and the evolution rule uses the Q-gate update procedure. The Q-gate rotation angle $\theta = s(\alpha, \beta) \cdot \Delta\theta$, where $s(\alpha, \beta)$ and $\Delta\theta$ can be obtained from Table I [30].
5) The communication rules are used to exchange some information among the $m$ regions or between each region and the skin membrane. QIEA employs Q-gates, which are related to only the searched best individual, to generate the offspring. Therefore, QEPS applies the communication rules to send the best fit individual Q-bit

```
Begin
    t ← 0
(i) Initialize the membrane structure;
    While (not termination condition) do
(ii)    Allocate Q-bit individuals for each elementary membrane;
(iii)   Determine the iterations for each elementary membrane;
        For i =1: m
(iv)        Perform QIEA inside the ith elementary membrane;
        End
(v)     Execute communication rules
        t ← t +1
    End
End
```

Fig. 3. Pseudocode algorithm for QEPS

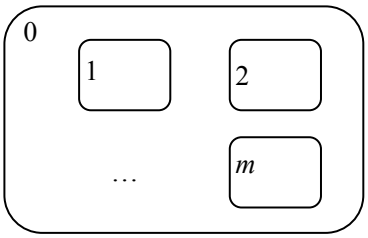


Fig. 4. A one level membrane structure (OLMS)

TABLE I
LOOK-UP TABLE OF $s(\alpha, \beta)$ AND $\Delta\theta$, WHERE $f(.)$ IS THE FITNESS, AND $b$ AND $x$ ARE CERTAIN BITS OF THE CURRENT BEST SOLUTION $\boldsymbol{b}$ AND THE BINARY SOLUTION $\boldsymbol{x}$, RESPECTIVELY [30]

| $x$ | $b$ | $f(\boldsymbol{x}) \geq f(\boldsymbol{b})$ | $\Delta\theta$ | $s(\alpha, \beta)$ |
|---|---|---|---|---|
| 0 | 0 | False | 0 | $\pm 1$ |
| 0 | 0 | True | 0 | $\pm 1$ |
| 0 | 1 | False | $0.01\pi$ | +1 |
| 0 | 1 | True | 0 | $\pm 1$ |
| 1 | 0 | False | $0.01\pi$ | -1 |
| 1 | 0 | True | 0 | $\pm 1$ |
| 1 | 1 | False | 0 | $\pm 1$ |
| 1 | 1 | True | 0 | $\pm 1$ |

representation from each elementary membrane into the skin membrane and the best Q-bit representation from the skin to each compartment.

## IV. Experimental Results

This section uses the QEPS to solve 3-SAT problems. Firstly, it is discussed how to set the number of elementary membranes by using 10 bench SAT problems. Subsequently, another 55 bench SAT problems are applied to draw a comparison between QEPS and its evolutionary algorithm counterpart, QIEA. Finally, several parametric and non-parametric tests are employed to analyse the QEPS' behaviour.

### A. Parameter Setting

This subsection focuses on how to set the number of elementary membranes in an empirical way. Ten bench 3-SAT problems [33], 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 [7], 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 $2.75 \times 10^6$ 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. 5 and Fig. 6, which illustrate that the successful rates and the AES vary with the number of elementary membranes.

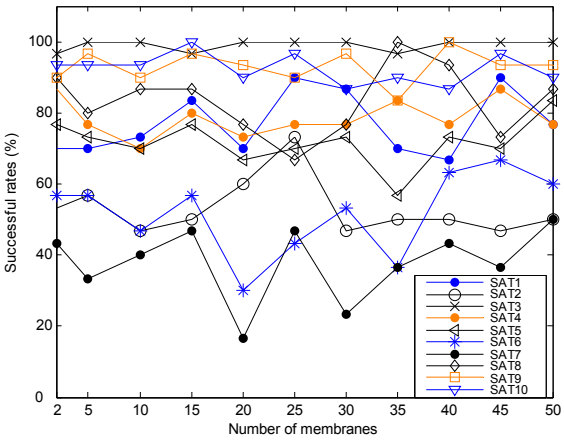


Fig. 5. Successful rates as the number of elementary membranes

As shown in Fig. 5 and Fig. 6, the successful rates and the AES show a broad range of variability with respect to

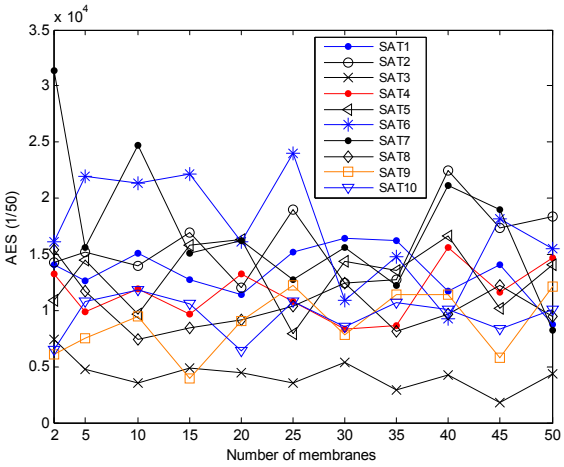


Fig. 6. AES as the number of elementary membranes

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.

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 II. The best experimental results of the QEPS are also shown in Table II. Actually, QIEA is a special version of QEPS with only one elementary membrane in our investigation. Table II shows that the QEPS greatly outperforms QIEA in terms of the successful rates, and furthermore the QEPS has smaller average number of evaluations for half of the SAT problems.

TABLE II
Comparisons of QIEA and QEPS on 10 SAT Problems. NoV, NoC, SR and AES Represent the Number of Boolean Variables, the Number of Clauses, Successful Rates and Average Number of Evaluations to Solutions, Respectively

| | | | QIEA | | QEPS | |
|---|---|---|---|---|---|---|
| Problems | NoV | NoC | SR(%) | AES | SR(%) | AES |
| SAT1 | 20 | 91 | 77 | 572750 | 100 | 528850 |
| SAT2 | | | 70 | 496700 | 90 | 701200 |
| SAT3 | | | 53 | 638250 | 73 | 949400 |
| SAT4 | | | 100 | 218250 | 100 | 90300 |
| SAT5 | | | 87 | 575900 | 87 | 582300 |
| SAT6 | | | 57 | 469650 | 83 | 701700 |
| SAT7 | | | 53 | 1137750 | 67 | 907300 |
| SAT8 | | | 27 | 1120300 | 50 | 410300 |
| SAT9 | | | 87 | 684800 | 100 | 405450 |
| SAT10 | | | 93 | 281050 | 100 | 572750 |

### B. Comparisons and Statistical Analysis

To further test the performances of the QEPS and draw a convincing comparison between QEPS and QIEA, additional 55 3-SAT benchmark problems [33] are employed to carry out experiments in this subsection.

Both QEPS and QIEA use 50 individuals as a population, the prescribed number of $2.75 \times 10^6$ 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 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 III and Table IV.

TABLE III
COMPARISONS OF QIEA AND QEPS USING 55 SAT PROBLEMS. NOV, NOC, MEAN AND STD REPRESENT THE NUMBER OF BOOLEAN VARIABLES, THE NUMBER OF CLAUSES, THE MEAN OF BEST SOLUTIONS AND THE STANDARD DEVIATION OF BEST SOLUTIONS, RESPECTIVELY. + AND - REPRESENT SIGNIFICANT DIFFERENCE AND NO SIGNIFICANT DIFFERENCE, RESPECTIVELY (TO BE CONTINUED)

| | | | QIEA | | QEPS | | | Imp. |
|---|---|---|---|---|---|---|---|---|
| SAT | NoV | NoC | Mean | Std | Mean | Std | *t*-test | (%) |
| 1 | 50 | 218 | 7.67 | 0.82 | 6.60 | 1.18 | 0.00766(+) | +13.91 |
| 2 | | | 8.27 | 2.12 | 7.40 | 1.12 | 0.17264(-) | +10.48 |
| 3 | | | 7.40 | 1.40 | 6.27 | 0.88 | 0.01322(+) | +15.32 |
| 4 | | | 7.87 | 1.19 | 6.33 | 0.98 | 0.00060(+) | +19.49 |
| 5 | | | 7.13 | 1.41 | 6.20 | 0.86 | 0.03700(+) | +13.08 |
| 6 | | | 7.27 | 0.80 | 5.93 | 0.80 | 0.00009(+) | +18.35 |
| 7 | | | 7.73 | 1.16 | 6.40 | 1.18 | 0.00424(+) | +17.24 |
| 8 | | | 8.73 | 0.70 | 7.67 | 1.18 | 0.00540(+) | +12.21 |
| 9 | | | 7.87 | 1.13 | 6.87 | 1.19 | 0.02505(+) | +12.71 |
| 10 | | | 8.33 | 0.82 | 6.93 | 0.88 | 0.00011(+) | +16.80 |
| 11 | 75 | 325 | 16.67 | 1.11 | 15.40 | 0.99 | 0.00264(+) | +7.60 |
| 12 | | | 15.07 | 1.49 | 14.60 | 0.74 | 0.28525(-) | +3.10 |
| 13 | | | 16.33 | 0.82 | 14.80 | 2.01 | 0.01056(+) | +9.39 |
| 14 | | | 14.00 | 1.20 | 12.87 | 1.60 | 0.03623(+) | +8.10 |
| 15 | | | 15.07 | 1.03 | 14.87 | 0.92 | 0.57909(-) | +1.33 |
| 16 | | | 16.13 | 1.06 | 14.87 | 1.19 | 0.00458(+) | +7.85 |
| 17 | | | 15.33 | 1.40 | 14.53 | 1.64 | 0.16174(-) | +5.22 |
| 18 | | | 15.73 | 1.44 | 14.53 | 1.51 | 0.03375(+) | +7.63 |
| 19 | | | 14.93 | 1.49 | 13.80 | 1.97 | 0.08628(-) | +7.59 |
| 20 | | | 14.40 | 1.45 | 13.93 | 1.22 | 0.34958(-) | +3.24 |

According to these experimental results, we employ statistical techniques to analyse the two algorithms' behaviour over the 55 SAT problems. There are two ways of statistical methods: parametric and non-parametric [34]. The former, also called single-problem analysis, uses a parametric statistical analysis $t$-test to analyse whether there is a significant difference over one optimization problem between two algorithms. The latter, also called multiple-problem analysis, applies non-parametric statistical tests such as Wilcoxon's and Friedman's

TABLE IV
COMPARISONS OF QIEA AND QEPS USING 55 SAT PROBLEMS. NOV, NOC, MEAN AND STD REPRESENT THE NUMBER OF BOOLEAN VARIABLES, THE NUMBER OF CLAUSES, THE MEAN OF BEST SOLUTIONS AND THE STANDARD DEVIATION OF BEST SOLUTIONS, RESPECTIVELY. + AND - REPRESENT SIGNIFICANT DIFFERENCE AND NO SIGNIFICANT DIFFERENCE, RESPECTIVELY (CONTINUED)

| | | | QIEA | | QEPS | | | Imp. |
|---|---|---|---|---|---|---|---|---|
| SAT | NoV | NoC | Mean | Std | Mean | Std | *t*-test | (%) |
| 21 | 100 | 430 | 24.53 | 1.81 | 22.93 | 1.39 | 0.01109(+) | +6.52 |
| 22 | | | 24.20 | 1.21 | 22.80 | 2.14 | 0.03598(+) | +5.79 |
| 23 | | | 24.27 | 1.28 | 22.93 | 1.79 | 0.02632(+) | +5.49 |
| 24 | | | 23.40 | 1.68 | 22.80 | 1.08 | 0.25509(-) | +2.56 |
| 25 | | | 24.27 | 1.22 | 22.80 | 1.47 | 0.00610(+) | +6.04 |
| 26 | | | 24.00 | 1.51 | 22.47 | 1.60 | 0.01163(+) | +6.39 |
| 27 | | | 24.13 | 1.06 | 23.40 | 1.35 | 0.10951(-) | +3.04 |
| 28 | | | 24.00 | 1.85 | 23.40 | 1.30 | 0.31296(-) | +2.50 |
| 29 | | | 25.13 | 1.68 | 24.13 | 2.00 | 0.14921(-) | +3.98 |
| 30 | | | 22.93 | 2.15 | 22.27 | 1.98 | 0.38507(-) | +2.91 |
| 31 | 125 | 538 | 33.53 | 1.96 | 32.87 | 1.51 | 0.30496(-) | +1.99 |
| 32 | | | 33.80 | 1.90 | 32.07 | 2.12 | 0.02551(+) | +5.13 |
| 33 | | | 34.47 | 1.81 | 33.73 | 1.33 | 0.21659(-) | +2.13 |
| 34 | | | 34.06 | 1.84 | 33.27 | 2.34 | 0.09438(-) | +3.85 |
| 35 | | | 34.67 | 1.76 | 33.60 | 2.20 | 0.15336(-) | +3.08 |
| 36 | | | 34.80 | 2.21 | 33.93 | 1.44 | 0.21348(-) | +2.49 |
| 37 | | | 32.80 | 2.86 | 32.93 | 1.33 | 0.87115(-) | -0.41 |
| 38 | | | 32.80 | 1.97 | 32.47 | 1.19 | 0.57924(-) | +1.02 |
| 39 | | | 34.20 | 2.08 | 33.40 | 1.76 | 0.26526(-) | +2.34 |
| 40 | | | 33.93 | 1.67 | 33.20 | 2.34 | 0.33087(-) | +2.16 |
| 41 | 150 | 645 | 42.60 | 1.50 | 41.20 | 2.01 | 0.03926(+) | +3.29 |
| 42 | | | 43.07 | 1.67 | 40.00 | 2.67 | 0.00078(+) | +7.12 |
| 43 | | | 41.73 | 1.71 | 41.53 | 1.06 | 0.70314(-) | +0.48 |
| 44 | | | 42.80 | 3.28 | 41.07 | 1.94 | 0.08907(-) | +4.05 |
| 45 | | | 43.93 | 1.67 | 42.60 | 2.64 | 0.10938(-) | +3.03 |
| 46 | | | 43.00 | 3.23 | 42.07 | 1.62 | 0.32587(-) | +2.17 |
| 47 | | | 43.20 | 2.04 | 42.73 | 1.87 | 0.51924(-) | +1.08 |
| 48 | | | 44.27 | 2.19 | 43.60 | 2.32 | 0.42522(-) | +1.51 |
| 49 | | | 44.67 | 2.26 | 43.53 | 2.10 | 0.16557(-) | +2.54 |
| 50 | | | 43.13 | 1.55 | 41.47 | 2.50 | 0.03690(+) | +3.86 |
| 51 | 250 | 1065 | 83.07 | 3.45 | 81.27 | 2.91 | 0.13411(-) | +2.17 |
| 52 | | | 83.40 | 3.44 | 82.73 | 2.96 | 0.57406(-) | +0.80 |
| 53 | | | 83.00 | 2.73 | 81.60 | 2.50 | 0.15388(-) | +1.69 |
| 54 | | | 85.13 | 3.23 | 83.60 | 3.14 | 0.19750(-) | +1.80 |
| 55 | | | 84.87 | 2.83 | 80.80 | 2.31 | 0.00018(+) | +4.79 |

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 algorithms is significantly different or not. Furthermore, the percentage of improvement ($\%$) in the average number of false clauses due to the QEPS algorithm over QIEA is also listed in Table III and Table IV. And then two non-parametric tests, Wilcoxon's and Friedman's tests, are employed to check whether there are significant differences between QEPS and QIEA. The level of significance considered is 0.05. The results of Wilcoxon's and Friedman's tests are shown in Table V. In Tables III, IV and V, the symbols + and - represent significant difference and no significant difference, respectively.

As shown in Tables III and IV, the QEPS achieves better results than the QIEA in 54 out of 55 cases. The $t$-test results demonstrate that there are 24 significant differences between the two algorithms. The $p$-values of the two non-parametric tests in Table V are far smaller than the level of significance 0.05, which indicates that the QEPS really outperforms the QIEA by introducing the framework and some rules of P systems. It is worth noting that the study in [34] shows that the non-parametric statistical tests are more appropriate than parametric statistical tests in the analysis of evolutionary algorithms' behaviour over multiple optimization problems.

TABLE V
RESULTS OF NON-PARAMETRIC STATISTICAL TESTS FOR QEPS AND QIEA IN TABLES III AND IV. + AND - REPRESENT SIGNIFICANT DIFFERENCE AND NO SIGNIFICANT DIFFERENCE, RESPECTIVELY

| Tests | QEPS vs QIEA |
|---|---|
| Wilcoxon test ($p$-value) | 1.17e-010 (+) |
| Friedman test (p-value) | 8.90e-013 (+) |

## V. CONCLUSION

Membrane algorithms, formed by carefully choosing ingredients of P systems and metaheuristic search methodologies, and the interaction between P systems and quantum computing, are highly promising and challenging research issues, which are also mentioned as open problems and research topics in [6]. As an instance of the cross-domains of P systems, evolutionary computation and quantum computing, this paper discussed the novel membrane algorithm combing P systems and QIEA to solve satisfiability problems. A large number of experiments show that QEPS performs better than QIEA. As further work, we will discuss various interplays of the three disciplines and their applications to specific problems

## ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments. The research of GZ is supported by the National Natural Science Foundation of China (60702026), the Scientific and Technological Funds for Young Scientists of Sichuan (09ZQ026-040) and the Open Foundation of Engineering Research Centre of Safety Transportation of the Ministry of Education of China. The research of MG and FI is supported by CNCSIS grant no.643/2009, *An integrated evolutionary approach to formal modelling and testing*.

## REFERENCES

[1] G. Păun and M. J. Pérez-Jiménez, "Fourth brainstorming week on membrane computing," *Theoretical Computer Science*, vol. 372, pp. 123–124, Mar. 2007.
[2] M. Gheorghe, N. Krasnogor, and M. Camara, "P systems applications to systems biology," *BioSystems*, vol. 91, pp. 435–437, Mar. 2008.
[3] G. Ciobanu, G. Păun, and M. J. Pérez-Jiménez, Eds., *Applications of membrane computing*, ser. Natural Computing. Berlin, Germany: Springer-Verlag, 2006.
[4] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, pp. 108–143, Aug. 2000.
[5] G. Păun and G. Rozenberg, "A guide to membrane computing," *Theoretical Computer Science*, vol. 287, pp. 73–100, Sep. 2002.
[6] G. Păun, "Tracing some open problems in membrane computing," *Romanian Journal of Information Science and Technology*, vol. 10, pp. 303–314, 2007.
[7] G. X. Zhang, M. Gheorghe, and C. Z. Wu, "A quantum-inspired evolutionary algorithm based on p systems for a class of combinatorial optimization," *Fundamenta Informaticae*, vol. 87, pp. 93–116, Nov. 2008.
[8] G. Păun, "Bio-inspired computing paradigms (natural computing)," in *Proc. UPP'2004*, ser. Lecture Notes in Computer Science, J. P. Banâtre *et al.*, Eds. Berlin Heidelberg: Springer-Verlag, 2005, vol. 3566, pp. 155–160.
[9] P. P. Bonissone, R. Subbu, N. Eklund, and T. R. Kiehl, "Evolutionary algorithms + domain knowledge = real-world evolutionary computation," *IEEE Trans. Evol. Comput.*, vol. 10, pp. 256–280, Jun. 2006.
[10] D. Whitley, "An overview of evolutionary algorithms practical issues and common pitfalls," *Information and Software Technology*, vol. 43, pp. 817–831, Dec. 2001.
[11] T. Jones and S. Forrest, "Genetic algorithms and heuristic search," Santa Fe Institute, Tech. Rep., 1995.
[12] X. Yao, "An overview of evolutionary computation," *Chinese Journal of Advanced Software Research*, vol. 3, pp. 12–29, 1996.
[13] T. Y. Nishida, "An approximate algorithm for np-complete optimization problems exploiting p systems," in *Proc. Brainstorming Workshop on Uncertainty in Membrane Computing*, Palma de Mallorca, Nov. 2004, pp. 185–192.
[14] G. Păun, "P systems with active membranes: attacking np-complete problems," *Journal of Automata Language Combinatorics*, vol. 6, pp. 75–90, Jan. 2001.
[15] G. Ciobanu, "Distributed algorithms over communicating membrane systems," *BioSystems*, vol. 70, pp. 123–133, Jul. 2003.
[16] T. Y. Nishida, "Application of membrane computing: Membrane algorithms: an approximate algorithm for np-complete optimization problems," in *[3]*, pp. 303–314.
[17] ——, "Membrane algorithms," in *Proc. WMC 2005*, ser. Lecture Notes in Computer Science, vol. 3566. Berlin Heidelberg: Springer-Verlag, 2006, pp. 55–66.
[18] A. Leporati and D. Pagani, "A membrane algorithm for the min storage problem," in *Proc. WMC 2006*, ser. Lecture Notes in Computer Science, vol. 4361. Berlin Heidelberg: Springer-Verlag, 2006, pp. 443–462.
[19] L. Huang, X. X. He, N. Wang, and Y. Xie, "P systems based multi-objective optimization algorithm," *Progress in Natural Science*, vol. 17, pp. 458–465, 2007.
[20] L. Huang and N. Wang, "An optimization algorithm inspired by membrane computing," in *Proc. ICNC 2006*, ser. Lecture Notes in Computer Science, vol. 4222. Berlin Heidelberg: Springer-Verlag, 2006, pp. 49–52.
[21] D. Zaharie and G. Ciobanu, "Distributed evolutionary algorithms inspired by membranes in solving continuous optimization problems," in *Proc. WMC 2006*, ser. Lecture Notes in Computer Science, vol. 4361. Berlin Heidelberg: Springer-Verlag, 2006, pp. 536–553.
[22] G. Păun and M. J. Pérez-Jiménez, "Membranes computing: brief introduction, recent results and applications," *BioSystem*, vol. 85, pp. 11–22, Jul. 2006.

[23] G. Folino, C. Pizzuti, and G. Spezzano, "Parallel hybrid method for sat that couples genetic algorithms and local search," *IEEE Trans. Evol. Comput.*, vol. 5, pp. 323–334, Aug. 2001.
[24] J. Gottlieb, E. Marchiori, and C. Rossi, "Evolutionary algorithms for the satisfiability problem," *Evolutionary Computation*, vol. 10, pp. 35–50, Spring 2002.
[25] S. Cook, "The complexity of theorem-proving procedures," in *Proc. Third Annual ACM Symposium on Theory of Computing*, Shaker Heights, Ohio, United States, May 1971, pp. 151–158.
[26] A. Alhazov, C. Martin-Vide, and L. Pan, "Solving a pspace-complete problem by p systems with restricted active membranes," *Fundamenta Informaticae*, vol. 58, pp. 67–77, Apr. 2003.
[27] L. Q. Pan and A. Alhazov, "Solving hpp and sat by p systems with active membranes and separation rules," *Acta Informatica*, vol. 43, pp. 131–145, Spring 2006.
[28] I. Grigorenko and M. E. Garcia, "Calculation of the partition function using quantum genetic algorithms," *Physica A: Statistical Mechanics and its Applications*, vol. 313, pp. 463–470, 2002.
[29] M. S. Yee and L. Hanzo, "Implementing quantum genetic algorithms: a solution based on grover's algorithm," in *Proc. the 3rd Conference on Computing Frontiers*, Ischia, Italy, May 2006, pp. 14–16.
[30] K. H. Han and J. H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 580–593, Dec. 2002.
[31] M. Moore and A. Narayanan, "Quantum-inspired computing," Univ. of Exeter, Tech. Rep. Exeter 344, Nov. 1995.
[32] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in *Proceedings of the 1996 IEE InternationalConference on Evolutionary Computation*. IEEE Press, 1995, pp. 61–66.
[33] Benchmark. [Online]. Available: http://www.satlib.org/
[34] S. Garcia, D. Molina, M. Lozano, and F. Herrera, "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," *Journal of Heuristics*, vol. 87, doi: 10.1007/s10732-008-9080-4.