Evolutionary Computation Based on Membrane Computing

Liang Huang, Il Hong Suh
Intelligence and Communications for Robots Laboratory, College of Information and Communications, Hanyang University
e-mail: {hl, ilsuh}@incintl.hanyang.ac.kr

Abstract
Membrane computing is extended as an optimization algorithm. The algorithm inherits the evolutionary idea from evolutionary computing and the framework from membrane computing. Tested by two benchmark functions, the excellent performance of the new algorithm is described by the comparison with other algorithms.

1. Introduction
Many intractable problems are difficult to be solved by conventional evolutionary algorithms. Therefore, an evolutionary computation based on membrane computing (ECMC) is investigated. As a theoretic frame of computing devices, membrane computing is difficult to solve the optimization problems directly [1]. In ECMC, the frame of membrane computing is explored and the idea of evolutionary computing is inherited.

2. ECMC algorithm
The mechanism of the ECMC is illustrated by the following example. The test functions with different difficulty are shown below [2]:

\[
f(x) = \min(\sum x_i) ; x \in [-5,12,5,12] \tag{1}
\]

\[
f(x) = \min(\sum (x_i-10\cos(2\pi x_i))+10) ; x \in [-5,12,5,12] \tag{2}
\]

In principle, the structures of membrane computing are flexible and unlimited. In these optimization cases, the structure as Figure 1 is adopted. There are many strings of the independent variables in each membrane. These strings are feasible solutions which are called as the objects or chromosomes. The algorithm is realized according to these following steps.

[Figure 1] Cell-like membrane structure

Step 0: The algorithm structure and objects in membrane is designed and the parameters is given on the initial step. In following experiments, there are 100 objects in the skin \( m \), 25 objects in membrane \( m_1 \) and \( m_2 \), and 10 objects in each membrane \( m_i \) to \( m_n \).

Step 1: The objects in each membrane evolve according to their own associated rules.

Step 1.1: An object \( S \) evolve into \( S’ \) according to the evolution rule as formula (3).

\[
\begin{align*}
S &= x_1, x_2, \ldots, x_i \\
S’ &= y_1, y_2, \ldots, y_l \\
y_i &= x_1 \text{ or } x_i + d
\end{align*}
\tag{3}
\]

where, \( l \) is the length of the strings, \( x, y \) are the variables of a function. \( d \) is a uniformly...
distributed random number. The distribution ranges are various in different membranes.

Step 1.2: The objects \( U, V \) are recombined as \( Z, W \), where \( U = w_k U_k V = w_k V_k Z = w_k Z_k W = w_k W_k \) and \( u_k, u_k, w_k, w_k \) are substrings of the variables. These called splicing rule or crossover rule.

Step 1.3: The selection rule exploits the idea of the heat motion of molecule. The selection rule is simplified as the following formula.

\[
\begin{align*}
S_{\text{new}} &= S_i, f_i = f_i, \text{ if } f_i \geq f_s, \\
S_{\text{new}} &= S_i, f_i = f_j, \text{ if } f_j < f_i, P > P, \\
S_{\text{new}} &= S_i, \text{ if } f_j < f_i, P \leq P;
\end{align*}
\]

where, \( S_i \) is an arbitrary string in a membrane. \( S_a \) is the string which has good fitness at last comparison, \( S_{\text{new}} \) is the new string which is produced according to the rule for the next generation: \( f_a \) is the fitness of \( S_a \) which was considered as a good string at the last comparison; \( f_i \) is the fitness of the current string \( S_i \); \( P_k = 0.21/g \in (0,1) \) is the given probability of the string \( S_i \) that is copied to the next generation; \( g \) is the generations which the systems has evolved; \( P_j(0,1) \) is a random constant for the current string \( S_j \). The rule describes that a string \( S_i \) will be selected and copied if its fitness \( f_i \) is higher than the fitness \( f_a \) of the last copied string \( S_a \). Otherwise it will be copied with a certain probability \( P_i \).

Step 1.4: The communication rule adopts the ideas of biochemistry and the heat motion of molecules. It is expressed by the following formula:

\[
P_{\text{communication}} = e^{-(f_{a} - f_{s})/g}
\]

where, \( f_{s} \) is the fitness of a string; \( f_{a} \) is the best fitness of the last generation; \( k \) is a given constant. The generations is represented by \( g \). The rule means that the string \( S_{\text{new}} \) will drill through the membrane if \( f_{a} \geq f_{s} \). Otherwise, it will pass the membrane into the outer membrane with the probability \( P_{\text{communication}} \) and replaced the worst string in the outer membrane.

Step 2: In a serial computer, each membrane evolves in turn. If the halt condition is not satisfied, the procedure jumps to Step 1 for simulating the evolution of another membrane.

After 2,500,000 function evaluations on \( f_1 \) and 5,000,000 function evaluations on \( f_2 \), the results of several algorithms is shown in Table 1 [2,3]. The table shows that ECMC converges fast with the highest precision. In fact, the FADE arrives at 258.49 with 5,000,000 function evaluations on the Rastrigin function. However, ECMC only costs 248,000 function evaluations to obtain the same value.

<table>
<thead>
<tr>
<th>Functions</th>
<th>CGA</th>
<th>DE</th>
<th>FADE</th>
<th>ECMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>33.94</td>
<td>39.7</td>
<td>2.35e-10</td>
<td>3.0e-12</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>357.15</td>
<td>472.68</td>
<td>258.49</td>
<td>2.378e-7</td>
</tr>
</tbody>
</table>

3. Conclusions

Membrane computing is extended for the optimization. It converges fast with high precision. The excellent performance is confirmed by the simulation experiments compared with other algorithms.

References