A Quantum-Inspired Evolutionary Algorithm Based on P systems for a Class of Combinatorial Optimization

Gexiang Zhang[1,2], Marian Gheorghe[2], Chaozhong Wu[3]

[1] School of Electrical Engineering, Southwest Jiaotong University, Chengdu, Sichuan, 610031, P.R. China
E-mail: zhgxdylan@126.com
[2] Department of Computer Science, The University of Sheffield, Regent Court, Portobello Street, Sheffield S1 4DP, UK
E-mail: M.Gheorghe@dcs.shef.ac.uk
[3] Engineering Research Center of Transportation Safety, Ministry of Education, Wuhan University of Technology, Wuhan, Hubei, 430063, P.R. China
E-mail: chaozhongwu@gmail.com

Summary. This paper introduces an evolutionary algorithm which uses the concepts and principles of the quantum-inspired evolutionary approach and the hierarchical arrangement of the compartments of a P system. The P system framework is also used to formally specify this evolutionary algorithm. Extensive experiments are conducted on a well-known combinatorial optimization problem, the knapsack problem, to test the effectiveness of the approach. These experimental results show that this evolutionary algorithm performs better than quantum-inspired evolutionary algorithms, for certain arrangements of the compartments of the P system structure utilized.

1 Introduction

Evolutionary algorithms (EAs) are practical and robust optimization and search methods inspired by evolutionary processes occurring in natural selection and molecular genetics. The main features of EAs are the representation and evaluation of individuals, population dynamics, evolutionary operators such as selection, crossover and mutation [18, 4]. As compared to conventional optimization methods, EAs are more suitable for solving complex optimization problems as they exhibit an intrinsic parallelism derived from dealing with multiple individuals, show remarkable adaptability and flexibility to application problems, good search capability and robust results [1].

As a new distributed-parallel computing model, membrane computing, also known as P system, has been introduced in [13] as an unconventional,

nature-inspired computational paradigm employing various features specific to the structure and functionality of the living cell. A P system consists of membranes delimiting compartments organized in a hierarchical structure. Objects are scattered across this structure and organized as multisets; specific rules are applied in parallel to these objects in every compartment [13, 15]. The hierarchical structure of membranes, type of rules (transformation, communication etc), intrinsic parallelism, are all very effective from a computational point of view and attractive for modelling various problems.

P systems and EAs are different with respect to the objects and rules used, computational strategies employed, but both are nature-inspired models and applied to solve complex problems, e.g., NP complete problems [9, 14]. P systems represent a suitable formal framework for parallel-distributed computation [2, 20] and EAs are very effective for implementing different algorithms to solve numerous problems [1]. Thus, the possible interaction between P systems and EAs, also mentioned by the list of twenty-six open problems in membrane computing [16], represents a fertile research field. In [9, 10, 11] it is proposed a membrane-based evolutionary algorithm combining a membrane structure where each membrane, but the deepest one, contains one membrane and a local search method. This membrane-based algorithm was also employed to solve the min storage problem [7]. In [5, 6] a hybrid algorithm combining P systems and genetic algorithms was presented to solve single-objective and multi-objective numerical optimization problems. In [20], 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.

This paper proposes a novel EA, called quantum-inspired evolutionary algorithm based on P systems (QEPS), which uses the concepts and principles of quantum-inspired evolutionary algorithms (QIEAs) within a P system framework. A quantum-inspired bit (Q-bit) representation and quantum-inspired gate (Q-gate) evolutionary rules together with a hierarchical membrane structure and transformation/communication-like rules are employed. To demonstrate the effectiveness and applicability scope of this approach, a large number of experiments are carried out for the knapsack problem, a well-known combinatorial optimization problem. The results obtained show that QEPS with a specific membrane structure performs better than known QIEAs.

The knapsack problem has been frequently analysed by both P systems and evolutionary algorithms communities as test-benches for various approaches. Recognizer P systems with active membranes were constructed to solve one-dimensional [17] and multi-dimensional [12] knapsack problems in linear time. Also, extensively convincing experiments show that QIEA is far better than conventional genetic algorithms [3, 4] in solving the same problem.

This paper is organized as follows. Section 2 introduces briefly QIEA and P systems, and then describes QEPS in detail. Section 3 presents an application example comparing QEPS and QIEAs for knapsack problems and summarizes

the experimental results. Section 4 discusses the parameter setting in QEPS and different membrane structures. Concluding remarks follow in Section 5.

2 QEPS – Basic Concepts

This section starts with brief introductions about QIEA and P systems goes then into presenting a P systems-like framework of QIEA, called QEPS.

2.1 QIEA

Inspired by concepts of quantum computing such as quantum bits and quantum gate, QIEA is a new evolutionary algorithm for a classical computer instead of a quantum algorithm. QIEA was first introduced by Narayanan and Moore in [8] and its practical form was proposed by Han and Kim in [3]. Various variants of QIEA can be categorized into two groups: real observation QIEA for numerical optimization [22] and binary observation QIEA (bQIEA) for combinatorial optimization. The latter, referred in this paper, can be sub-classified into four groups: original bQIEA (bQIEAo) [3], bQIEA with migration operator (bQIEAm) [4], bQIEA with a combination of crossover, mutation and selection operators (bQIEAcms) [19], and bQIEA with a novel update method for Q-gates (bQIEAn) [21]. QIEA approach is characterized by a Q-bit representation for individuals and a Q-gate as a variation operator to obtain better fitted individuals. In QIEA, a Q-bit is defined by a pair of numbers (α, β) as

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad , \tag{1}$$

where $|\alpha|^2$ and $|\beta|^2$ are the probabilities that the observation of a Q-bit will render a '0' or '1' state [4]. Normalization requires that $|\alpha|^2 + |\beta|^2 = 1$. Note that QIEA just needs real numbers for probability 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

$$\begin{bmatrix} \alpha_1 | \alpha_2 | \cdots | \alpha_l \\ \beta_1 | \beta_2 | \cdots | \beta_l \end{bmatrix} \quad , \tag{2}$$

where $|\alpha_i|^2 + |\beta_i|^2 = 1$ $(i = 1, 2, \cdots, l)$. A Q-gate in QIEA is defined as a variation operator for updating the Q-bit individuals such as to guarantee that they also satisfy the normalization condition $|\alpha|^2 + |\beta|^2 = 1$ [4]. To date, QIEA principally adopts quantum rotation gate as a Q-gate, as it is shown in Eq. (4).

The basic pseudocode algorithm for bQIEA is shown in Fig. 1. Each step in Fig. 1 is described 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. 1. Pseudocode algorithm for bQIEA [17]

(i) 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 $Q(t)$ and $\boldsymbol{q}_i^t$ is represented as

$$\boldsymbol{q}_i^t = \begin{bmatrix} \alpha_{i,1}^t | \alpha_{i,2}^t | \cdots | \alpha_{i,l}^t \\ \beta_{i,1}^t | \beta_{i,2}^t | \cdots | \beta_{i,l}^t \end{bmatrix} \quad , \tag{3}$$

where l is the number of Q-bits, i.e., the string length of the Q-bit individual. $\alpha_{i,j}^t = \beta_{i,j}^t = 1/\sqrt{2}$ $(j = 1, 2, \cdots, l)$. This means that all possible states are superposed with the same probability at the beginning.

(ii) 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 generated at step t. According to the current probability, either $|\alpha_i^t|^2$ or $|\beta_i^t|^2$ of $\boldsymbol{q}_i^t$ $(i = 1, 2, \cdots, l)$, a binary bit 0 or 1 is generated. Thus, a binary solution $\boldsymbol{x}_j^t$ $(j = 1, 2, \cdots, n)$ consists of l binary bits.

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

(iv) In this step, the Q-bit individuals in $Q(t)$ are updated by applying the current Q-gate. In bQIEA, the quantum rotation gate is used as a Q-gate; this is given by

$$\boldsymbol{G(t)} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad , \tag{4}$$

where θ is the Q-gate rotation angle.

(v) The best solutions among $P(t)$ are selected and stored.

Compared to local search methods [9, 10, 11, 7] and conventional genetic algorithms [20, 5, 6], QIEA has several special characteristics. Firstly, the Q-bit encoding can represent probabilistically a linear superposition of states in the search space, which makes QIEA rather good with respect to population

diversity. Secondly, with a small number of individuals, even with one individual, QIEA can exploit the search space for a global solution within a short span of time. Thirdly, the evolutionary rules are very simple, instead of selection, crossover and mutation operators, QIEA uses only a Q-gate operation, which is related to the current best Q-bit individual in the population.

2.2 P Systems

The membrane structure of a P system, shown in Fig. 2, is a hierarchical arrangement of membranes, embedded in the skin membrane, the one which separates the system from its environment [15]. A membrane without any membrane inside is called an elementary one. Each membrane defines a region. Each region constitutes a different compartment of the membrane structure and contains a multiset of objects and a set of transformation and communication rules.

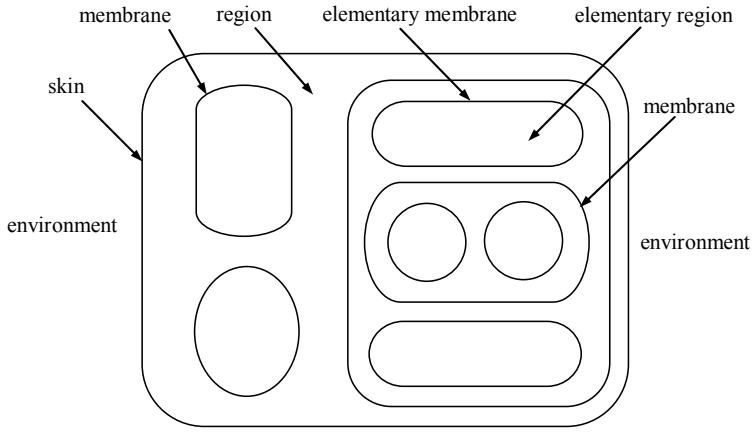


Fig. 2. A membrane structure [2]

The multisets associated to regions form a configuration of a P system. 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 processed. The system will halt when no more rules are available to be applied. A computation is a sequence of configurations obtained as it is described above, where the initial configuration consists of the initial multisets associated to regions and the final one is generated when the system halts. The result of a computation is obtained in the region defined by the output membrane.

In what follows a basic P system with an output set of objects and using transformation and communication rules is formally defined. Let us consider a construct [13, 15]

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

where

(i) V is an alphabet; its elements are called objects;
(ii) $T \subseteq V$ (the output alphabet);
(iii) μ 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 Λ – usually the set $\{1, \ldots, m\}$; m is called the degree of Π;
(iv) w_i, $1 \leq i \leq m$, are strings which represents multisets over V associated with the regions $1, 2, \cdots, m$ of μ;
(v) R_i, $1 \leq i \leq m$, are sets of rules associated to the regions $1, 2, \cdots, m$ of μ;
(vi) i_0 is a number between 1 and m which specifies the output membrane of Π.

The rules of R_i, $1 \leq i \leq m$, have the form $a \to v$, 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 rules are 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 provides a suitable framework for distributed parallel computation that develops in steps. Indeed, any computation starts by processing the initial multisets, w_i, $1 \leq i \leq m$, and then in each step the rules associated to each region are applied in a non-deterministic and maximally parallel manner. The computation, a multiset of simple objects, is obtained in region i_0. For more details about P systems definition see [15]. 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 simple objects may be replaced by strings or multisets of strings, the multiset rewriting by string rewriting and in the output region obtain a set or multiset of strings.

2.3 QEPS

This section will introduce a P systems-like framework that will help presenting the QEPS algorithm. This framework will use some of the elements of a P system, but others will be used in a rather metaphoric way. A specific membrane structure will be initially introduced, but this will be later on changed. 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 evolve the system and select the best fit Q-bit individuals.

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

(i) a membrane structure $[_1[_2]_2, [_3]_3, \cdots, [_{m+1}]_{m+1}]_1$ with m regions contained in the skin membrane, denoted by 1;
(ii) a vocabulary that consists of all the Q-bits and the set $\{0, 1\}$;
(iii) a set of terminal symbols, T, consisting of all the Q-bits;
(iv) initial multisets $w_1 = \lambda$,
$w_2 = q_1 q_2 \cdots q_{n_1}, n_1 \leq n$,
$w_3 = q_{n_1+1} q_{n_1+2} \cdots q_{n_2}, n_1 + n_2 \leq n$,
$\cdots \cdots$
$w_{m+1} = q_{n_{(m-1)}+1} q_{n_{(m-1)}+2} \cdots q_{n_m}$, $n_1 + n_2 + \cdots n_m \leq n$, where
q_i, $1 \leq i \leq n$, is a Q-bit individual;
(v) rules which are classified as
(a) evolution rules in each of the compartments 2 to $m + 1$; these are transformation-like rules which update a Q-bit individual according to the current Q-gate (see (iv) of the QIEA presentation);
(b) mapping rules which make binary solutions from Q-bit individuals (see (ii) of the QIEA presentation and algorithm in Fig. 3);
(c) communication rules which send the best fit individual binary representation from each of the m regions into the skin membrane and then the overall best binary representation from the skin back to each region.

In QEPS the initial population of Q-bit individuals is scattered across the membrane structure. The initial population will consist of the multisets $w_2, \cdots, w_{m+1}$. In any step the current generation is assessed compartment by compartment to select the best fit individual (applying rules of type (b)). The best solution is used to adjust the Q-gate which is then employed to produce the next generation by applying evolution rules. Every $g_i (1 \leq i \leq m)$ generations for each compartment, the communication rule is performed once. The process will stop when the best fit solution will remain unchanged for several generations.

```
Begin
   j ← 1
   While (j < l) do
      If random[0,1) < |β_ij^t|^2
      Then x_j ← 1
      Else x_j ← 0
   End
End
```

Fig. 3. Pseudocode algorithm for the mapping rule [4]

3 Application Example

In this section, the QEPS algorithm for the knapsack problem is presented in detail. The knapsack problem is applied to show the effectiveness of QEPS to a combinatorial optimization problem. To make a comparison, three types of QEPS and four types of QIEA are considered. The knapsack problem can be described as selecting from among various items those that are the most profitable, given that the knapsack has a limited capacity [4]. The knapsack problem requires to select a subset of a given set of items so as to maximize a profit function

$$f(x) = \sum_{i=1}^{k} p_i x_i \tag{5}$$

Subject to

$$\sum_{i=1}^{k} r_i x_i \leq C_a \tag{6}$$

where k is the number of items; p_i is the profit of the ith item; r_i is the weight of the ith item; C_a is the capacity of the given knapsack; and x_i is 0 or 1.

Table 1. Lookup table of θ, where $f(.)$ is the fitness, $s(\alpha, \beta)$ is the sign of θ, and b and x are certain bits of the current best solution $\boldsymbol{b}$ and the binary solution $\boldsymbol{x}$, respectively [3]

				$s(\alpha, \beta)$			
x	b	$f(\boldsymbol{x} \geq f(\boldsymbol{b})$	$\Delta\theta$	$\alpha\beta > 0$	$\alpha\beta < 0$	$\alpha = 0$	$\beta = 0$
0	0	False	0	0	0	0	0
0	0	True	0	0	0	0	0
0	1	False	0	0	0	0	0
0	1	True	0.05π	-1	+1	± 1	0
1	0	False	0.01π	-1	+1	± 1	0
1	0	True	0.025π	+1	-1	0	± 1
1	1	False	π	+1	-1	0	± 1
1	1	True	π	+1	-1	0	± 1

3.1 QEPS for the Knapsack Problem

In this paper, we consider three types of QEPS based on various QIEA approaches. Consequently, we have QEPS with different Q-gate update methods, namely used by bQIEAo (QEPSo), bQIEAm (QEPSm), and bQIEAn

(QEPSn). The three variants of QEPS use different methods for deriving the rotation angle θ in $G(t)$, where θ is defined as $\theta = s(\alpha, \beta)\Delta\theta$, where $s(\alpha, \beta)$ and $\Delta\theta$ are the sign and the value of θ, respectively. $s(\alpha, \beta)$ and $\Delta\theta$ can be obtained by using Table 1 for QEPSo, Table 2 for QEPSm and Table 3 for QEPSn, respectively.

Table 2. Lookup table of θ, where $f(.)$ is the fitness, $s(\alpha, \beta)$ is the sign of θ, and b and x are certain bits of the current best solution $\boldsymbol{b}$ and the binary solution $\boldsymbol{x}$, respectively [4]

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

Table 3. Look-up table of θ, where $d_1 = \alpha_1\beta_1$, $\xi_1 = \arctan(\beta_1/\alpha_1)$, α_1, β_1 are the probability amplitudes of the current best solution, and $d_2 = \alpha_2\beta_2$, $\xi_2 = \arctan(\beta_2/\alpha_2)$, α_2, β_2 are the probability amplitudes of the current solution, and $e = 0.5\pi||\alpha_1| - |\alpha_2||$

			$f(\alpha, \beta)$	
$d_1 > 0$	$d_2 > 0$	$\Delta\theta$	$\lvert\xi_1\rvert \geq \lvert\xi_2\rvert$	$\lvert\xi_1\rvert < \lvert\xi_2\rvert$
True	Ture	e	+1	-1
True	False	e	-1	+1
False	True	e	-1	+1
False	False	e	+1	-1

3.2 QIEA for the Knapsack Problem

Any QIEA for the knapsack problem consists of a basic structure (see Fig. 1) and a repair process to match the capacity constraint, illustrated by Eq.

6. The pseudocode algorithm for the repair process is shown in Fig. 4. Four types of QIEA are described and tested for the knapsack problem: bQIEAo [3], bQIEAm [4], bQIEAcms [19] and bQIEAn [21]. The four algorithms can be regarded as special kinds of QEPS with a skin membrane and one elementary membrane. The pseudocode algorithm for bQIEAo is illustrated in Fig. 1; bQIEAo uses Table 1 as a look-up table for determining the rotation angle of the Q-gate. bQIEAm is an improved version of the bQIEAo algorithm that uses inserting migration operation in the bQIEAo algorithm; this uses Table 2 to decide the rotation angle of the Q-gate. A modified algorithm obtained by adding selection, quantum crossover and quantum mutation operators to bQIEAo, and using the same method for determining the rotation angle of the Q-gate as bQIEAo is represented by the bQIEAcms algorithm. Lastly the bQIEAn algorithm appears as a modified version of the bQIEAo algorithm by introducing a modified update method for the Q-gate, whose rotation angle is changed through the look up Table 3. Extensively convincing comparisons between bQIEAm and conventional genetic algorithms show the advantages of bQIEAm. In this paper we will start from these results.

3.3 Experimental Results

In the following experiments, strongly correlated sets of unsorted data are used

r_i=uniformly random [1, 10]
p_i=r_i+5

and the average knapsack capacity

$$C_a = \frac{1}{2}\sum_{i=1}^{k} r_i \quad .$$

Three knapsack problems with 200, 400, and 600 items are considered. Because r_i is a random value, the experimental results in [4] and [3] cannot be referenced directly in this paper.

For the seven algorithms, the population size is set to 20. The parameters g_i, $1 \leq i \leq m$, of QEPSo, QEPSm, and QEPSn are set to be uniformly random integers ranging from 1 to 10. The parameters m and n_i, $1 \leq i \leq m$, are 20 and 1, respectively. The parameter setting for QEPS will be discussed in detail in the next section. To guarantee the seven algorithms have identical stopping criteria, the executions of QEPSo, QEPSm, and QEPSn are stopped when the best profit cannot be further improved in successive 20 iterations, and the executions of bQIEAo, bQIEAm, bQIEAcms and bQIEAn are stopped when the best profit cannot further increase in successive 100 iterations. The performances of the seven algorithms are evaluated by using the criteria: the best solution and the worst solution over 30 runs, the mean best solution over 30 runs, the standard deviation and the elapsed time. Experimental results for the three cases of 200, 400, and 600 items are shown in Table 4. All the experiments are performed on a MATLAB platform and on one machine. If the

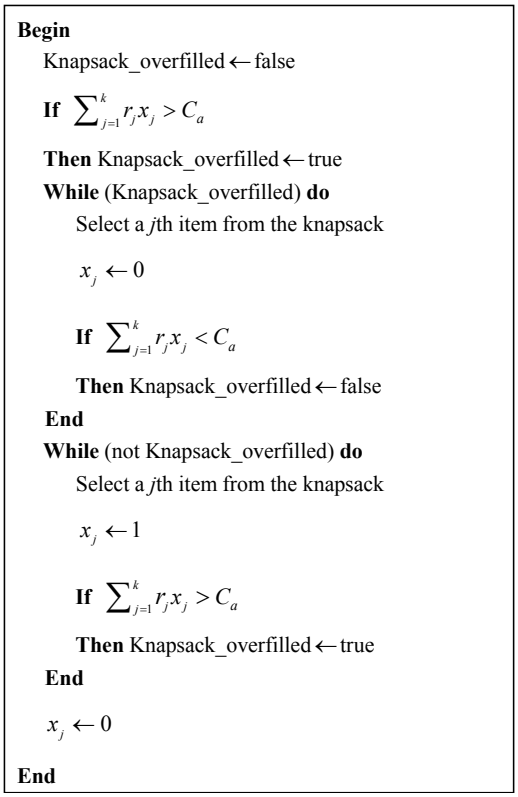


Fig. 4. Pseudocode algorithm for repair process [4]

experiments are conducted in a parallel-distributed way on several machines, the elapsed time can be greatly reduced.

As shown in Table 4, the three types of QEPS perform significantly better than the four types of QIEA in terms of profit results. QEPSm achieves the higher profit values than any other algorithm. Also, QEPSm, QEPSo and QEPSn outperform better than bQIEAm, bQIEAo and bQIEAn, respectively, with respect to profits. QEPSm and QEPSo are superior to bQIEAm and bQIEAo, respectively, with respect to the elapsed time.

The bQIEAm algorithm is the best out of the four types of QIEA and QEPSm is the best out of the seven algorithms. The profits increase at about 1.8% for 200 items, 2.6% for 400 items and 3.0% for 600 items with respect to MBS, which indicates that the increment is bigger as the number of items goes up.

Table 4. Experimental results of the knapsack problem: the number of items 200, 400 and 600, the number of runs 30. BS, MBS, WS, STD and ET represent best solution, mean best solution, worst solution, standard deviation and elapsed time (in seconds), respectively. IT and CRI are abbreviations for items and criteria, respectively

IT	CRI	QEPSm	QEPSo	QEPSn	bQIEAm	bQIEAo	bQIEAn	bQIEAcms
200	BS	**1188.31**	1089.90	1099.96	1178.33	1078.01	1088.27	1078.14
	MBS	**1179.65**	1056.24	1080.21	1159.27	1050.47	1064.90	1056.69
	WS	**1168.33**	1041.38	1057.85	1138.16	1032.56	1046.28	1032.90
	STD	**5.07**	10.82	9.81	9.26	10.91	11.56	12.43
	ET	2093.22	**847.64**	1076.95	2468.33	872.75	936.45	1014.00
400	BS	**2406.43**	2168.68	2215.23	2371.42	2150.47	2162.89	2170.44
	MBS	**2380.60**	2133.95	2177.03	2319.48	2130.82	2135.95	2132.92
	WS	**2361.43**	2101.38	2145.47	2281.34	2109.57	2110.97	2110.63
	STD	**8.91**	14.76	15.54	21.13	12.19	12.84	14.70
	ET	6988.12	**1495.03**	2129.05	7106.36	1574.77	1828.38	1757.16
600	BS	**3557.69**	3183.18	3262.69	3492.68	3172.15	3175.50	3177.64
	MBS	**3524.35**	3145.81	3202.06	3421.55	3143.61	3143.98	3177.64
	WS	**3492.68**	3116.26	3151.57	3362.53	3119.98	3115.38	3115.22
	STD	**14.81**	16.82	21.77	39.44	15.46	14.91	13.83
	ET	13231.31	**2216.11**	3557.66	13597.50	2525.98	2807.94	2372.56

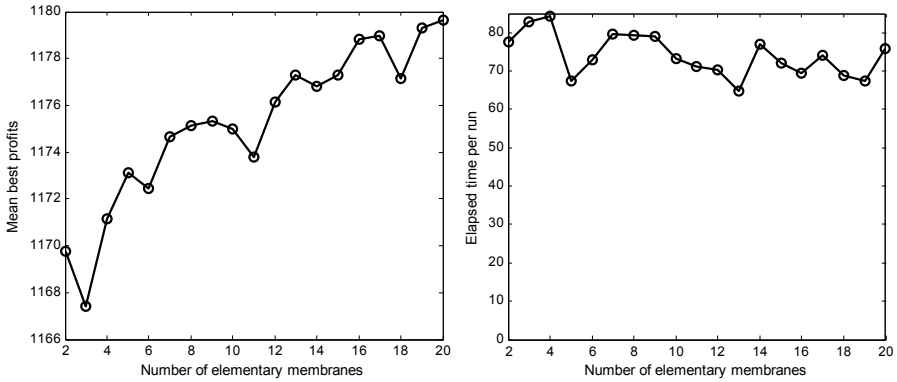
4 Discussion and Analysis

In this section we will discuss different values regarding the number m of elementary membranes, the number n_i, $1 \leq i \leq m$, of objects inside the ith elementary membrane, and the evolutionary generation g_i, $1 \leq i \leq m$, for the ith elementary membrane; finally different membrane structures will be considered and analyzed.

4.1 Parameters m and n_i

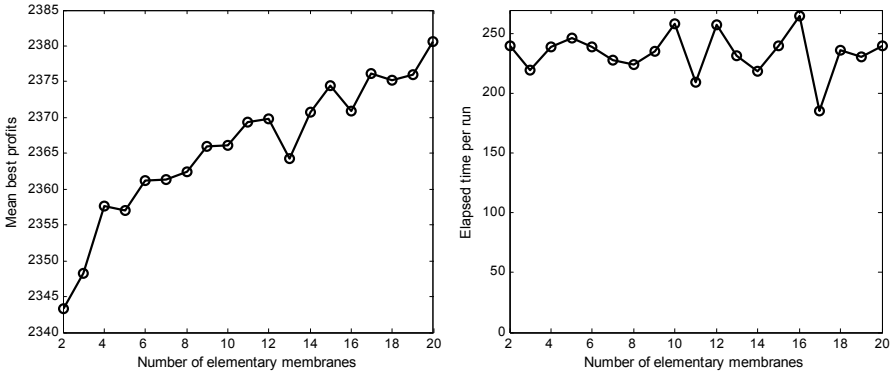
To investigate the effects of the parameters m and n_i, $1 \leq i \leq m$, on the performances of QEPS, experiments of QEPSm on the knapsack problems with 200, 400 and 600 items are tried. The population size is set to 20. For all experiments, when the best profit cannot be further improved in successive 20 iterations, the execution of the algorithm is stopped. The parameter m varies from 2 to 20. The parameter n_i, $1 \leq i \leq m$, is set to a uniformly random integers ranged from 1 to 20 on condition that the sum of $n_1, n_2, \cdots, n_m$ is 20. Also, the parameter g_i, $1 \leq i \leq m$, is set to a uniformly random integer ranged from 1 to 10. The mean best profits over 30 runs and the elapsed time per run for the three cases of 200, 400, and 600 items are shown in Fig. 5, Fig. 6 and

Fig. 7, respectively. From these experimental results, the parameters m could be assigned as 20, i.e., the number of elementary membranes is identical with the number of individuals in the population, which also means that $n_i = 1$, where $i = 1, 2, \cdots, m$. Fig. 5, Fig. 6 and Fig. 7 also illustrate that the elapsed time stays a steady level when the number of membranes increases from 2 to 20.



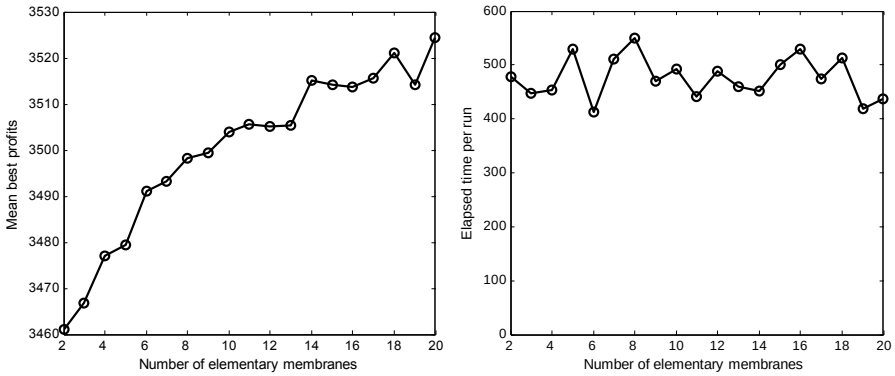
(a) Mean best profits (b) The elapsed time per run (Sec.)

Fig. 5. Experimental results of 200 items with different membranes



(a) Mean best profits (b) The elapsed time per run (Sec.)

Fig. 6. Experimental results of 400 items with different membranes



(a) Mean best profits (b) The elapsed time per run (Sec.)

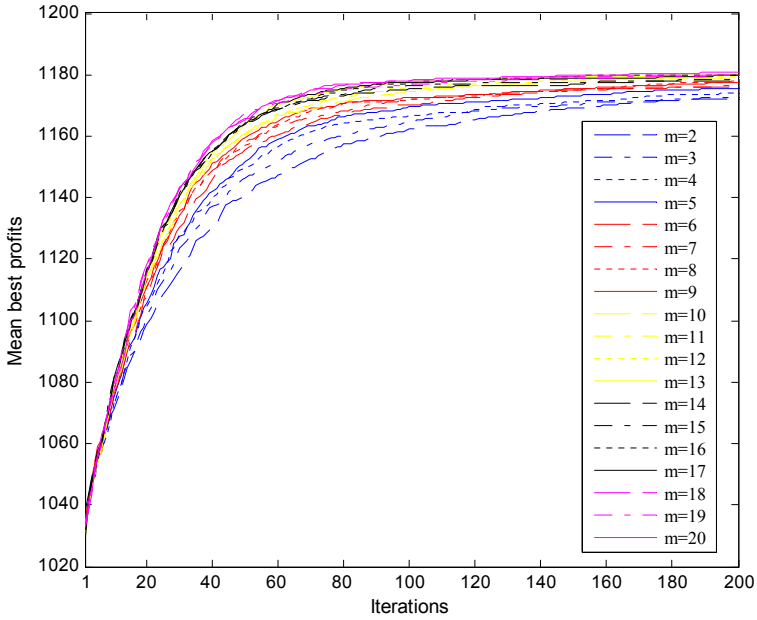
Fig. 7. Experimental results of 600 items with different membranes

Experiments are carried out with the knapsack problems for 200, 400 and 600 items to track the progress of the mean of best profits and the mean of average profits of all individuals. The population size is set to 20. The parameter n_i, $1 \leq i \leq m$, is assigned the value 1. The parameter g_i, $1 \leq i \leq m$, is set to a uniformly random integer ranged from 1 to 10. The execution of every algorithm is stopped when the best profit cannot be further improved in successive 20 iterations. The number m of elementary membranes varies from 2 to 20. Fig. 8 shows the progress of the mean of best profits and the mean of average profits of the population over 30 runs for 200, 400 and 600 items.

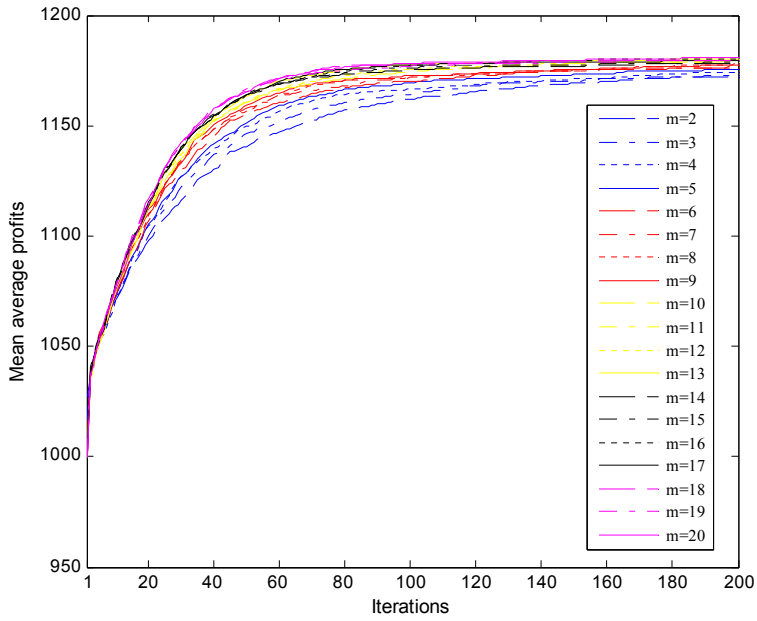
The experimental results in Fig. 8 show that the mean of best profits and the mean of average profits have steady increases with the number of membranes going up. These results indicate that QEPS has better balance between exploration and exploitation as the number of membranes rises from 2 to 20. The more the membranes are, the more directions toward the optimal solution can be explored by the QEPS. Also, the results of the mean of average profits of population show clearly the tendency of convergent rate.

4.2 Parameter g_i

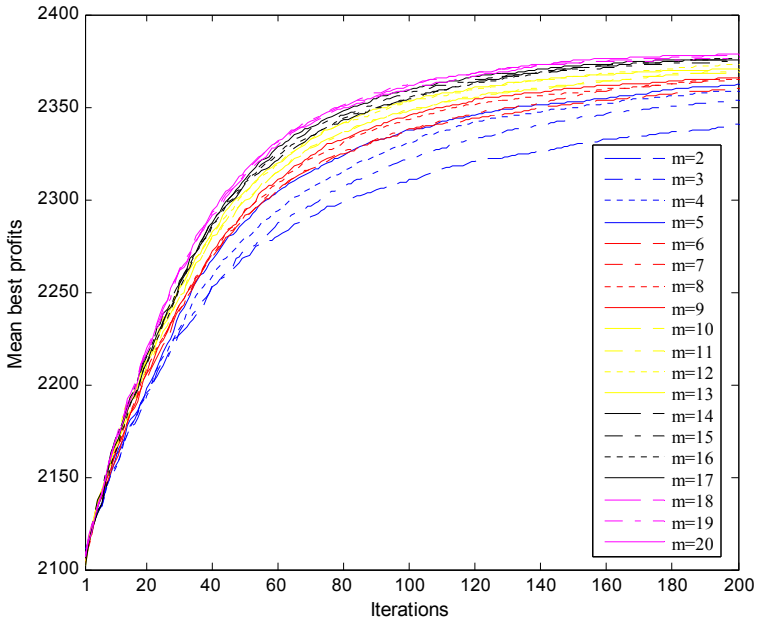
In this subsection, experiments of QEPSm for the knapsack problem with 200, 400 and 600 items are carried out to investigate the effect of the number of evolutionary generations parameter, g_i, $1 \leq i \leq m$, on the performances of this algorithm. Both the population size and the parameter m are set to 20. The parameter n_i, $1 \leq i \leq m$, is then becoming 1. For all experiments, when the best profit cannot be further improved in successive 20 iterations,



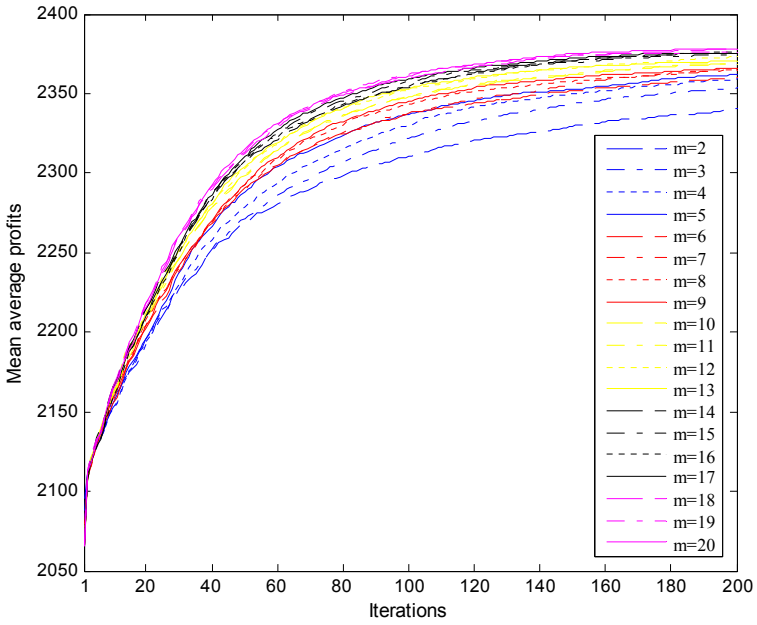
(a) Mean best profits of 200 items



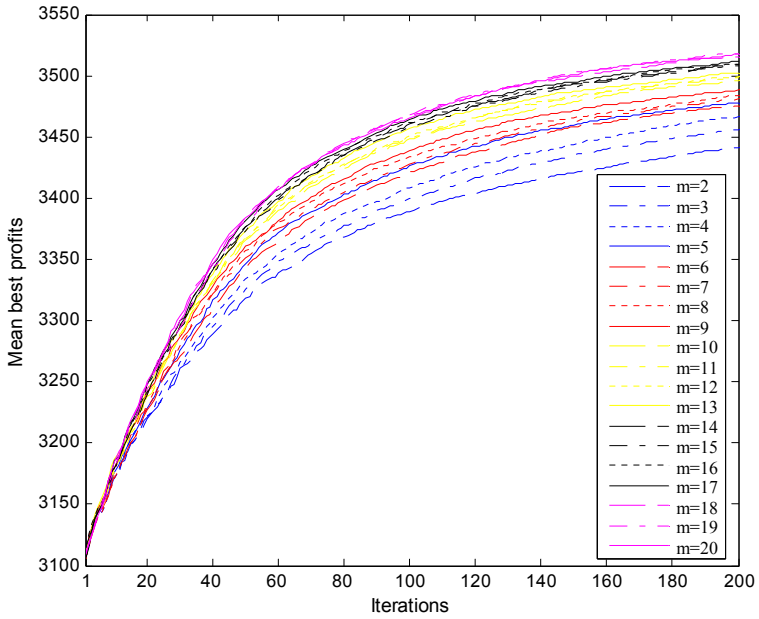
(b) Mean average profits of 200 items



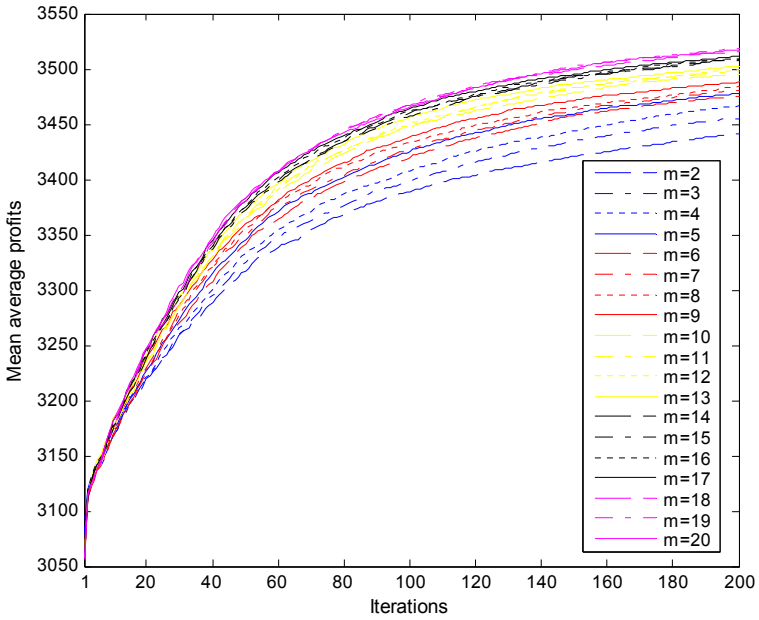
(c) Mean best profits of 400 items



(d) Mean average profits of 400 items



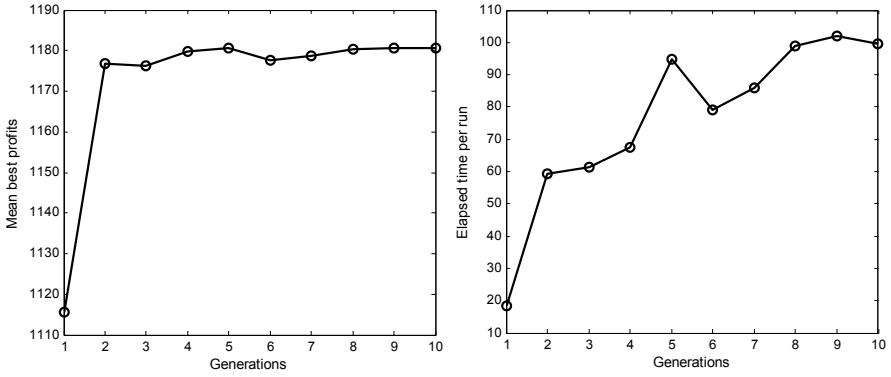
(e) Mean best profits of 600 items



(f) Mean average profits of 600 items

Fig. 8. Progress of solutions with different membranes

the execution of the algorithm is stopped. The parameter g_i, $1 \leq i \leq m$, varies between 1 and 10. In each of the above mentioned experiments, the mean best profits over 30 runs and the elapsed time per run are shown in Fig. 9, Fig. 10 and Fig. 11. The mean best profit values for $m = 20$ in Fig. 5, Fig. 6 and Fig. 7 are very close to the values shown in Fig. 9, Fig. 10 and Fig. 11, for the parameter g_i, $1 \leq i \leq m$, arbitrarily chosen between 2 and 10. These experiments indicate that the values associated to the parameter g_i, $1 \leq i \leq m$, do not influence the mean best profit when they are within the range 2 to 10.



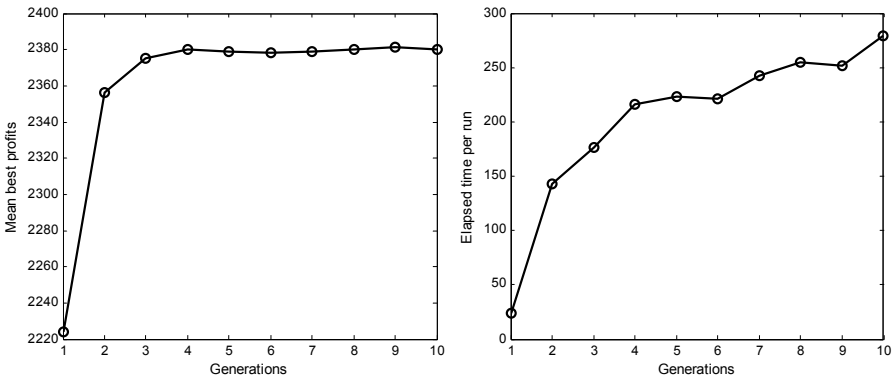
(a) Mean best profits (b) The elapsed time per run (Sec.)

Fig. 9. Mean best profits and elapsed time of 200 items

4.3 Membrane Structures

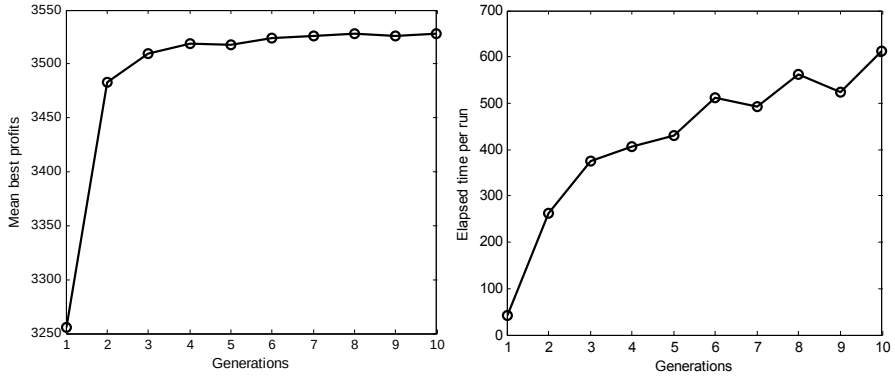
In the above experiments, the membrane structure consisted of a skin membrane and m elementary membranes inside. This membrane structure, called one level membrane structure (OLMS), is illustrated in Fig. 12, and considered in the context of QEPS. In the sequel another membrane structure will be discussed, a nested membrane structure (NMS) shown in Fig. 13. Experiments will be conducted with respect to the knapsack problem to assess the use of NMS. This membrane structure, also called linear topology in [20], was used in [9, 10, 11, 7] in combination with various evolutionary approaches.

In the case of the nested membrane structure we will run experiments under the same conditions we have considered for OLMS. Consequently, the number of individuals contained by each region is arbitrarily chosen between 1 and 20 under the condition that the overall sum equals the population size, which is 20 and experiments are carried out with the knapsack problem for



(a) Mean best profits (b) The elapsed time per run (Sec.)

Fig. 10. Mean best profits and elapsed time of 400 items



(a) Mean best profits (b) The elapsed time per run (Sec.)

Fig. 11. Mean best profits and elapsed time of 600 items

200, 400 and 600 items. All the other parameters and the stopping criterion are the same as those considered for OLMS. An important difference between the two approaches is given by the way the communication rules defined by the P system-like framework are applied. For the OLMS case we remember that the best fit individual binary representation from each of the m regions is sent into the skin membrane and then the overall best fit element is then sent back in each compartment. In the NMS case, the better fit individual will be selected between adjacent neighbours in compartments 2 to $m+1$ and the skin

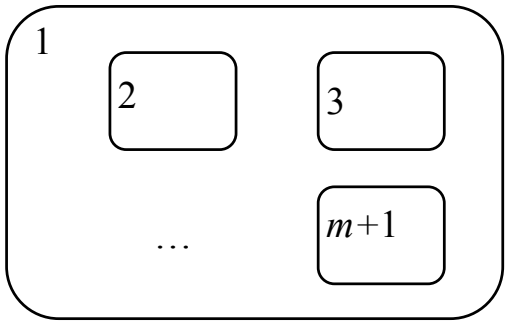


Fig. 12. A one level membrane structure

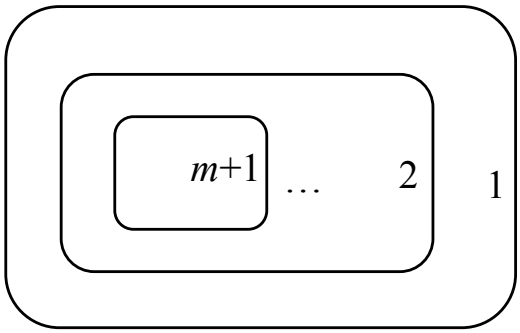


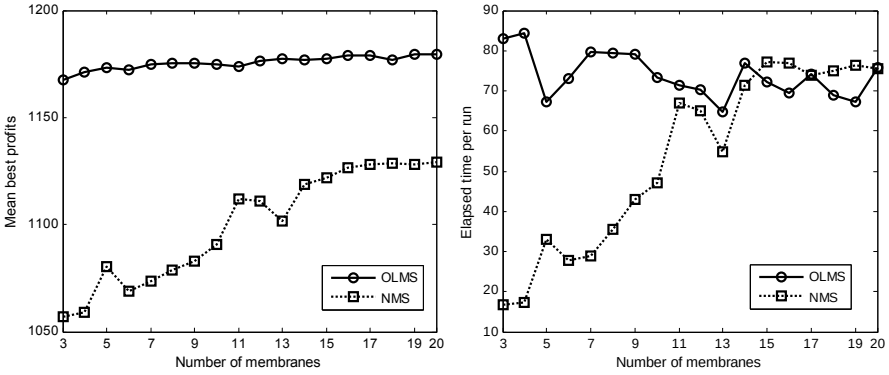
Fig. 13. A nested membrane structure

membrane, denoted by 1, does not play any role in this case. Subsequently, the better fit individual between any two adjacent compartments, i and $i+1$, will be pushed back into the lower compartment, i.e., $i+1$. Through this process the best fit individual will be popped up into the top compartment, i.e., 2. Fig. 14, Fig. 15 and Fig. 16 show the comparative results of using these two membrane structures. All the experimental results are averaged over 30 runs. Table 5 shows the best and worst solutions as well as the mean best solution over 30 runs; the standard deviations and the elapsed time for each of two membrane structures, when the number of membranes varies between 3 and 20 are also shown. Obviously, NMS and OLMS with two membranes show the same behaviour.

The experimental results shown in Fig. 14, Fig. 15 and Fig. 16, prove that, irrespective of the number of membranes used, the profit values obtained in the OLMS case are consistently better than those using NMS, but, on the other hand, the OLMS case requires more computing time than NMS. It is also worth pointing out that, when the number of membranes is above 15, the elapsed time for the QEPS algorithm using either OLMS or NMS is

approximately the same. These results indicate that QEPS with OLMS has better search capabilities than QEPS with NMS.

Table 4 and Table 5 show that QEPS with OLMS is better than QEPS with NMS with respect to the best and worst solutions, the mean best solution, the standard deviations and the elapsed time. These results show that, in the case of the knapsack problem, using the current best solution to control the production of the next generation of individuals (OLMS case) works better than using the best solution between two neighbouring regions (NMS case). Both these approaches produce, in general, better results than most of the bQIEA strategies. More precisely, QEPS with NMS performs better than QEPSo, QEPSn, bQIEAo, bQIEAn and bQIEAcms, but bQIEAm is between QEPSm with OLMS and QEPSm with NMS, in terms of profits. These results show that the choice of the membrane structure for a QEPS algorithm matters and the results might go either way with respect to bQIEA.

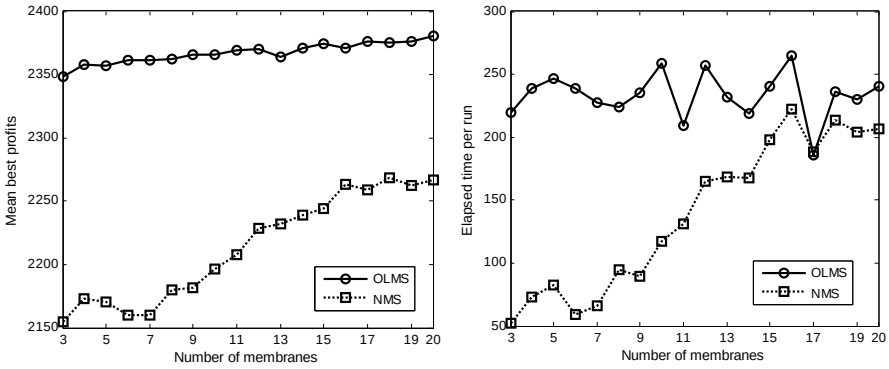


(a) Mean best profits (b) The elapsed time per run (Sec.)

Fig. 14. Comparisons of two structures with 200 items

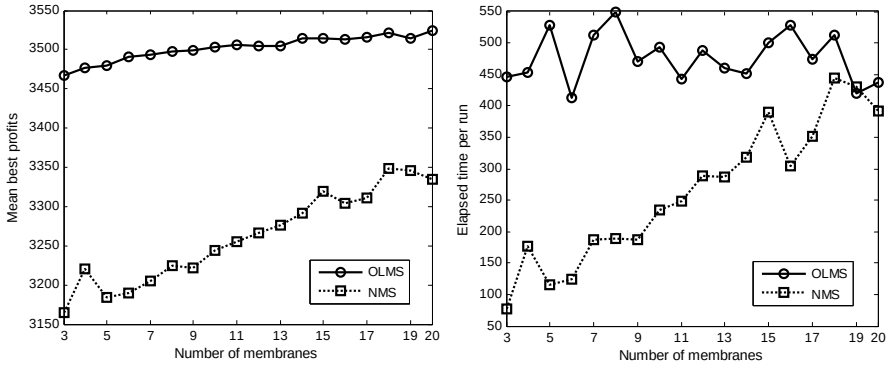
5 Conclusions

This paper proposes the use of quantum-inspired evolutionary algorithms within the parallel-distributed framework of the membrane computing. The algorithms defined in this respect are characterized by a certain membrane structure, string-like objects encoding for Q-bit individuals, and evolution rules usually defined for QIEA approaches. The knapsack problem is considered as an application example to investigate the performances of these



(a) Mean best profits (b) The elapsed time per run (Sec.)

Fig. 15. Comparisons of two structures with 400 items



(a) Mean best profits (b) The elapsed time per run (Sec.)

Fig. 16. Comparisons of two structures with 600 items

evolutionary algorithms. Experimental results show that QEPS algorithms perform in general better than their QIEA counterparts and they can be used to produce effective and efficient solutions to hard combinatorial optimization problems.

Table 5. Experimental results of two structures: the number of items 200, 400 and 600, the number of runs 30. BS, MBS, WS, STD and ET represent best solution, mean best solution, worst solution, standard deviation and elapsed time (in seconds), respectively

Items	Criterions	OLMS	NMS
200	BS	**1188.31**	1153.25
	MBS	**1179.65**	1129.18
	WS	**1168.33**	1052.51
	STD	**5.07**	19.53
	ET	**2093.22**	2261.94
400	BS	**2406.43**	2296.32
	MBS	**2380.60**	2268.46
	WS	**2361.43**	2236.28
	STD	**8.91**	16.66
	ET	6988.12	**6420.31**
600	BS	**3557.69**	3392.53
	MBS	**3524.35**	3348.84
	WS	**3492.68**	3142.04
	STD	**14.81**	45.52
	ET	**13231.31**	13349.63

Acknowledgements

The first author is supported by the National Natural Science Foundation of China (60702026, 60572143). The third author acknowledges the National Basic Research Program of China (2005CB724205, 2006CB705505).

References

1. T. Bäck, U. Hammel, H. P. Schwefel, Evolutionary computation: comments on the history and current state, *IEEE Transactions on Evolutionary Computation*, **1** (1997) 3–17.
2. G. Ciobanu. Distributed Algorithms over Communicating Membrane Systems, *BioSystems*, **70** (2003) 123–133.
3. K. H. Han, J. H. Kim, Genetic quantum algorithm and its application to combinatorial optimization problem, in: *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*, 2000, 1354–1360.
4. K. H. Han, J. H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, *IEEE Transactions on Evolutionary Computation*, **6** (2002) 580–593.
5. L. Huang and N. Wang, An optimization algorithm inspired by membrane computing, L. Jiao et al. (Eds.): *ICNC2006*, Part II, Lecture Notes in Computer Science, **4222** (2006) 49–52.

6. L. Huang, X. X. He, N. Wang and Y. Xie, P systems based multi-objective optimization algorithm, *Progress in Natural Science*, **17** (2007) 458–465.
7. A. Leporati and D. Pagani, A membrane algorithm for the min storage problem, H. J. Hoogeboom et al. (Eds.): *WMC2006*, Lecture Notes in Computer Science, **4361** (2006) 443–462.
8. A. Narayanan, M. Moore, Quantum-inspired genetic algorithm, in: *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, 61–66.
9. T. Y. Nishida, An approximate algorithm for NP-complete optimization problems exploiting P systems, in: *Proceedings of Brainstorming Workshop on Uncertainty in Membrane Computing*, Palma de Majorca, 2004, 185–192.
10. T. Y. Nishida, Membrane algorithms: an approximate algorithm for NP-complete optimization problems exploiting P systems, in: *Application of Membrane Computing* (G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, eds.), Springer, Berlin, 2005, 303–314.
11. T. Y. Nishida, Membrane Algorithms, R. Freund et al. (Eds.): *WMC2005*, Lecture Notes in Computer Science, **3850** (2006) 55–66.
12. L. Q. Pan, C. Martin-Vide, Solving multidimensional 0-1 knapsack problem by P systems with input and active membranes, *Journal of Parallel Distributed Computing*, **65**, (2005) 1578–1584.
13. Gh. Păun, Computing with Membranes, *Journal of Computer and System Sciences*, **61** (2000) 108–143.
14. Gh. Păun, P systems with active membranes: attacking NP-complete problems, *Journal of Automata Language Combinatorics*, **6** (2001) 75–90.
15. Gh. Păun and G. Rozenberg, A guide to membrane computing, *Theoretical Computer Science*, **287** (2002) 73–100.
16. Gh. Păun, Further twenty-six open problems in membrane computing, in: *Proceedings of 3rd Brainstorming Meeting on Membrane Computing*, Sevilla, Spain, 2005, 249–262.
17. M. J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, Computationally hard problems addressed through P systems, G. Ciobanu, G. Păun, M. J. Pérez-Jiménez (Eds.), *Applications of Membrane Computing*, Springer, 2006, 315–346.
18. M. Srinivas, L .M. Patnaik, Genetic algorithms: a survey, *Computer*, **27** (1994) 17–26.
19. S. Y. Yang, M. Wang, L. C. Jiao, A novel quantum evolutionary algorithm and its application, in: *Proc. of the 2004 Congress on Evolutionary Computation*, 2004, 820–826.
20. D. Zaharie and Gabriel Ciobanu, Distributed evolutionary algorithms inspired by membranes in solving continuous optimization problems, H. J. Hoogeboom et al. (Eds.): *WMC2006*, Lecture Notes in Computer Science, **4361** (2006) 536–553.
21. G. X. Zhang, L. Z. Hu, W. D. Jin, Resemblance coefficient and a quantum genetic algorithm for feature selection. E. Suzuki, S. Arikawa (Eds.): *DS2004*, Lecture Notes in Artificial Intelligence, **3245** (2004) 155–168.
22. G. X. Zhang, H. N. Rong, Real-observation quantum-inspired evolutionary algorithm for a class of numerical optimization problems. Y. Shi et al. (Eds.): *ICCS2007*, Part IV, Lecture Notes in Computer Science, Springer, **4490** (2007) 989–996.