Journal of Universal Computer Science, vol. 23, no. 7 (2017), 652-672
submitted: 31/3/17, accepted: 30/6/17, appeared: 28/7/17 © J.UCS

An Adaptive Membrane Evolutionary Algorithm for Solving Constrained Engineering Optimization Problems

Jianhua Xiao
(The Research Center of Logistics, Nankai University, Tianjin 300071, China
jhxiao@nankai.edu.cn)

Ying Liu
(The Research Center of Logistics, Nankai University, Tianjin 300071, China
18662150807@163.com)

Shuai Zhang
(Asper School of Business, University of Manitoba, Winnipeg, Manitoba, R3T5V4, Canada
Zhangs15@myumanitoba.ca)

Ping Chen*
(Business School, Nankai University, Tianjin 300071, China
chenpingteda@nankai.edu.cn)

Abstract: In this paper, an adaptive membrane evolutionary algorithm (AMEA) is proposed, which combines a dynamic membrane structure and a differential evolution with the adaptive mutation factor. In the AMEA, the feasibility proportion method is used to dynamically adjust the size of the elementary membrane in the optimization process. The results of the experimental indicate that the proposed algorithm outperforms other evolutionary algorithms on five well-known constrained engineering optimization problems.

Keywords: Membrane Computing, Membrane Algorithm, Differential Evolution, Engineering Optimization Problem
Category: I.2.8

1 Introduction

In real engineering applications, such as automobile cab layout, engineering design, and structural optimization, many optimization problems need to ensure that the feasible solutions satisfy many constraints. Thus, constrained optimization problems (COPs) have come to the foreground in recent years. Generally, COPs can be formulated as follows:

Minimize

$$f(X)$$

Subject to

* Corresponding author.

$$\begin{aligned} & g_i(X) \le 0,\ i = 1, 2, \ldots, l \\ & h_j(X) = 0,\ j = 1, 2, \ldots, m \\ & L \le X \le U \end{aligned} \tag{1}$$

where $X = (x_1, x_2, ..., x_n) \in E^n$ denotes an *n*-dimensional vector that satisfies $l+m$ constraints and minimizes the objective function *f(X)*; $g_i(X) \le 0$ *and* $h_j(X) = 0$ are the *i*th inequality constraint and the *j*th equality constraint, respectively; $L = (l_1, l_2, \ldots, l_n)$ is the lower bound vector, and $U = (u_1, u_2, ..., u_n)$ is the upper bound vector.

A number of engineering design optimization problems can be formulated as COPs. However, solving such problems is computationally complex and traditional optimization techniques are currently inadequate. In this study, an adaptive membrane evolutionary algorithm combining dynamic membrane structure and differential evolution with an adaptive mutation factor is developed to solve constrained engineering optimization problems (CEOPs). Experimental results on five well-known CEOPs indicate that the proposed algorithm outperforms other evolutionary algorithms.

The remainder of this paper is organized as follows. In section 2, a brief review of the literature is presented. Section 3 proposes an adaptive membrane evolutionary algorithm for CEOPs. In section 4, the simulation results and analyses are reported. Section 5 concludes the paper and describes future work.

2 Related work

In the past decade, various optimization algorithms have been proposed to solve constrained engineering design problems. [Lee et al., 05] proposed a new meta-heuristic algorithm based on the harmony search to solve CEOPs. [Wang et al., 08] developed a new ranking selection-based particle swarm algorithm to solve optimization problems in engineering design. [Wang, 10] applied level comparison to a simple differential evolution algorithm to solve CEOPs. [Zhao, 12] developed a hybrid genetic algorithm and applied the flexible allowance technique to this algorithm with the aim of optimizing constrained engineering design problems. [Kashan, 11] adapted a league championship algorithm (LCA) for solving mechanical engineering design problem. [Melo, 13] put forward a multi-view differential evolution algorithm that applied mutation strategies to solve constrained engineering design problems. [Bulatovic, 14] proposed an improved cuckoo search (ICS) algorithm to solve CEOPs, and this algorithm gave better solutions than the standard CS algorithm. [Xiao et al., 16] developed an improved membrane algorithm based on the P system, particle swarm optimization and differential evolution (PSO/DE) for solving CEOPs. [Askarzadeh et al., 16] put forward a novel crow search algorithm to solve CEOPs. [Kohlia et al., 17] applied chaos theory to the gray wolf optimizer algorithm with the aim of optimizing the constrained engineering design.

Natural computing is a computational process inspired by nature, and has been intensively studied in recent years. Types of natural computing include membrane

computing [Zhang, 15(a); Song, 14; Pan, 16; Song, 17], DNA computation [Xu, 16], evolutionary algorithms [Zhang, 15(b, c)-16(a, b)], and neural computing [Zeng, 14(a, b, c); Song, 15]. As a novel paradigm of natural computing, membrane computing has been developed rapidly at the theoretical level and has many applications to various complex problems [Pan, 10; Wang, 10; OnoltIshdorj, 10; Pan, 11; Niu, 11]. Inspired by membrane computing, [Nishida, 04(a, b)] made the first attempt in this direction and developed a nest-like membrane algorithms to solve TSP. After Nishida, [Huang, et al., 06-07] attempted to apply the membrane algorithm to solve objective optimization problems. [Zhao, 11] proposed a membrane evolutionary algorithm (MEA) with the aim of solving gasoline blending and scheduling problems. [Idowu et al., 13] attempted to build a new robust membrane algorithm to optimize anomaly-based intrusion detection system. [Wang et al., 13] developed a novel membrane algorithm based on the tissue-like membrane system with a ring-shaped topology structure to optimize infinite impulse response design. [Zhang et al., 13] proposed a hybrid method based on tissue membrane systems and differential evolution to optimize the parameters in constrained manufacturing problems. [Yan et al., 14] developed a novel approach based on membrane computing to optimize the attribute weights. [Peng, 15] proposed a membrane clustering algorithm using hybrid evolutionary mechanisms to address data clustering problem. [Zhang(C) et al., 16] proposed a novel multi-objective membrane algorithm guided by the skin membrane. [Deng et al., 16] developed a novel approach based on the membrane computing and pigeon-inspired optimization algorithm to solve the parameter design problem of brushless direct current moto. [Singh and Deep, 16] presented a new membrane algorithm based on cell-like P-systems and PSO to solve Sudoku.

In the common MEA, the size of the elementary membrane is usually a constant. However, there is no theoretical basis. In recent years, the random method is used to dynamically adjust the size of the elementary membrane in the optimization process, but there is a lack of the information of the current populations to guide the next search. In this paper, the feasibility proportion method is used to adaptively adjust the size of the elementary membrane in the optimization process, and an adaptive membrane evolutionary algorithm is proposed to solve CEOPs. This algorithm combines the dynamic structure of membrane computing and the differential evolution (DE) with the adaptive mutation factor.

3 Adaptive membrane evolutionary algorithm for CEOPs

3.1 P systems

This section provides a brief background on cell-like P systems. Generally, the essential constituent elements of P systems include membrane structure, rules and objects. The membrane structure is a hierarchical arrangement of membranes. Inside of this structure is a main membrane, called the skin, which contains several membranes to delimit compartments where multisets of objects are located. The specific membrane structure is shown in Figure 1. If there is not any other membrane inside a membrane, the membrane is defined as an elementary membrane. The space between each adjacent pair of membranes is called a region. There are a series of

rules regarding the communication and evolution of the objects within the regions and compartments of a membrane. Generally, a P system is defined as follows [Xiao, 16].

$$\Pi = (O, \mu, L_1, ..., L_n, R_1, ..., R_n) \tag{2}$$

where

1) $n \geq 1$ is the degree of P system,
2) O is the alphabet of objects,
3) μ represents a membrane structure,
4) $L_i (1 \leq i \leq n)$ are sets of strings over O, and
5) $R_i (1 \leq i \leq n)$ are the developmental rule sets of membrane computing. There are five main rules as follows:

(a) $[_i s_1 \rightarrow s_2]_i$, $i \in \{1, 2, ..., n\}$, $s_1, s_2 \in O^*$
(Evolution rules: The object is modified by using various evolution operators.)

(b) $s_1[_i\ ]_i \rightarrow [_i s_2]_i$, $i \in \{1, 2, ..., n\}$, $s_1, s_2 \in O^*$
(In-communication rules: an object is sent into the membrane.)

(c) $[_i s_1]_i \rightarrow [_i]_i s_2$, $i \in \{1, 2, ..., n\}$, $s_1, s_2 \in O^*$
(Out-communication rules: an object is sent out from the membrane)

(d) $[_i s_1]_i [_i s_2]_i \rightarrow [_i s_1, s_2]_i$, $i \in \{1, 2, ..., n\}$, $s_1, s_2 \in O^*$
(Merging rules: two membranes are merged into a single membrane.)

(e) $[_i W]_i \rightarrow [_i U]_i [_i W - U]_i$, $\quad W, U \in O$
(Separation rules: An elementary membrane is separated into two membranes.)

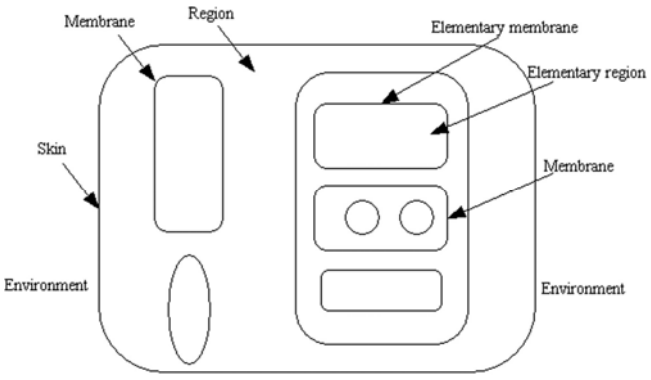

Figure 1: The membrane structure

In P systems, different computational processes correspond to different evolutionary rules. However, the implementation of these rules follows the principles of non-determinacy (system random selection rules) and maximum parallelism. In the

initial configuration, the rules are implemented in a maximally parallel and non-deterministic manner. The configuration of a P system is updated once every time the rules are implemented. When no rules can be used in any region, the system will halt and the configuration will be produced. Then, the output result or a multiset of objects can be obtained in region i_0.

3.2 Differential evolution

Differential evolution (DE) was first mentioned by [Storn, 97]. DE is a population-based evolutionary algorithm that is used to solve multidimensional optimization problems involving continuous variables. DE has been extended to address integer, discrete and mixed-integer problems. Compared with other evolutionary algorithms, DE has a simple structure, is easy to use, is quick to compute, and is robust. The main idea of DE is to obtain a difference vector by subtracting two different individuals, which are randomly selected from the population. Next, add the difference vector to a third vector to obtain the mutated vector. Then, compare the mutated individual with the third individual. If the mutated individual is better, it is accepted into the next generation. The pseudocode of DE is shown in Algorithm 1.

Like other heuristic algorithms, DE is sensitive to its tuning parameters. The proper parameters are essential to the performance of the algorithm. In DE, the mutation factor F controls the scaling ratio of the differential vector, and can improve the local exploitation and global exploration of DE. In the classical DE, F is a constant and cannot reflect the actual search process. In the paper, the self-adaptive parameter-setting technique of the mutation factor is defined as follows [Wang, 15].

$$F = F_{\min} + (F_{\max} - F_{\min}) \times e^{1-\frac{Iter_{\max}}{Iter_{\max}-t+1}} \quad (3)$$

where $Iter_{\max}$ and t are the maximum and current iteration number, respectively. $F_{\min}$ and $F_{\max}$ denote the minimum and maximum value of the mutation factor, respectively.

Algorithm 1: DE with the "DE/*rand*/1" strategy

1 // Initialization:
2 G=0;
3 Generate the initial population $P_G = \{X_{1,G}, X_{2,G}, ..., X_{NP,G}\}$ randomly;
4 Calculate the initial fitness of P_G;
5 // Loop:
6 **While** the stopping criteria are not met **DO**
7 **For** *i*=1 **to** *NP* **DO**
8 Generate three different integers $r_1, r_2, r_3 \in \{1,2,...,i-1,i+1,...,NP\}$;
9 Generate the mutant vector $V_{i,G}$ using $V_{i,G} = X_{r_1,G} + F \cdot (X_{r_2,G} - X_{r_3,G})$;
10 Generate $j_{rand} = rand\,\mathrm{int}[1, D]$;
11 **For** *j*=1 **to** *D* **DO**
12 **If** $rand_j(0,1) \le CR$ **or** $j = j_{rand}$, **then**
13 $u_{ij,G} = v_{ij,G}$
14 **Else**
15 $u_{ij,G} = x_{ij,G}$
16 **End If**
17 **End for**
18 Calculate the fitness of $U_{i,G}$
19 **If** $f(U_{i,G}) \le f(X_{i,G})$ **then**
20 $X_{i,G+1} = U_{i,G}$
21 **Else**
22 $X_{i,G+1} = X_{i,G}$
23 **End If**
24 **End For**
25 G=G+1
26 **End While**

3.3 Adaptive membrane evolutionary algorithm

This section will introduce our proposed algorithm, AMEA, which combines P systems and adaptive differential evolution. The AMEA uses a dynamic P systems-like framework by implementing the merging rule and separation rule of membrane computing. In each iteration, the size of the elementary membrane can adjust adaptively by using the feasibility proportion of the current population. The algorithmic flowchart of the AMEA is demonstrated in Figure 2.

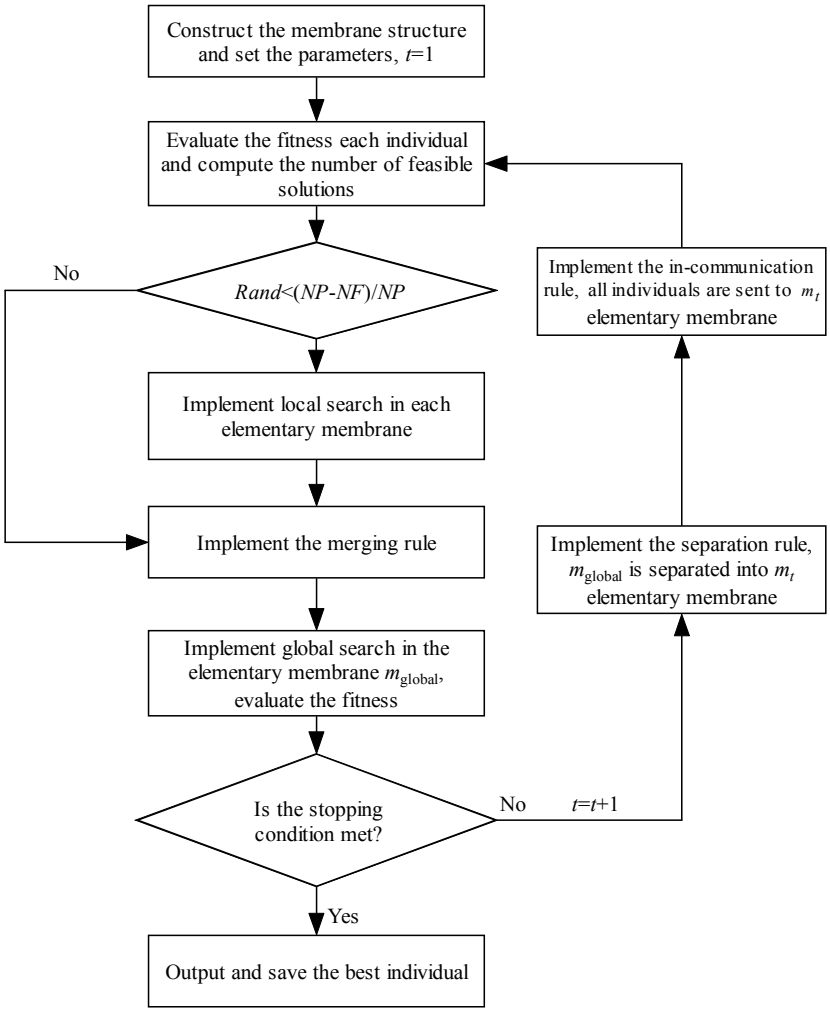

Figure 2: The flowchart of the AMEA

Next, we describe the steps of the adaptive membrane evolutionary algorithm in details.

Step 1: Initialize the parameters of the AMEA, including the population size *NP*, the crossover rate *CR*, the scaling factor *R*, the max number of iteration $Iter_{\max}$, and the dynamic membrane structure $[_0 [_1]_1 [_2]_2 \cdots [_{m_t}]_{m_t}]_0$ with m_t elementary membranes and a skin membrane 0 at iteration t=1, where m is an unfixed number in the optimization process.

Step 2: In each elementary membrane, generate $\lfloor NP / m_t \rfloor$ individuals by randomly and uniformly sampling search space, evaluate the fitness and the degree value of constraint violation for each individual, and compute the number of feasible solutions *NF* in all elementary membranes.

Step 3: If *rand*< (*NP*-*NF*)/*NP*, the evolutionary rule based on the adaptive differential evolution is simultaneously implemented in each elementary membrane, and the individuals are updated inside. Otherwise, go to Step 4.

Step 4: Implement the merging rule. m_t elementary membranes are merged into a single membrane m_{global}; the individuals of each elementary membranes are put into the elementary membrane m_{global} together.

Step 5: In m_{global}, implement the adaptive differential evolution for all individuals, evaluate the fitness and the degree value of constraint violation for each individual, and compute the number of feasible solutions *NF*.

Step 6: The out-communication rule is implemented, and a copy of the current best individual in the elementary membrane m_{global} is sent to the skin membrane.

Step 7: If the stopping criterion is met, stop and output the best individual. Otherwise, go to Step 8.

Step 8: Implement the separation rule. The elementary membrane m_{global} is separated into m_t elementary membranes. In the paper, m_t is defined as follows:

$$m_t = m_{\max} - \left\lfloor (m_{\max} - m_{\min}) \times \frac{NP - NF}{NP} \right\rfloor \qquad (4)$$

In above equation, m_t is the number of elementary membranes at iteration *t*; $m_{\max}$ and $m_{\min}$ are the maximum and minimum number of elementary membranes, respectively.

Step 9: Implement the in-communication rule. The individuals selected in elementary membrane m_{global} are sent into m_t elementary membrane. The sending process is defined as follows: (1) Randomly select an individual from the elementary membrane m_{global}; (2) Find $\lfloor NP / m_t - 1 \rfloor$ nearest individuals

of the selected individual using the minimum Euclidean distance; (3) Implement the in-communication rule and $\lfloor NP / m_t \rfloor$ individuals are sent to an elementary membrane. Repeat the above processes until all individuals are sent into each elementary membrane. Set $t = t + 1$, and return to Step 3.

4 Simulation results

To evaluate the effectiveness of the proposed algorithm, five well-known engineering optimization examples [Sadollah, 13] are considered. In the experiment, the AMEA is executed in Matlab 7.0 on a PC with Intel CPU 3.6 GHz 12GB RAM. Each example is performed 50 times, independently. The size of the population is 60; the maximum number of iteration is 5000; the crossover rate CR is 0.1. The maximum and minimum number of the elementary membrane are 10 and 5, respectively. $F_{\min}$ and $F_{\max}$ of ADE are set 0.2 and 0.9, respectively.

4.1 Welded beam design problem

In 2000, [Coello et al., 00] first proposed the welded beam design problem (WBDP). The objective is to minimize the cost of a welded beam subject to constraints, including end deflection (δ) in the beam, shear stress (τ), buckling load (P_b) of the bar, bending stress (σ) of the beam and side constraints. Four design variables of this problem are shown in Figure 3, namely, h, l, t and b.

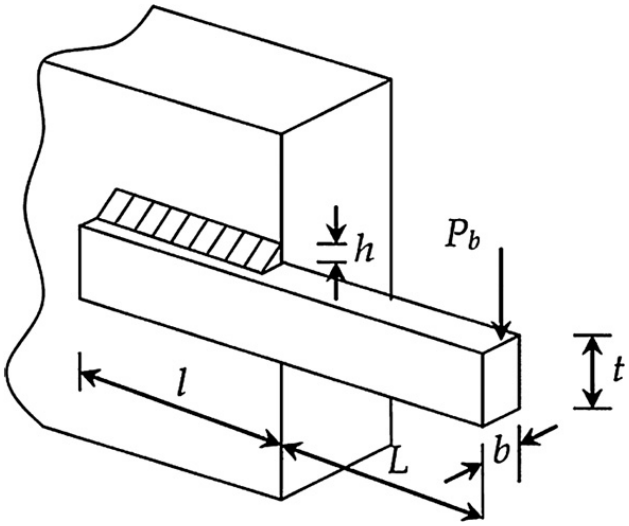

Figure 3: Welded beam design problem

The problem can be formulated as follows:
Minimize:

$$f = 1.10471 l h^2 + 0.04811 t b (14 + l) \tag{5}$$

Subject to:

$$\tau - \tau_{\max} \le 0 \tag{6}$$

$$\sigma - \sigma_{\max} \le 0 \tag{7}$$

$$h - b \le 0 \tag{8}$$

$$0.10471h^2 + 0.04811tb(14 + l) - 5 \le 0 \tag{9}$$

$$0.125 - h \le 0 \tag{10}$$

$$\delta - \delta_{\max} \le 0 \tag{11}$$

$$P - P_c \le 0 \tag{12}$$

$$0.1 \le h, b \le 2 \tag{13}$$

$$0.1 \le l, t \le 10 \tag{14}$$

where:

$$\tau = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{l}{2R} + (\tau'')^2}, \quad \tau' = \frac{P}{\sqrt{2}hl}, \quad \tau'' = \frac{MR}{J} \tag{15}$$

$$M = P(L + \frac{l}{2}) \tag{16}$$

$$R = \sqrt{\frac{l^2}{4} + (\frac{h+l}{2})^2} \tag{17}$$

$$J = 2\left\{\sqrt{2}hl\left[\frac{l^2}{12} + (\frac{h+t}{2})^2\right]\right\} \tag{18}$$

$$\sigma = \frac{6PL}{bt^2} \tag{19}$$

$$\delta = \frac{4PL^3}{Ebt^3} \tag{20}$$

$$P_c = \frac{4.013E\sqrt{(t^2b^6/36)}}{L^2} \times (1 - \frac{t}{2L}\sqrt{\frac{E}{4G}}) \tag{21}$$

$$P = 6000 \text{ lb}, \quad L = 14 \text{ in}, \quad E = 30 \times 10^6 \text{ psi}, \quad G = 12 \times 10^6 \text{ psi} \tag{22}$$

$$\tau_{\max} = 13600 \text{ psi}, \quad \sigma_{\max} = 30000 \text{ psi}, \quad \delta_{\max} = 0.25 \text{ in} \tag{23}$$

The best solution generated by the AMEA is (h, l, t, b) =(0.20572963978, 3.47048866562, 9.03662391035, 0.20572963978). The statistical simulation results are listed in Table 1. From Table 1, it can be observed that the proposed algorithm can find better solutions than those found by GA3 [Coello, 00], GA4 [Coello, 02], SC [Ray, 03], PSO-DE [Liu, 10], CPSO [He, 07(a)], HPSO [He, 07(b)], UPSO

[Parsopoulos, 2005], (μ + λ)-ES [Mezura-Montes, 05], and MBA [Sadollah, 13], except for IDMEA [Xiao, 2013] in terms of *Best*. However, the AMEA can obtain better values than those by the IDMEA in terms of *Worst*, *Mean* and *Std*.

Algorithm	Best	Mean	Worst	Std.
GA3	1.748309	1.771973	1.785835	1.12e-02
SC	2.3854347	3.0025883	6.3996785	9.60e-01
GA4	1.728226	1.792654	1.993408	7.47e-02
PSO-DE	1.724852	1.724852	1.724852	6.70e-16
CPSO	1.728024	1.748831	1.782143	1.29e-02
HPSO	1.724852	1.749040	1.814295	4.01e-02
UPSO	1.92199	2.83721	N.A.	0.683
(μ + λ)-ES	1.724852	1.777692	N.A.	8.80e-02
MBA	1.724853	1.724853	1.724853	6.94e-19
IDMEA	1.7248523	1.7248524	1.7248526	6.58e-08
AMEA	**1.7248523**	**1.7248523**	**1.7248523**	**1.0e-15**

Table 1: Comparison of results for the welded beam problem

4.2 Three-bar truss design problem

The three-bar truss design is one of the benchmarks for constrained engineering design problem and aims to minimize the volume of a truss subjected to constraints on stress (σ) by adjusting cross sectional areas. There are two decision variables as shown in Figure 4.

The problem can be formulated as follows：

Minimize:

$$f(x) = (x_2 + 2\sqrt{2}x_1) \times l \tag{24}$$

Subject to:

$$\frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + x_1x_2} P - \sigma \le 0 \tag{25}$$

$$\frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0 \tag{26}$$

$$\frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \le 0 \tag{27}$$

$$0 \le x_1, x_2 \le 1 \tag{28}$$

where:

$$l = 100\ cm,\ \sigma = 2\ kN/cm^2, P = 2\ kN/cm^2 \tag{29}$$

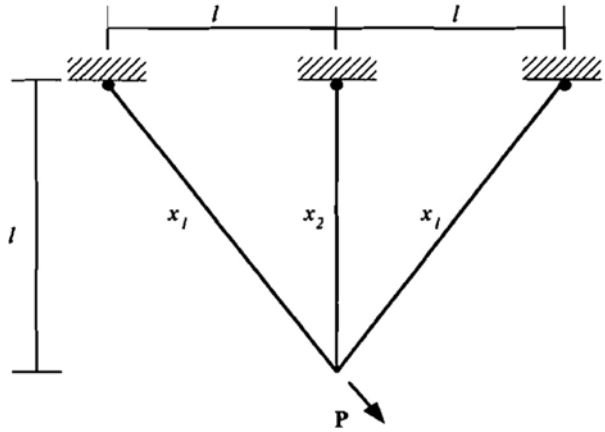

Figure 4: Three-bar truss design problem

The best solution obtained by our algorithm is (0.78867513297, 0.40824829505). The comparison and statistical results are shown in Table 2. From Table 2, it can be noted that the AMEA can obtain better solution than those by SC [Ray, 03] and MBA [Sadollah, 13] in terms of *Best*, *Worst*, *Mean* and *Std*. The results obtained by the AMEA are the same as the results obtained by PSO-DE and DSS-MDE, but the AMEA shows superiority in terms of *Std*.

Algorithm	Best	Mean	Worst	Std.
SC	263.895846	263.903356	263.969756	1.30e-2
DSS-MDE	263.895843	263.895843	263.895849	9.72e-7
PSO-DE	263.895843	263.895843	263.895843	4.50e-10
MBA	263.895852	263.897996	263.915983	3.93e-3
IDMEA	**263.895843**	**263.895843**	**263.895843**	**0**
AMEA	**263.895843**	**263.895843**	**263.895843**	3.0e-15

Table 2: Comparison of results for the three-bar truss design problem

4.3 Pressure vessel design problem

In 1994, [Kannan et al., 94] proposed the pressure vessel design problem. This problem aims to minimize the total cost, including the cost of welding, forming and the materials of the pressure vessel. There are four design variables as shown in

Figure 5, namely, T_s (the thickness of the shell), R (the inner radius), L (the length of the cylindrical section of the vessel) and T_h (the thickness of the head), where T_s and T_h are integer multiplies of 0.0625 in; R and L are continuous variables.

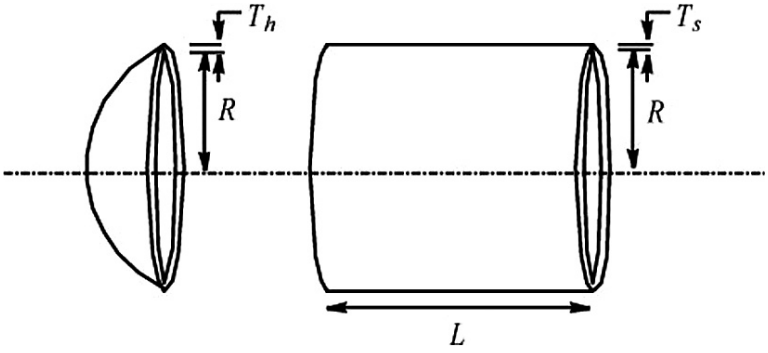

Figure 5: Pressure vessel design problem

The problem can be formulated as follows:
Minimize:

$$f = 0.6224T_s RL + 1.7781T_h R^2 + 3.1661LT_s^2 + 19.84RT_s^2 \quad (30)$$

Subject to:

$$-T_s + 0.0193R \le 0 \quad (31)$$

$$-T_h + 0.00954R \le 0 \quad (32)$$

$$-\pi LR^2 - \frac{4}{3}\pi R^3 + 1296000 \le 0 \quad (33)$$

$$L - 240 \le 0 \quad (34)$$

$$T_s, T_h \in [0, 100] \quad (35)$$

$$R, L \in [10, 200] \quad (36)$$

The best solution obtained by the AMEA is (T_s, T_h, R, L)=(0.8125, 0.4375, 42.098445595, 176.636596108). The comparison and statistical results obtained for the pressure vessel design problem are shown in Table 3. From Table 3, it can be observed that our algorithm can get better results than those obtained by GA3 [Coello, 00], GA4 [Coello, 02], CPSO [He, 07(a)], HPSO [He, 07(b)], UPSO [Parsopoulos, 05] in terms of *Best*, *Mean*, *Worst* and *Std*. However, the (μ + λ)-ES [Mezura-Montes, 05] can find better result than that obtained by our algorithm in terms of *Best but* not *Mean*, *Worst* or *Std*.

Algorithm	Best	Mean	Worst	Std.
GA3	6288.7445	6293.8432	6308.4970	7.4133
GA4	6059.9463	6177.2533	6469.3220	130.9297
CPSO	6061.0777	6147.1332	6363.8041	86.45
HPSO	6059.7143	6099.9323	6288.6770	86.20
UPSO	6154.70	8016.37	9387.77	745.869
(μ + λ)-ES	**6059.701610**	6379.938037	N.A.	210
AMEA	6059.7143412	**6075.854380**	**6370.779718**	**61.6432**

Table 3: Comparison of results for the pressure vessel problem

4.4 Tension/compression string design problem

In 1989, [Arora et al., 89] proposed the tension/compression string design problem. This problem aims to minimize the weight subject to constraints, including limits on the outside diameter, minimum deflection, surge frequency, shear stress, and design variables. There are three continuous design variables, namely, P (the number of active coils), D (the mean coil diameter) and d (the wire diameter), as shown in Figure 6.

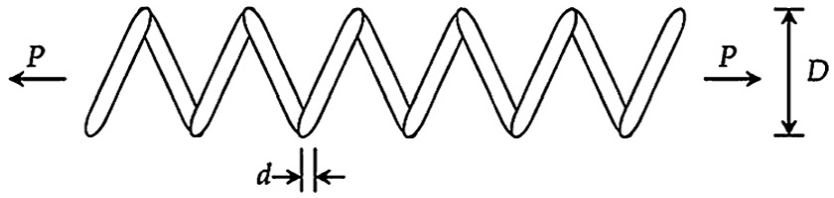

Figure 6: Tension/compression string design problem

The problem can be formulated as follows:
Minimize:

$$f = (P+2)Dd^2 \tag{37}$$

Subject to:

$$1 - \frac{PD^3}{71785d^4} \le 0 \tag{38}$$

$$\frac{4D^3 - dD}{12566(Dd^3 - d^4)} + \frac{1}{5108d^2} - 1 \le 0 \tag{39}$$

$$1 - \frac{140.45d}{D^2 P} \le 0 \tag{40}$$

$$\frac{d+D}{1.5}-1\le 0 \tag{41}$$

$$d\in[0.05,2],\quad D\in[0.25,1.3],P\in[2,15] \tag{42}$$

The best solution obtained by our algorithm is (d, D, P)=(0.05168906111, 0.35671774056, 11.2889657067). The comparison and statistical results obtained for the tension/compression string design problem are reported in Table 4. From Table 4, it can be seen that the proposed algorithm performs better than GA3 [Coello, 00], GA4 [Coello, 02], SC [Ray, 03], PSO-DE [Liu, 10], CPSO [He, 07(a)], HPSO [He, 07(b)], QPSO [Coelho, 10], PSO [Coelho, 10], (μ + λ)-ES [Mezura-Montes, 05], MBA [Sadollah, 13] in terms of *Best*, *Mean*, *Worst*. The IDMEA [Xiao, 13] obtains the same results as that by our proposed algorithm in terms of *Best*, *Mean*, and *Worst* but not *Std*.

Algorithm	Best	Mean	Worst	Std.
GA3	0.0127048	0.0127690	0.0128220	3.94e-5
SC	0.012669249	0.012922669	0.016717272	5.90e-4
GA4	0.0126810	0.0127420	0.0129730	5.90e-5
PSO-DE	0.012665233	0.012665244	0.012665304	1.20e-8
CPSO	0.0126747	0.0127300	0.0129240	5.20e-4
PSO	0.012857	0.019555	0.071802	0.011662
HPSO	0.0126652	0.0127072	0.0127190	1.58e-5
QPSO	0.012669	0.013854	0.018127.	0.001341
(μ + λ)-ES	0.012689	0.013165	N.A.	3.90e-4
MBA	0.012665	0.012713	0.012900	6.30e-5
IDMEA	0.012665233	0.012665232	0.012665232	1.42e-10
AMEA	**0.012665232**	**0.012665232**	**0. 012665232**	**6.0e-14**

Table 4: Comparison of results for the tension/compression string problem

4.5 Speed reducer design problem

In 2005, [Mezura-Montes et al., 05] proposed the speed reducer design problem. This problem aims to minimize the weight subjected to constraints, including the transverse defections of the shafts, the bending stress of the gear teeth, the stresses in the shaft and surface stress. There are seven variables as shown in Figure 7, namely, x_1 (the face width), x_2 (the module of teeth), x_3 (the number of teeth in the pinion), x_4 (the length of the first shaft between bearings), x_5 (the length of the second shaft between bearings), x_6 (the diameter of first shaft), and x_7 (the diameter of second shaft). The variable x_3 is an integer.

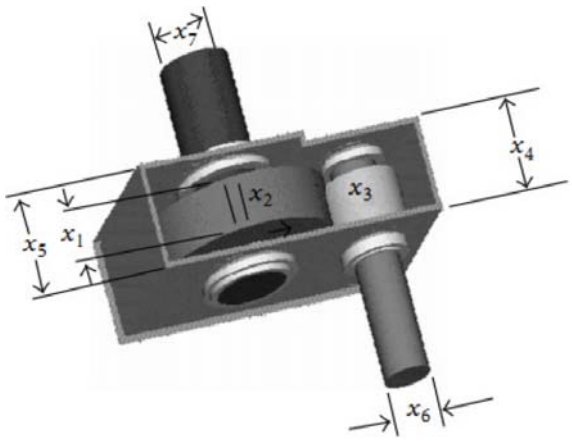

Figure 7: Speed reducer design problem

The problem can be formulated as follows:
Minimize:

$$f(x) = 0.7854x_1x_2^3(3.3333x_3^2 - 43.0934 + 14.9334x_3) + 7.4777(x_7^3 + x_6^3) - 1.508x_1(x_7^2 + x_6^2) + 0.7854(x_5x_7^2 + x_4x_6^2) \quad (43)$$

Subject to:

$$\frac{27}{x_1x_2^2x_3} \le 1 \quad (44)$$

$$\frac{397.5}{x_1x_2^3x_3^2} \le 1 \quad (45)$$

$$\frac{1.93x_4^3}{x_2x_6^4x_3} \le 1 \quad (46)$$

$$\frac{1.93x_5^3}{x_2x_7^4x_3} \le 1 \quad (47)$$

$$\frac{[16.9\times10^6 + (745(x_4 / x_3x_2))^2]^{1/2}}{110x_6^3} \le 1 \quad (48)$$

$$\frac{[157.5\times10^6 + (745(x_5 / x_2x_3))^2]^{1/2}}{85x_7^3} \le 1 \quad (49)$$

$$x_2x_3 \le 40 \quad (50)$$

$$5x_2 \le x_1 \quad (51)$$

$$x_1 \le 12x_2 \tag{52}$$

$$1.5x_6 + 1.9 \le x_4 \tag{53}$$

$$1.1x_7 + 1.9 \le x_5 \tag{54}$$

$$x_1 \in [2.6, 3.6], \quad x_2 \in [0.7, 0.8], \quad x_3 \in [17, 28] \tag{55}$$

$$x_4, x_5 \in [7.3, 8.3], \quad x_6 \in [2.9, 3.9], \quad x_1 \in [5.0, 5.5] \tag{56}$$

The best solution obtained by the AMEA is X^*=(3.5, 0.7, 17, 7.3, 7.715319911478249, 3.35021466609, 5.28665446498). The comparison and statistical results obtained for the speed reducer design problem are reported in Table 5. From Table 5, it can be observed that the proposed algorithm can obtain better results than those obtained by SC [Ray, 03], PSO-DE [Liu, 10], HEAA [Zhang, 08], SBO [Ray, 03], MDE [Montes, 06], MBA [Sadollah, 13] and IDMEA [Xiao, 13] in terms of *Best*, *Mean*, *Worst*, and *Std*. The proposed algorithm obtains the same results as that by DEDS [Zhang, 08] and DELC [Wang, 10] in terms of *Best*, *Mean*, and *Worst* but not *Std*.

Algorithm	Best	Mean	Worst	Std.
SC	2994.744241	3001.758264	3009.964736	4.00
PSO-DE	2996.348167	2996.348174	2996.348204	6.40e-6
DELC	2994.471066	2994.471066	2994.471066	1.90e-12
SBO	2994.744241	3001.758264	3009.964736	4.00e+0
DEDS	2994.471066	2994.471066	2994.471066	3.60e-12
MDE	2996.356689	2996.367220	N.A.	8.20e-3
HEAA	2994.499107	2994.613368	2994.752311	7.00e-2
MBA	2994.482453	2996.769019	2999.652444	1.56
IDMEA	2994.476877	2994.473265	2994.471555	1.32e-03
AMEA	**2994.471066**	**2994.471066**	**2994.471066**	**4.00e-15**

Table 5: Comparison of results for the speed reducer problem

5 Conclusions and Future Work

This paper develops an adaptive membrane evolutionary algorithm that combines the dynamic membrane structure and the differential evolution with the adaptive mutation factor for solving CEOPs. The proposed hybrid method employs the feasibility proportion method to adjust the size of elementary membrane dynamically in the optimization process. Additionally, the proposed algorithm demonstrates better

performance than other approaches in the literature with respect to solving five well-known engineering design optimization problems. In the future, the improved AMEA will be applied to solve various optimization problems in real engineering. Furthermore, we will extend our algorithm to solve dynamic multi-objective optimization problems.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. 61373066), Humanity and Social Science Youth Foundation of Ministry of Education of China (13YJC630010), Beijing Natural Science Foundation (No.4164096), Science and Technology Development Strategy Research Program of Tianjin (16ZLZXZF00030), and Collaborative Innovation Centre for China Economy.

References

[Arora, 89] Arora, J.S.: Introduction to optimum design. New York: McGraw-Hill, 1989.

[Akay, 12] Akay, B., Karaboga, D.: Artificial bee colony algorithm for large-scale problems and engineering design optimization. Journal of Intelligent Manufacturing, 2012, 23: 1001-14.

[Askarzadeh, 16] Askarzadeh, A.: A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Computers and Structures, 2016, 169: 1-12.

[Bulatovic, 14] Bulatovic, R.R.: Improved Cuckoo Search (ICS) algorithm for constrained optimization problems. Latin American Journal of Solids and Structures, 2014, 11: 1349-1362.

[Coello, 00] Coello, C.A.C.: Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry, 2000, 41: 112-27.

[Coello, 02] Coello, C.A.C., Mezura, M.E.: Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics, 2002, 16:193-203.

[Coelho, 10] Coelho, L.D.S.: Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Systems with Applications, 2010, 37: 1676-83.

[Deng, 16] Deng, Y.M., Zhu, W.R., Duan, H.B.: Hybrid membrane computing and pigeon-inspired optimization algorithm for brushless direct current motor parameter design. Science China-Technological Sciences, 2016, 59(9): 1435-1441.

[Huang, 06] Huang, L., Wang, N.: An optimization algorithm inspired by membrane computing. Advance in Natural Computation, 2006, 4222: 49-52.

[He, 07(a)] He, Q., Wang, L.: An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Engineering Applications of Artificial Intelligence, 2007, 20: 89-99.

[He, 07(b)] He, Q., Wang, L.: A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Applied Mathematics and Computation, 2007, 186: 1407-22.

[Huang, 07] Huang, L., He, X.X., Wang, N.: P systems based multiobjective optimization algorithm. Progress Natural Science, 2007, 17: 458-465.

[Huang, 08] Huang, L., Wang, N., Zhao, J.H.: Multiobjective optimization for controller design. ACTA Automatica Sinica, 2008, 34 (4): 472-477.

[Idowu, 13] Idowu, R.K., Maroosi, A., Muniyandi, C.R., Othman, A.Z.: An Application of Membrane Computing to Anomaly-Based Intrusion Detection System. Procedia Technology, 2013, 11: 585-592.

[Kannan, 94] Kannan, B.K., Kramer, S. N.: An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. Journal of Mechanical Design, 1994, 116: 405-411.

[Kashan, 11] Kashan, A.H.: An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA), Computer-Aided Design, 2011, 43: 1769-1792.

[Kohlia, 17] Kohlia, M., Arora, S.: Chaotic grey wolf optimization algorithm for constrained optimization problems. Journal of Computational Design and Engineering, 2017, doi: http://dx.doi.org/10.1016/j.jcde. 2017.02.005.

[Lee, 05] Lee, K.S., Geem, Z.W.: A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering, 2005, 194: 3902-3933.

[Liu, 10] Liu, H., Cai, Z., Wang, Y.: Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Applied Soft Computing, 2010, 10: 629-40.

[Mezura-Montes, 05] Mezura-Montes, E., Coello, C.A.C.: Useful infeasible solutions in engineering optimization with evolutionary algorithms. Lecture Notes in Artificial Intelligence, 2005, 3789: 652-662.

[Montes, 06] Montes, E.M., Coello, C.A.C., Reyes, J. V.: Increasing successful offspring and diversity in differential evolution for engineering design. In: Proceedings of the seventh international conference on adaptive computing in design and manufacture, 2006, 131-139.

[Melo, 13] Melo, V.V., Carosio. G.L.C.: Investigating Multi-View Differential Evolution for solving constrained engineering design problems. Expert Systems with Applications, 2013, 40: 3370-3377.

[Nishida, 04(a)]T.Y. Nishida. An application of P-system: a new algorithm for NP-complete optimization problems. The 8th World Multi-Conference on Systems, Cybernetics and Informatics, 2004, 109-112.

[Nishida, 04(b)] Nishida, T.Y.: An approximate algorithm for NP-complete optimization prob-lems exploiting P-systems, in: The Brainstorming Workshop on Uncertainty inMembrane Computing, 2004, 185-192.

[Niu, 11] Niu, Y.Y., Pan, L.Q., Perez-Jimenez, M.J.: A tissue P systems based uniform solution to tripartite matching problem. Fundamenta Informaticae, 2011, 109: 1-10.

[OnoltIshdorj, 10] OnoltIshdorj, T., Leporati, A., Pan, L.Q.: Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-Computed resources. Theoretical Computer Science, 2010, 411: 2345-2358.

[Parsopoulos, 05] Parsopoulos, K., Vrahatis, M.: Unified particle swarm optimization for solving constrained engineering optimization problems. Advances in natural computation LNCS, 3612. Berlin: Springer-Verlag, 2005, 582-91.

[Pan, 10] Pan L.Q., Paun, G.: Spiking neural P systems: an improved normal form. Theoretical Computer Science, 2010, 411: 906-918.

[Pan, 11] Pan, L.Q., Zeng, X.X., Zhang, X.Y.: Time-free spiking neural P systems. Neural Comput, 2011, 23: 1-23.

[Peng, 15] Peng, H., Jiang, Y., Wang, J., Pérez-Jiménez, M.J.: Membrane clustering algorithm with hybrid evolutionary mechanisms, Journal of Software, 2015, 26(5): 1001-1012.

[Pan, 16] Pan, L.Q., Paun, G., Song, B.S.: Flat maximal parallelism in P systems with promoters. Theoretical Computer Science, 2016, 623: 83-91.

[Ray, 03] Ray, T., Liew, K.M.: Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE Transactions on Evolutionary Computation, 2003, 7: 386-96.

[Storn, 97] Storn, R., Price, K.: Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341-59.

[Sadollah, 13] Sadollah, A., Bahreininejad, A., Eskandar, H., Hamdi, M.: Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. Applied Soft Computing, 2013, 13: 2592-2612.

[Song, 14] Song, T., Macías-Ramos, L.F., Pan, L.Q., Pérez-Jiménez, M.J.: Time-free solution to SAT problem using P systems with active membranes. Theoretical Computer Science, 2014, 529: 61-68.

[Song, 15] Song, T., Pan, L.Q.: Spiking neural P systems with rules on synapses working in maximum spikes consumption strategy. IEEE Transactions on NanoBioscience, 2015, 14: 38-44.

[Singh, 16] Singh, G., Deep, K.: A new membrane algorithm using the rules of Particle Swarm Optimization incorporated within the framework of cell-like P-systems to solve Sudoku. Applied Soft Computing, 2016, 45(C): 27-39.

[Song, 17] Song, B.S., Zhang, C., Pan, L.Q.: Tissue-like P systems with evolutional symport/ antiport rules. Information Sciences, 2017, 378: 177-193.

[Wang, 08] Wang, J., Yin, Z.: A ranking selection-based particle swarm optimizer for engineering design optimization problems. Structural and Multidisciplinary Optimization, 2008, 37: 131-147.

[Wang, 10] Wang, L., Li, L.P.: An effective differential evolution with level comparison for constrained engineering design. Structural and Multidisciplinary Optimization, 2010, 41: 947-963.

[Wang, 10] Wang J., Hoogeboom, H.J., Pan, L.Q.: Spiking neural P systems with weights. Neural Computing, 2010, 22: 2615-2646.

[Wang, 13] Wang, J., Shi, P., Peng, H.: Membrane computing model for IIR filter design. Information Sciences, 2016, 329: 164-176.

[Wang, 15] Wang, L., Zeng, Y., Chen, T.: Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. Expert Systems with Applications, 2015, 42: 855-863.

[Xiao, 16] Xiao, J.H, He, J.J.: An improved dynamic membrane evolutionary algorithm for constrained engineering design problems. Nat Computing, 2016, 15: 579-589.

[Xu, 16] Xu, J.: Probe machine. IEEE Transactions on Neural Network and Learning Systems, 2016, 27: 1405-1416.

[Yan, 14] Yan, A.J., Shao, H.S., Guo, Z.: Weight optimization for case-based reasoning using membrane computing. Information Sciences, 2014, 287:109-120.

[Zhang, 08] Zhang, M., Luo, W., Wang, X.F.: Differential evolution with dynamic stochastic selection for constrained optimization. Information Sciences, 2008, 178(15): 3043-74.

[Zhao, 11] Zhao, J.H., Wang, N.: A bio-inspired algorithm based on membrane computing and its application to gasoline blending scheduling. Computers & Chemical Engineering, 2011, 35(2): 272-283.

[Zhao, 12] Zhao, J.P., Wang, L., Zeng, P., Fan, W.H.: An effective hybrid genetic algorithm with flexible allowance technique for constrained engineering design optimization. Expert Systems with Applications, 2012, 39: 6041-6051.

[Zhang, 13] Zhang, G., Cheng, J., Gheorghe, M., Meng, Q.: A hybrid approach based on different evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems. Applied Soft Computing, 2013, 13(3): 1528-1542.

[Zeng, 14(a)] Zeng, X.X., Zhang, X.Y., Song, T., Pan, L.Q.: Spiking neural P systems with thresholds. Neural Computation, 2014, 26: 1340-1361.

[Zeng, 14(b)] Zeng, X.X., Xu, L., Liu, X.R., Pan, L.Q.: On languages generated by spiking neural P systems with weights. Information Sciences, 2014, 278: 423-433.

[Zeng, 14(c)] Zeng, X.X., Pan, L.Q., Pérez-Jiménez, M.J.: Small universal simple spiking neural P system with weights. Science China: Information Science, 2014, 57: 92-102.

[Zhang, 15(a)] Zhang, X.Y., Pan, L.Q., Paun, A.: On the universality of axon P systems. IEEE Transactions on Neural Networks and Learning Systems, 2015, 26: 2816-2829.

[Zhang, 15(b)] Zhang, X.Y., Tian, Y., Jin, Y.C.: A knee point driven evolutionary algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation, 2015, 19(6): 761-776.

[Zhang, 15(c)] Zhang, X.Y., Tian, Y., Cheng, R., Jin, Y.C.: An efficient approach to non-dominated sorting for evolutionary multi-objective optimization. IEEE Transactions on Evolutionary Computation, 2015, 19(2): 201-213.

[Zhang, 16(a)] Zhang, X.Y., Tian, Y., Cheng, R., Jin, Y.C.: A decision variable clustering based evolutionary algorithm for large-scale many-objective optimization. IEEE Transactions on Evolutionary Computation, 2016, 10.1109/TEVC.2016.2600642.

[Zhang, 16(b)] Zhang, X.Y., Tian, Y., Jin, Y.C.: Approximate non-dominated sorting for evolutionary many-objective optimization. Information Sciences, 2016, 369(10): 14-33.

[Zhang, 16(c)] Zhang, X.Y., Li, J., Zhang, L.: A multi-objective membrane algorithm guided by the skin membrane. Nature Computing, 2016, 15(4): 597-610.