Article

Chinese Science Bulletin

Computer Science & Technology

February 2012 Vol.57 No.6: 698–706
doi: 10.1007/s11434-011-4928-7

A membrane evolutionary algorithm for DNA sequence design in DNA computing

XIAO JianHua[1*], ZHANG XingYi[2] & XU Jin[3]

[1] *The Research Center of Logistics, Nankai University, Tianjin 300071, China;*
[2] *Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230039, China;*
[3] *School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China*

Received June 3, 2011; accepted October 21, 2011

DNA sequence design has a crucial role in successful DNA computation, which has been proved to be an NP-hard (non-deterministic polynomial-time hard) problem. In this paper, a membrane evolutionary algorithm is proposed for the DNA sequence design problem. The results of computer experiments are reported, in which the new algorithm is validated and out-performs certain known evolutionary algorithms for the DNA sequence design problem.

DNA computing, membrane computing, P system, DNA sequence design

Citation: Xiao J H, Zhang X Y, Xu J. A membrane evolutionary algorithm for DNA sequence design in DNA computing. Chin Sci Bull, 2012, 57: 698–706, doi: 10.1007/s11434-011-4928-7

Membrane computing is an emergent branch of natural computing, first introduced by Paun [1]. This unconventional model of computation is a type of distributed parallel system, which is inspired by the structure and function of living cells. The devices of this model are called P systems. Roughly speaking, a P system consists of a cell-like membrane structure, in the compartments of which there are multisets of objects that evolve according to given rules in a synchronous, non-deterministic maximally parallel manner. Many different classes of such computing devices have already been investigated. Most of them are computationally universal, i.e., able to compute whatever a Turing machine can do [2–4], and are computationally efficient, i.e., able to trade space for time and in this way solve presumably intractable problems in a feasible time (e.g., [5–9]). Membrane computing is very attractive from a computational point of view because of its hierarchical structure and intrinsic parallelism.

Evolutionary algorithms are robust optimization and search methods inspired by evolutionary processes occurring in natural selection and molecular genetics. They are approximation algorithms that seek a trade-off between solution quality and computational costs. The main features of evolutionary algorithms are the representation and evaluation of individuals, population dynamics, and evolutionary operators, such as selection, crossover, and mutation [10]. Evolutionary algorithms exhibit an intrinsic parallelism derived from dealing with multiple individuals, show remarkable adaptability and flexibility to various applications, and provide good search capabilities and robust results [11].

Both P systems and evolutionary algorithms are nature-inspired models and are used to solve complex problems; however, they are different with respect to the objects and rules used and in the computational strategies employed. While P systems represent a suitable formal framework for parallel-distributed computation, evolutionary algorithms are very effective for implementing different algorithms to solve numerous problems [11]. Thus, the possible interaction between P systems and evolutionary algorithms represents a fertile research field, as suggested by the list of 26 open problems in membrane computing [12]. The first attempt in this direction was by Nishida [13,14], who devel-

*Corresponding author (email: jhxiao@nankai.edu.cn)

oped membrane (evolutionary) algorithms to solve the traveling salesman problem. In those membrane algorithms, membrane structure was used together with ideas from genetic algorithms (cross-over and mutation operators), and the traveling salesman problem was solved. A membrane algorithm was also employed to solve the minimum storage problem [15]. The quantum-inspired evolutionary algorithm based on P systems was also developed to solve the knapsack problem and the radar emitter signals problem [16,17]. The similarities between distributed evolutionary algorithms and P systems have been analyzed and new variants of distributed evolutionary algorithms were suggested and applied for some continuous optimization problems [18]. In this paper, a membrane evolutionary algorithm is proposed to solve the DNA sequence design problem.

1 The DNA sequence design problem

DNA computing is a new paradigm of computation. DNA computing is attractive both theoretically and technically because of its intrinsic parallelism. DNA computing has been used to solve various computationally complex problems, such as the Hamiltonian problem [19], the SAT problem [20,21], the Steiner tree problem [22], the maximal clique problem [23], and the maximum independent set problem [24]. Because the problems solved by DNA computing are encoded by a DNA sequence, the design of DNA sequences is crucial for successful DNA computation. To make a set of DNA sequences effective in DNA computing, they must fulfill a number of combinatorial and thermodynamic constraints. Such a task is not easy to achieve, and it has been shown that designing a set of good DNA sequences is an NP-hard (non-deterministic polynomial-time hard) problem [25,26].

Various algorithms and methods have been proposed for reliable DNA sequence design. Marathe et al. [27] proposed a dynamic programming approach. Frutos et al. [28] developed a template strategy to select a great number of dissimilar sequences. Arita et al. [29] introduced a genetic algorithm into the DNA sequence design system and proposed a random generate-and-test algorithm. Tanaka et al. [30] applied simulated annealing to optimize the set of DNA sequences. Cui et al. [31] proposed DNA sequence design algorithm based on the PSO optimization. Wang et al. [32] developed the GA/SA algorithm for DNA sequence design. Qiu et al. [33] designed a hardware microprocessor to discover the DNA code under thermodynamic constraints. Dyachkov et al. [34] introduced the concept of a stem similarity function and discussed DNA codes based on stem similarity. Kawashimo et al. [35] presented a local search based algorithm for designing DNA short sequence sets satisfying thermodynamic constraints with minimum free energy criteria. Zhang et al. [36] proposed an invasive weed optimization algorithm to optimize encoding sequences.

Successful DNA computing relies heavily on designing or selecting proper DNA sequence to realize desired chemical reactions and solution extractions. To ensure that chemical reactions are controllable, some thermodynamic and combinatorial constraints must be satisfied in the design of DNA sequences.

1.1 Sequence design constraints

In the following context, $x_i(1 \leqslant i \leqslant m)$ and $x_j(1 \leqslant j \leqslant m)$ are used to denote DNA sequences with length n, and m is the cardinality of a set of DNA sequences. For convenience, DNA sequence x is oriented from the 5′ to 3′ end, and the reverse orientation is the 3′ to 5′ end. The Watson-Crick complement of a sequence x is denoted by $\overline{x}$, while the reverse sequence of sequence x is denoted by x^R.

(1) Thermodynamic constraint. **T1** (melting temperature): Melting temperature is an important factor in the efficiency of a DNA reaction, and can be used to select DNA sequences with uniform melting temperatures. The nearest neighbor mode [37] was used to calculate melting temperature, and the evaluation function $F_{Tm}(\sum)$ of the melting temperature is defined as follows:

$$F_{Tm}(\sum) = \sum_{i=1}^{i=m} f_{Tm}(x_i), \tag{1}$$

$$f_{Tm}(x_i) = [Tm_{\text{tar}}(x_i) - Tm_{\text{gen}}(x_i)]^2, \tag{2}$$

$$Tm_{\text{gen}} = \frac{\Delta H^{\circ}}{\Delta S^{\circ} + R\ln(C_T/4)} - 273.15, \tag{3}$$

where Tm_{tar} is the target melting temperature, $Tm_{\text{gen}}(x_i)$ is the melting temperature of the generated sequence x_i, R is the gas constant (1.987 cal mol^{-1} K^{-1}), ΔH° and ΔS° are the enthalpy and the entropy, respectively, and C_T is the salt concentration.

(2) Combinatorial constraints. **C1** (similarity measure constraint): The similarity measure [38] is used to describe the degree of similarity of two DNA sequences, and ensures that each sequence is as unique as possible. The evaluation function $F_{\text{Similarity}}(\sum)$ of the similarity measure constraint is defined by eqs. (4) and (5).

$$F_{\text{Similarity}}(\sum) = \sum_{i=1}^{i=m} f_{\text{Similarity}}(x_i), \tag{4}$$

$$f_{\text{Similarity}}(x_i) = \sum_{\substack{1 \leqslant j \leqslant m \\ j \neq i}} \max_{0 \leqslant g \leqslant n} \max_{0 \leqslant k \leqslant n+g-1} S(x_i(-)^g x_i, \sigma^k(x_j)), \tag{5}$$

where $(-)^g$ denotes g gabs, $\sigma^k(x_j)$ denotes the k position right shift for DNA sequence x_j, $S(*,*)$ is the number of corresponding places where two characters are the same. For more information, refer to [38].

C2 (H-measure constraint): The H-measure is akin to the similarity measure. The difference is that the H-measure

compares two given sequences from opposite directions, while the similarity measure works in the same direction. The H-measure computes how many nucleotides are complementary between the given sequences to prevent cross hybridization of two DNA sequences. Like the similarity measure, the H-measure also uses the shift sequences. The evaluation function of the H-measure measure constraint is defined as follows:

$$F_{\text{H\_measure}}(\sum)=\sum_{i=1}^{i=m} f_{\text{H\_measure}}(x_i), \tag{6}$$

$$f_{\text{H\_measure}}(x_i)=\sum_{\substack{1\leqslant j\leqslant m\\ j\neq i}} \max_{0\leqslant g\leqslant n} \max_{0\leqslant k\leqslant n+g-1} C(x_i(-)^g x_i, \sigma^k(x_j^R)), \tag{7}$$

where $C(x_i(-)^g x_i, \sigma^k(x_j^R))$ is the number of corresponding places where two nucleotides are the same. For more information, refer to [38].

C3 (hairpin structure constraint): Hairpin structure consists of a ring part and a stem part, which can hybridize itself by forming a loop. The measure of hairpin structure constraint calculates the probability to form a secondary structure. Generally, the hairpin structure is not desirable for DNA encoding. The equations are defined as follows [38]:

$$F_{\text{Hairpin}}(\sum)=\sum_{i=1}^{m} f_{\text{Hairpin}}(x_i), \tag{8}$$

$$f_{\text{Hairpin}}(x_i)=\sum_{r}^{(n-2*pinlen)} \sum_{c=pinlen+\lceil r/2\rceil}^{(n-pinlen-\lfloor r/2\rfloor)} \text{Hairpin}(x_i,c), \tag{9}$$

where r is the minimum length to form a hairpin, *pinlen* denotes the minimum length of the stem. A hairpin structure is formed at position c for sequence x_i, if more than half of the bases in the subsequence $x_{c-pinlen},\cdots,x_c$ hybridize to the subsequence $x_{c+r},\cdots,x_{c+r+pinlen}$. Hairpin (x_i,c) is 1; if DNA sequence x_i can form a hairpin structure, the value of Hairpin (x_i,c) is 0.

C4 (GC content constraint): The GC content is the percentage of G and C in a DNA sequence. It is important to arrange the GC content such that the chemical character is uniform. Thus, the evaluation function $f_{GC}(\sum)$ of the GC content constraint is described as follows:

$$F_{\text{GC}}(\sum)=\sum_{i=1}^{m} f_{\text{GC}}(x_i), \tag{10}$$

$$f_{\text{GC}}(x_i)=[\text{GC}_{\text{gen}}(x_i)-\text{GC}_{\text{tar}}(x_i)]^2, \tag{11}$$

where $\text{GC}_{\text{tar}}(x_i)$ is the target value of GC content of DNA sequence x_i, and $\text{GC}_{\text{gen}}(x_i)$ is the GC content of the generated sequence.

C5 (continuity constraint): Continuity is often used to describe the degree of successive occurrence of the same base in a sequence. In a DNA sequence, if the same base occurs continuously, the reaction is not well controllable because they may cause an unexpected biological structure. The formulations are defined as follows [38]:

$$F_{\text{Con}}(\sum)=\sum_{i=1}^{m} f_{\text{Con}}(x_i), \tag{12}$$

$$f_{\text{Con}}(x_i)=\sum_{j=1}^{n-t+1} \sum_{\alpha\in\{\text{A,T,C,G}\}} T(C_\alpha(x_i,j),t)^2, \tag{13}$$

$$C_\alpha(x_i,j)=\begin{cases} c, & \text{if there is } c \text{ such that } x_i\neq\alpha, x_i^{j+k}=\alpha,\\ & \text{for } 1\leqslant k\leqslant c, x_i^{j+k+1}\neq\alpha,\\ 0, & \text{otherwise,}\end{cases} \tag{14}$$

$$T(i,j)=\begin{cases} i, & \text{if } i>j,\\ 0, & \text{otherwise,}\end{cases} \tag{15}$$

where $C_\alpha(x,i)$ is the number of the i-th base α occurring continuously in DNA sequence x, and t is the target.

1.2 Evaluation function

In DNA sequence optimization, the content can be naturally regarded as a constraint, not an objective function. So, more precise formulation of DNA sequence optimization is a constrained multi-objective optimization problem.

Using the weight-sum method, the multi-objective optimization for DNA encoding can be transformed into a single objective optimization problem. The fitness function can be described as follows:

$$\begin{aligned}&\text{Fitness}(\sum)=\sum_i w_i F_i(\sum)\\ &\quad i\in\{Similarity, H\_measure, Con, Hairpin, Tm\}\\ &\textit{Subject to}\\ &\quad \{F_{GC}(\sum)\}.\end{aligned} \tag{16}$$

For simplicity, we set each weight w_i to unity. In this paper, the constraint handling technology [39] is applied. For more details, refer to [30].

2 A membrane evolutionary algorithm for DNA sequence design

Before the membrane evolutionary algorithm is described in detail, some concepts related to string object P systems are briefly introduced.

2.1 A string object P systems

In this paper, several basic notions of membrane computing are introduced. For more information, refer to [1].

The membrane structure of a P system, shown in Figure 1, consists of several membranes arranged in a hierarchical structure inside a main membrane, called the skin. A membrane without any other membranes inside is said to be elementary. A space delimited by one membrane and its immediately lower membranes is called a region, and the region of an elementary membrane is the space delimited by it. Each region can contain a multiset of objects and a set of evolutionary rules, by which objects can evolve, and communication rules, by which objects can be moved between regions.

In string object P systems, the membranes can be marked with + or −, and this is interpreted as an "electrical charge", or with 0, and this means "neutral charge". We will denote the three cases by $[_i]_i^+$, $[_i]_i^-$ and $[_i]_i^0$.

A string object P system with active membranes and priority relations among the rules from each region is defined as follows:

$$\Pi = (O, u, L_1, \cdots, L_n, (R_1, \rho_1), \cdots, (R_n, \rho_n), i_0),$$

where:

(i) O is an alphabet of objects; its elements are called objects;

(ii) μ is a membrane structure consisting of n membranes (and hence the regions) injectively labeled with $1, 2, \cdots, n$, $n \geqslant 1$; n is called the degree of the system $\prod$;

(iii) $L_i (1 \leqslant i \leqslant n)$, are strings that represent multisets over O, L_i is initially placed in region i;

(iv) ρ_i is a partial order relation on R_i, $1 \leqslant i \leqslant n$. Often, instead of $(r_1, r_2) \in \rho_i$ we will write $r_1 > r_2$. This symbol has the following meaning: a rule $r_1 : u_1 \to v_1$ from a set R_i is used in a transition step with respect to $\prod$ only if there is no rule $r_2 : u_2 \to v_2$ in R_i which can be applied at the same step and $r_2 > r_1$;

(v) $i_0 \in \{1, 2, \cdots, n\}$ is the label of the output membrane;

(vi) R_i $(1 \leqslant i \leqslant n)$, are finite sets of evolution rules over O^*, R_i is associated with region i of μ, and it is of the following forms:

(a) $[_i s \to s_1 \| \cdots \| s_k]_i^\alpha, k \geqslant 1, \alpha \in \{+, -, 0\}, s_1, s_2, \cdots, s_k \in O^*$.

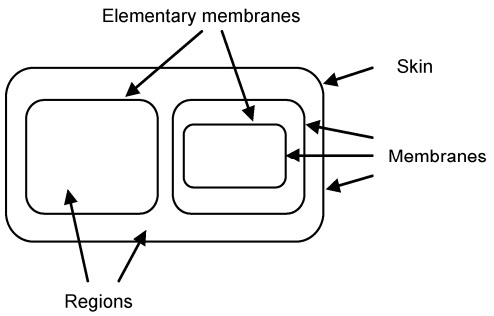

Figure 1 A membrane structure.

Mutation rule $M(s)$: This is similar to the uniform mutation operator of a genetic algorithm, which is a simple replacement with a randomly selected character within a given alphabet set of objects. It is described in detail as follows. Assign every bit as a mutation point in turn. For each mutation point, we select a random value between 0 and 1 and compare this with the mutation probability p_m. If the random value is less than p_m, a random character is substituted for the original character.

Mutation rules with replication work on string objects, are associated with membranes and depend on the label and the charge of membranes, but do not directly involve the membranes, in the sense that the membranes are neither taking part in the application of these rules nor are they modified by them. Moreover, one string can evolve to more than one string in that membrane using a rule of this type.

(b) $[_i(s_1, s_2) \to (s_3, s_4)]_i^\alpha, \alpha \in \{+, -, 0\}, s_1, s_2, s_3, s_4 \in O^*$.

Crossover rule $C(s_1, s_2)$: A rule of this type works on a couple of string objects and can possibly be used only if the polarization of the membrane is α.

Similar to the binary genetic algorithm, we adopted a simple crossover. Let n be the dimension of the string and p_c be the crossover rate, now choose a pair of strings:

$$s_1 = (x_1, \cdots, x_{pos1}, \cdots, x_{pos2}, \cdots, x_n),$$
$$s_2 = (y_1, \cdots, y_{pos1}, \cdots, y_{pos2}, \cdots, y_n),$$

and randomly generate a decimal r. If $r < p_c$, apply a simple two-point crossover to them as follows. Generate two random integers $pos1$ and $pos2$ in the interval [1, n]. The components of two individuals between the numbers $pos1$ and $pos2$ will be exchanged. Then the new strings are generated as follows:

$$s_3 = (x_1, \cdots, y_{pos1}, \cdots, y_{pos2}, \cdots, x_n),$$
$$s_4 = (y_1, \cdots, x_{pos1}, \cdots, x_{pos2}, \cdots, y_n).$$

By crossing a string with another string, two new strings are created in region i and also the polarization of the membrane is not modified.

(c) $s_1[_i]_i^{\alpha_1} \to [_i s_2]_i^{\alpha_2}, \alpha_1, \alpha_2 \in \{+, -, 0\}, s_1, s_2 \in O^*$.

Communication rule: An object is introduced in the membrane, possibly modified during this process; the polarization of the membrane can be modified, but not its label.

(d) $[_i s_1]_i^{\alpha_1} \to [_i]_i^{\alpha_2} s_2, \alpha_1, \alpha_2 \in \{+, -, 0\}, s_1, s_2 \in O^*$.

Communication rule: An object is sent out of the membrane, which is possibly modified during this process; the polarization of the membrane can also be modified, but not its label.

In P systems, the computation is a sequence of transitions between the configurations starting from the initial configuration by applying the rules associated with regions in a non-deterministic maximally parallel manner. The compu-

tation will halt in each region of the system when no rule can be applied. The result of the computation is collected in membrane i_0 when the system arrives at the final configuration. For more details about P systems information, refer to [2].

2.2 Membrane evolutionary algorithm for DNA sequence design

For convenience, we first provide some definitions and notations before constructing the P systems for DNA sequence design. Ω, Ω' and Ω'' are defined as DNA sequences with the same length over $\{A,C,G,T\}$, $\{A',C',G',T'\}$ and $\{A'',C'',G'',T''\}$, respectively.

Formally, the new system with active membranes and priority relation is constructed as follows:

$$\Pi = (O,u,L_0,L_1,L_1',\cdots,L_n,L_n',R_1,R_1',\cdots,R_n,R_n',n)$$

where:

$$O = \{A,C,G,T,A',C',G',T',A'',C'',G'',T''\} U \{p_i \mid 0 \leqslant i \leqslant 5\}$$
$$U \{d_i \mid 0 \leqslant i \leqslant m\} U \{t_i \mid 0 \leqslant i \leqslant 5\},$$
$$u = [_0[_1^0[_{1'}^0]_{1'}^0[_2^0[_{2'}^0]_{2'}^0 \cdots [_n^0[_{n'}^0]_{n'}^0]_n^0 \cdots]_2^0]_1^0]_0^0,$$
$$L_0 = \Phi, L_i = \{s_{i1}^i, s_{i2}^i, p_0\},\ 1 \leqslant i \leqslant n-1,$$
$$L_i = \{s_{i1}^i, s_{i2}^i, p_0\},$$
$$L_i' = \{t_0\},\ 1 \leqslant i \leqslant n,$$

The set $R_0 = \varnothing$ and the sets $R_i\ (1 \leqslant i \leqslant n)$, contain the following rules:

(1) $[_i s \to s' \,||\, M(s)]_i^0, s \in \Omega, s' \in \Omega', M(s) \in \Omega''$.

Using this rule, two DNA sequences can be obtained. One is obtained by first mutating the initial one, and then by attaching two primes to its every symbol; the other is obtained by attaching a prime to each symbol of the initial one.

(2) $[_i s' \to s]_i^0, s \in \Omega, s' \in \Omega'$.

The prime is removed using this rule.

(3) $[_i (s_1'', s_2'') \to C(s_1'', s_2'')]_i^0, s_1'', s_2'' \in \Omega'', C(s_1'', s_2'') \in \Omega \times \Omega$.

Two new DNA sequences can be created using the rule as above and their two primes are also removed.

(4) $[_i s]_i^+ \to [_i]_i^- s, s \in \Omega$.

A DNA sequence is sent out of a region and the polarization of the membrane is translated from + to −. For this type of rule, priority is given as follows: for any $s_1, s_2 \in \Omega$, if there is $f(s_1) > f(s_2)$, then the corresponding rule $[_i s_1]_i^+ \to [_i]_i^- s_1$ has a higher priority than the rule $[_i s_2]_i^+ \to [_i]_i^- s_2$, where f is the fitness function.

(5) $s[_i]_i^- \to [_i s]_i^0, s \in \Omega$.

A DNA sequence is introduced into a region from the outside region and the polarization of the membrane is translated from − to 0. For this type of rule, priority is given as follows: for any $s_1, s_2 \in \Omega$, if there is $f(s_1) < f(s_2)$, then the corresponding rule $s_1[_i]_i^- \to [_i s_1]_i^0$ has a higher priority than the rule $s_2[_i]_i^- \to [_i s_2]_i^0$, where f is the fitness function.

Using the rules of types (4) and (5), the worst and best DNA sequences in every region are sent to the outer region and adjacent inner region, respectively.

(6) $[_i p_0]_i^0 \to [_i]_i^0 p_1$.

(7) $p_1[_i]_i^0 \to [_i p_2]_i^+$.

(8) $[_i p_2 \to p_3]_i^+$.

(9) $[_i p_3 \to p_4]_i^-$.

(10) $[_i p_4 \to p_5]_i^0$.

(11) $[_i p_5 \to p_0]_i^0$.

For the above rules, there are priority relationships as follows: the rule of type (10) always has a higher priority than the rules of type (1); the rule of type (11) always has a higher priority than the rules of type (1).

The role of objects $p_i (0 \leqslant i \leqslant 5)$ is to cause the rules of types (1), (2) and (3), (4), (5) be used in turn, and wait two steps for the two worst DNA sequences to be deleted. Initially, there is a p_0 in every region $i (1 \leqslant i \leqslant n)$, and then only the rules of type (1) are used together with the rule of type (6), and p_0 is sent out of membrane i and becomes p_1. Moreover, the polarization of each membrane i is unchanged; in the next step, the rules of types (2) and (3) can be used. At the same time, using the rule of type (7), the polarization of each membrane is translated from 0 to +, and p_1 is introduced in membrane i and becomes p_2; then the rules of type (4) can be used, which makes the polarization become −. Moreover, the rule of type (8) can also be used, which makes p_2 translate to p_3. In the next step, the rules of types (5) and (9) can be used simultaneously; therefore, the polarization is translated from − to 0 and p_3 becomes p_4. The rules of types (10) and (11) always have a higher priority than the rules of type (1); therefore, in the following two steps, the rules of types (10) and (11) must be used in turn, the polarization keeps unchanged and p_4 becomes p_5, and then becomes p_0 again.

(12) $[_n d_i \to d_{i+1}]_n^-, 0 \leqslant i \leqslant m-1$.

The objects d_i are counters. Initially, d_0 is placed in membrane n. In each computation period, a rule of this type can be used, and there is only one rule of this type being used, which is used together with the rule of type (9). Therefore, the subscript of d_i only adds one in each computation.

(13) $[_{i'} t_i \to t_{i+1}]_{i'}^0, 0 \leqslant i \leqslant 1$.

(14) $[_{i'} t_2]_{i'}^0 \to [_{i'}]_{i'}^0 t_3$.

(15) $t_3[_{i'}]_{i'}^0 \to [_{i'} t_4]_{i'}^+$.

(16) $[_{i'} t_4 \to t_5]_{i'}^+$.

(17) $[_{i'}t_5 \to t_0]^-_{i'}$.

The objects t_i have the similar role as the objects p_i. They make the rules of types (18) and (19) wait four steps, and then only the rules of types (18) and (19) are used in turn in each membrane i' ($i' \in \{1', 2', \cdots, n'\}$). When the rules of types (1), (2), (3), (4), and (5) are used in turn in membrane i, the rules of types (13), (14), (15) can be used simultaneously. Thus, after these four steps, the polarization of membrane i' is translated from 0 to + and t_0 becomes t_4.

(18) $s[_{i'}\ ]^+_{i'} \to [_{i'}\lambda]^-_{i'},\ s \in \Omega, i' \in \{1', 2', \cdots, n'\}$.

In this rule, the priority is given as follows: for any $s_1, s_2 \in \Omega$, if $f(s_1) > f(s_2)$, the corresponding rule $s_1[_{i'}\ ]^+_{i'} \to [_{i'}\lambda]^-_{i'}$ has a higher priority than the rule $s_2[_{i'}\ ]^+_{i'} \to [_{i'}\lambda]^-_{i'}$, where f is the fitness function.

(19) $s[_{i'}\ ]^-_{i'} \to [_{i'}\lambda]^0_{i'},\ s \in \Omega, i' \in \{1', 2', \cdots, n'\}$.

Similarly, there are priority relationships as follows: for any $s_1, s_2 \in \Omega$, if $f(s_1) > f(s_2)$, the corresponding rule $s_1[_{i'}\ ]^-_{i'} \to [_{i'}\lambda]^0_{i'}$ has a higher priority than the rule $s_2[_{i'}\ ]^-_{i'} \to [_{i'}\lambda]^0_{i'}$, where f is the fitness function.

The aim of the rules of types (18) and (19) is to delete the two worst DNA sequences in membrane i at that moment. After the polarization of membrane i' becomes +, only the rules of type (18) can be used together with the rule of type (16), and at that moment there is only a rule of type (10) being used in membrane i. Therefore, the worst DNA sequence in membrane i is deleted. At the same time, the polarization of membrane i' is translated from + to $-$, and t_4 becomes t_5. In the next step, in membrane i', only the rules of types (17) and (19) can be used, and in membrane i, the rule of type (11) can be used. Therefore, the second worst DNA sequence is deleted in membrane i, and the polarization of membrane i' becomes 0 again and t_5 becomes t_0 again. At the same time, the polarization of membrane i is 0 and simultaneously p_5 evolves to p_0.

From the previous explanation of the rules, we can easily see how this P system works. Therefore, the results of a computation with respect to this P system are all strings over $\{A, G, C, T\}$ collected in membrane n, at the moment of the subscript i of d_i adding up to a given value m from 0. Using this P system, the DNA sequence design problem can be solved and two better DNA sequence can be obtained in membrane n. The basic pseudocode of the membrane evolutionary algorithm is shown in Figure 2.

The procedure of the membrane evolutionary algorithm is shown as follows:

Step 1: Specify membrane structure $[_0[^0_1[^0_{1'}]^0_{1'}[^0_{2'}[^0_{2'}$ $\cdots[^0_n[^0_{n'}]^0_{n'}]^0_n\cdots]^0_2]^0_1]^0_0$ with n regions contained in the skin membrane; generate the initial strings in each region;

Step 2: In each elementary membrane, the mutation rule and crossover rule are implemented simultaneously, and strings are evaluated by a fitness function;

```
Begin
  Initialize membrane structure and parameters.
  While (not termination condition) do
    Execute mutation rule and crossover rule.
    Evaluate the fitness by fitness function.
    Execute communication rule.
    Update every region, delete worst strings.
    Record the current best solution.
  End
End
```

Figure 2 Pseudocode algorithm for our membrane evolutionary algorithm.

Step 3: Communication rules are used to exchange some information among the n regions or between each region; the best and worst strings are sent to the adjacent inner and outer regions, respectively;

Step 4: Update each region simultaneously; delete the worst strings, and save the current best strings;

Step 5: If the stopping condition is satisfied, then output the results; otherwise, return to step (2).

3 Simulation results

3.1 Algorithm parameters

In the simulation, the bases A, C, G and T are mapped to 0, 1, 2 and 3, respectively. m n-mer DNA sequences are connected one by one in the same direction to form an m*n-mer DNA sequences. We denote it as a string in the membrane system.

The novel algorithm based on the new membrane system constructed above for DNA sequence design is implemented with Matlab 7.0. The parameters of the algorithm used in our example are: the number of membranes is 20, the maximum iteration number is 1000, the probabilities of crossover and mutation rate are 0.7 and 0.03, respectively, the threshold value t of continuity is 2, and salt concentration is 0.1 mol/L. For hairpins, we assumed that hairpin formation requires at least six-base-pairings and a six-base loop.

3.2 Results and analyses

First, we compared the new algorithm with the multi-objective evolutionary algorithm from [38]. In [38], Shin et al. proposed a constrained multi-objective evolutionary algorithm to solve DNA sequences optimization for reliable DNA computing. Table 1 presents the sequences for Adleman's Hamilton problem in [38] and the sequences generated by our algorithm. The comparison results in terms of averages of fitness are shown in Figure 3.

From Table 1 and Figure 3, it is clear that our proposed

Table 1 Comparison of the sequences in the multi-objective evolutionary algorithm [38] and our sequences

DNA sequences (5′→3′)	Continuity	Hairpin	H-measure	Similarity	*Tm*	GC (%)
Our sequences						
GCCGGAGCCTTCTTGATAAT	0	0	68	53	49.7408	50
AATCCTGCTTGTCCTCCTAC	0	0	63	50	48.5503	50
TGAGCTCTCTGTTCCAACGA	0	0	64	52	50.6557	50
ATGTAACACGCGGCCACTAA	0	0	63	50	52.0768	50
ACTCGGATTGTGTTGAACGC	0	0	71	51	51.3381	50
CGTTGTTGGCACCTACGTTA	0	0	68	54	50.6851	50
ATCCAGACTACCAAGGCCAA	0	0	61	48	50.0914	50
MOEA algorithm						
CTCTTCATCCACCTCTTCTC	0	0	43	58	46.6803	50
CTCTCATCTCTCCGTTCTTC	0	0	37	58	46.9393	50
TATCCTGTGGTGTCCTTCCT	0	0	45	57	49.1066	50
ATTCTGTTCCGTTGCGTGTC	0	0	52	56	51.1380	50
TCTCTTACGTTGGTTGGCTG	0	0	51	53	49.9252	50
GTATTCCAAGCGTCCGTGTT	0	0	55	49	50.7224	50
AAACCTCCACCAACACACCA	9	0	55	43	51.4735	50

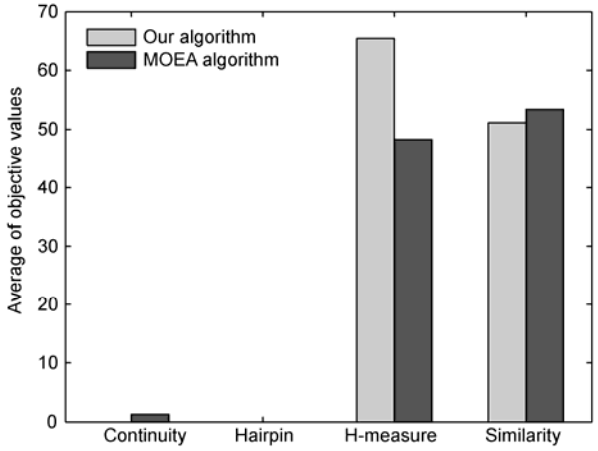

Figure 3 Comparison between the multi-objective evolutionary algorithm [38] and our membrane evolutionary algorithm.

algorithm performs better than the multi-objective evolutionary algorithm according to the average of fitness values (Continuity, Similarity), except for the H-measure. Our algorithm performed the same with regard to Hairpin and GC content fitness. Furthermore, the range of melting temperatures (from 48.5503 to 52.0768) for our algorithm is better than that in [38] (from 46.6803 to 51.4735).

Then, our algorithm was compared with the hybrid quantum chaotic swarm evolutionary algorithm [40]. In [40], Xiao et al. developed a quantum chaotic swarm evolutionary algorithm to select good DNA sequences. DNA sequences and corresponding fitness values, including Similarity, H-measure, Continuity and GC content as listed in Table 2.

To evaluate the performance of the algorithms, the averages of objective values from Table 2 were calculated and

Table 2 Comparison of the sequences in the hybrid quantum chaotic swarm evolutionary algorithm [40] and our sequences

DNA sequences (5′→3′)	Continuity	Hairpin	H-measure	Similarity	*Tm*	GC(%)
Our sequences						
TCTCTACGCCCACGCCCCAT	25	0	50	56	57.4337	65
TTGTGGAGTCCTGAGGTTAG	0	0	68	48	48.1325	60
GGTGTCGGGTGCACTAGGAG	9	0	65	46	54.2526	65
ACTCCAAGTACTCACCGCCT	0	0	62	58	52.3851	55
TACCAACGCAAATCAAAGAC	18	0	60	49	46.7491	40
TTTCTGTCCCTGATCAACTT	18	0	57	52	46.0839	40
ATGTCTCCGCCTTCTTCTCG	0	0	58	57	51.6151	45
QCSEA algorithm						
CCATCTGCTTCACCGATTTA	9	3	65	51	47.6345	45
AGTGCAGTACCGAGAATATT	0	0	67	51	45.8979	40
ATTGAGCGCCCGGACTTCTC	9	0	64	56	54.5984	60
GATTGCGAGAAGGTGTGGAT	0	0	58	55	50.0279	50
GGGTGTAGAGTAGTCTCAGA	9	0	63	58	46.7121	50
CGTGTTCCTATTCCTTGTCC	0	0	57	54	48.0176	50
TAGTCTCTAACTCGGTTGTC	0	0	62	55	45.7525	45

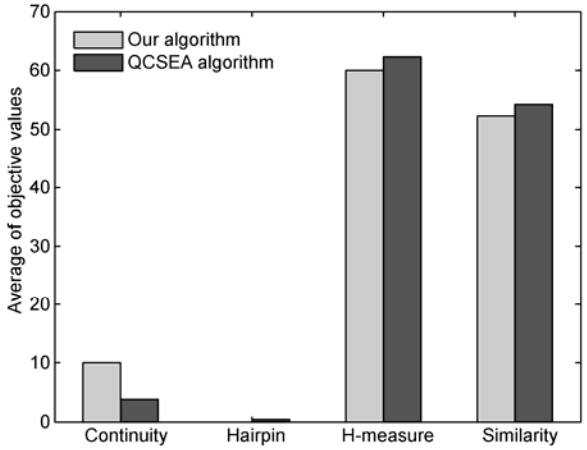

Figure 4 Comparison between the quantum chaotic swarm evolutionary algorithm and our membrane evolutionary algorithm.

shown in Figure 4. From Figure 4, it can be seen that the DNA sequences generated by our membrane evolutionary algorithm performed better than the DNA sequences from [40], according to three criteria (Hairpin, H-measure, and Similarity), but not for Continuity.

4 Conclusions

In this paper, we propose a membrane evolutionary algorithm for solving the DNA sequences optimization problem, and apply it to produce good DNA sequences for DNA computing. The simulation results show that our algorithm is efficient in generating a set of high quality DNA sequences. Although this novel algorithm, based on membrane computing, for DNA sequence design looks simplistic, it has many advantages, such as simplicity, fast convergence, and theoretical elegance. The algorithm deserves to be further investigated, and can be modified to solve other hard optimization problems.

The DNA sequence design problem is important in DNA computing and biology. Further research will focus on more accurate model formulations, and the development of efficient algorithms based on dynamic P systems.

This work was supported by the National Natural Science Foundation of China (*60903105 and 61003038*)*, the 2008 Program Project of Humanity and Social Science of Nankai University* (*NKQ08058*)*, the Opening Foundation of the Key Laboratory of the University of Science and Technology of China for High-Performance Computing and Applications* (*NHPCC-KF-1102*)*, and the Scientific Research Foundation for Doctor of Anhui University* (*02203104*).

1 Paun G H. Computing with membranes. Technical Report. Finland: Turku Center for Computer Science, 1998
2 Pan L Q, Paun G. Spiking neural P systems: An improved normal form. Theor Comput Sci, 2010, 411: 906–918
3 Wang J, Hoogeboom H J, Pan L, et al. Spiking neural P systems with weights. Neural Comput, 2010, 22: 2615–2646
4 Pan L Q, Zeng X X, Zhang X Y. Time-free spiking neural P systems. Neural Comput, 2011, 23: 1–23
5 Alhazov A, Martin-Vide C, Pan L Q. Solving a PSPACE-complete problem by Prelognizing systems with restricted active membranes. Fund Inform, 2003, 58: 67–77
6 Pan L Q, Martin-Vide C. Further remark on P systems with active membranes and two polarizations. J Parallel Distr Com, 2006, 66: 867–872
7 Pan L Q, Martin-Vide C. Solving multidimensional 0-1 knapsack problem by P systems with input and active membranes. J Parallel Distr Com, 2005, 65: 1578–1584
8 Pan L Q, Alhazov A. Solving HPP and SAT by P systems with active membranes and separation rules. Acta Inform, 2006, 43: 131–145
9 Pan L Q, Perez-Jimenez M J. Computational complexity of tissue-like P systems. J Complexity, 2010, 26: 296–315
10 Srinivas M, Patnaik L M. Genetic algorithms: A survey. Computer, 1994, 27: 17–26
11 Back T, Hammel U, Schwefel H P. Evolutionary computation: Comments on the history and current state. IEEE T Evolut Comput, 1997, 1: 3–17
12 Paun G H. Further twenty six open problems in membrane computing. In: Gutierrez-Naranjo M A, Riscos-Nunez A, Romero-Campero F J, eds. Proceedings of the 3rd Brainstorming Week on Membrane Computing, 2005 January 31—February 4. Sevilla: Fenix Editora, 2005. 249–262
13 Nishida T Y. Membrane algorithms: Approximate algorithms for NP-complete optimization problems. In: Ciobanu G, Paun G, Perez-Jimenez M J, eds. Applications of Membrane Computing. Berlin: Springer-Verlag, 2006. 303–314
14 Nishida T Y. Membrane algorithms. LNCS, 2006, 3850: 55–56
15 Leporati A, Pagani D. A membrane algorithm for the min storing problem. LNCS, 2006, 4361: 443–462
16 Zhang G X, Gheorghe M, Wu C Z. A quantum-inspired evolutionary algorithm based on P systems for knapsack problem. Fund Inform, 2008, 87: 93–116
17 Zhang G X, Liu C X, Rong H N. Analyzing radar emitter signals with membrane algorithms. Math Comput Model, 2010, 52: 1997–2010
18 Zaharie D, Ciobanu G. Distributed evolutionary algorithms inspired by membranes in solving continuous optimization problems. LNCS, 2006, 4361: 536–553
19 Adleman L M. Molecular computation of solutions to combinatorial problems. Science, 1994, 266: l021–1024
20 Lipton R. DNA solution of hard computational problems. Science, 1995, 268: 542–545
21 Braich R S, Chelyapov N, Johnson C, et al. Solution of a 20-variable 3-SAT problem on a DNA computer. Science, 2002, 296: 499–502
22 Zimmermann K. Efficient DNA sticker algorithms for NP-complete graph problems. Comput Phys Commun, 2002, 144: 297–309
23 Ouyang Q, Kaplan P D, Liu S, et al. DNA solution of the maximal clique problem. Science, 1997, 278: 446–449
24 Zhang C, Yang J, Xu J, et al. A DNA length reducing computing model for maximum independent set problem. Chin Sci Bull, 2010, 55: 890–896
25 Garzon M, Deaton R, Neathery P, et al. On the encoding problem for DNA computing. In: Rubin H, Wood D H, eds. Proceedings of 3rd DIMACS Workshop on DNA Based Computers, 1997 June 23–27, Philadelphia. Providence: American Mathematical Society, 1997. 230–237
26 Sager J, Stefanovic D. Designing nucleotide sequences for computation: A survey of constraints. LNCS, 2006, 3892: 275–289
27 Marathe A, Condon A E, Corn R M. On combinatorial DNA word design. J Comput Biol, 2001, 8: 201–219
28 Frutos A G, Thiel A J, Condon A E, et al. DNA computing at surfaces: Four base mismatch word designs. In: Rubin H, Wood D H, eds. Proceedings of 3rd DIMACS Workshop on DNA Based Computers, 1997 June 23–27, Philadelphia. Providence: American Mathematical Society, 1997. 238
29 Arita M, Nishikawa A, Hagiya M, et al. Improving sequence design for DNA computing. In: Whitley L D, Goldberg D E, Cantu-Paz E,

eds. Proceedings of Genetic and Evolutionary Computation, 2000 July 8–12, Las Vegas. San Fransisco: Morgan Kaufmann, 2000. 875–882
30 Tanaka F, Nakatsugawa M, Yamamoto M, et al. Developing support system for sequence design in DNA computing. LNCS, 2002, 2340: 129–137
31 Cui G Z, Niu Y Y, Wang Y F, et al. A new approach based on PSO algorithm to find good computational encoding sequences. Prog Nat Sci, 2007, 17: 712–716
32 Wang W, Zheng X D, Zhang Q, et al. The optimization of DNA encodings based on GA/SA algorithm. Prog Nat Sci, 2007, 17: 739–744
33 Qiu Q R, Mukre P, Bishop M, et al. Hardware acceleration for thermodynamic constrained DNA code generation. LNCS, 2008, 4848: 201–210
34 Dyachkov A G, Macula A, Rykov V, et al. DNA codes based on stem similarities between DNA sequences. LNCS, 2008, 4848: 146–151
35 Kawashimo S, Ono H, Sadakane K, et al. Dynamic neighborhood searches for thermodynamically designing DNA sequence. LNCS, 2008, 4848: 130–139
36 Zhang X C, Wang Y F, Cui G Z, et al. Application of a novel IWO to the design of encoding sequences for DNA computing. Comput Math Appl, 2009, 57: 2001–2008
37 Wetmur J G. DNA probes: Applications of the principles of nucleic acid hybridization. Crit Rev Biochem Mol, 1991, 26: 227–259
38 Shin S Y, Lee I H, Kim D, et al. Multi-objective evolutionary optimization of DNA sequences for reliable DNA computing. IEEE T Evolut Comput, 2005, 9: 143–158
39 Meng W C, Qiu J J, Zhang Y H. Immune chaotic algorithm for constrained optimization problems (in Chinese). J Zhejiang Univ, 2007, 41: 299–303
40 Xiao J H, Xu J, Chen Z H, et al. A hybrid quantum chaotic swarm evolutionary algorithm for DNA encoding. Comput Math Appl, 2009, 57: 1949–1958