MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **70** (2013) 987-1004

ISSN 0340 - 6253

A Dynamic Membrane Evolutionary Algorithm for Solving DNA Sequences Design with Minimum Free Energy*

Jian-hua Xiao[1]j, Yun Jiang[2], Juan-juan He[3], Zhen Cheng[4]j

[1] *Logistics Research Center, Nankai University, Tianjin 300071, China*

[2] *School of Computer Science and Information Engineering, Chongqing Technology and Business University, Chongqing 400067, China*

[3] *Key Laboratory of Image Processing and Intelligent Control, Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China*

[4] *College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China*

(Received January 13, 2013)

Abstract

In DNA computation, the DNA encoding is a key problem, and its quantity and quality directly affect the computing efficiency and solution extractions. In recent years, the DNA sequences design has been one of the most practical and important research topics in DNA computing. However, DNA sequences design should simultaneously satisfy various combinational and thermodynamic constraints, which has been proved to be NP-hard problem. In the paper, a dynamic membrane evolutionary algorithm (DMEA) is proposed to solve the DNA sequences design. The method combines the fusion and division rules of P systems with active membranes and ADE/PSO search strategy. The results of simulation experiments show that the proposed algorithm is valid and outperforms other evolutionary algorithms.

*Supported by NSF of China (Grant Nos. 60903105, 60974112, 61100055 and 61202204), Natural Science Foundation Project of CQ CSTC (No. cstc2012jjA40059), and the Fundamental Research Funds for the Central Universities (NKZXB1110).

j Corresponding author. (email: jhxiao@nankai.edu.cn, chengzhen0716@163.com)

1. Introduction

DNA computing is a new computation paradigm with DNA molecules and enzymes working as carrier, and biochemistry trials as information processing instruments. In recent years, DNA computing has been extensively used to solve various NP-complete and intractable problems, such as Hamiltonian path problem [1], the satisfaction problem (SAT) [2], traveling salesman problem (TSP) [3], maximal clique problem [4], and Ramsey number problem [5]. In DNA computing, since the information of the problem to be solved needs to map into DNA sequences, the design of DNA sequences is important to successful DNA computing. A set of good DNA sequences must satisfy many combinatorial and thermodynamic constraints, and can prevent unwanted hybridization errors during the computation, and enable easy retrieval of the answer in the extraction phase. However, DNA sequences design is not an easy task since it is a NP-hard problem, and is difficult to be solved by the traditional optimization methods.

In recent ten years, various kinds of methods and strategies have been proposed to solve the DNA sequences design problem. Arita and Kobayashi [6] proposed a template-map strategy to select a huge number of dissimilar sequences by using only a significantly small number of templates and maps. Penchovsky and Ackermann [7] designed DNA sequences by a random search algorithm. Shin et al. [8] proposed a constrained multi-objective evolutionary algorithm to solve DNA sequences optimization for reliable DNA computing. Khalid et al. [9] used the continuous particle swarm optimization to solve the DNA sequence design. Wang et al. [10] developed GA/SA algorithm for DNA sequences design. Qiu et al. [11] designed a hardware microprocessor to discover the DNA code under the thermodynamic constraints. Zhang et al. [12] proposed a taboo search algorithm to design a set of DNA sequences. Kawashimo et al. [13] presented a local search algorithm for designing sets of short DNA sequences to satisfy thermodynamic constraints with minimum free energy criteria. Zhang et al. [14] proposed an invasive weed optimization algorithm to optimize encoding sequences. Xiao et al. [15] proposed quantum chaotic swarm evolutionary algorithm to select good DNA sequences. Zhang et al. [16-18] proposed an improved dynamic genetic algorithm to solve various DNA sequences design problems based on minimum free energy or H-distance. The improved dynamic genetic algorithm could conquer the shortages and enhance the global search capability of traditional genetic algorithm based on the characteristic of DNA word set design. Xiao et al. [19] developed a membrane evolutionary algorithm based on crossover and mutation rules to optimize DNA sequences design.

Initiated by Gheorghe Paun [20] in 1998, membrane computing is a branch of natural computing dealing distributed parallel computing devices of a biochemical type. Membrane

computing aims to abstract computing ideas and models from the structure and functioning of living cells, as well as from the interactions of living cells in tissues or higher order biological structures. These computing ideas and models from the biology have been turned out to be useful for the purpose of computing. Since Gheorghe Paun introduced it, the area of membrane computing has developed rapidly, and it also has turned out that membrane computing has significant potential to be applied to various hard problems, such as PSPACE-complete problem [21], 0-1 knapsack problem [22], Tripartite matching problem [23], Hamilton path problem [24], and Maximum clique problem [25]. In recent years, the nature-inspired algorithms based on membrane computing have been used to solve complex problems. Nishida [26, 27] firstly developed a membrane algorithm with nested membrane structure to solve the traveling salesman problem. After Nishida, Leporati [28] also used the nested membrane evolutionary algorithm to optimize the minimum storage problem. Huang et al. [29] proposed a membrane algorithm based on the conventional genetic algorithm to solve multi-objective numerical optimization problems. Zhang et al. [30] proposed a membrane algorithm combining one-level membrane structure with binary-observation quantum-inspired evolutionary algorithm to solve the knapsack problem. In [31], a quantum membrane evolutionary based on the real-observation was proposed to solve the numerical optimization problems. In [32], Liu et al. proposed a hybrid membrane algorithm combining P systems with active membranes and real-observation quantum-inspired evolutionary algorithm to solve the time-frequency atom decomposition. In the paper, a dynamic membrane evolutionary algorithm based on DE/PSO is proposed to deal with DNA sequences design problem.

2. The DNA sequence Design

In DNA computing, good DNA sequences should have good chemical properties, and can avoid non-cross hybridized with others. The goal of the DNA sequence design optimization is to select a set of DNA sequences with equal-length n, and each sequence can satisfy certain combinatorial and thermodynamic constraints. In published papers, there are various kinds of criteria to constraint the set of sequences, such as *Continuity*, *Similarity*, *H_measure*, and so on. In the paper, the objective functions and constraints from [19] are used, where the objective functions *H_measure* and *Similarity* are chosen to estimate the uniqueness of each DNA sequence; *Hairpin* and *Continuity* are used to avoid the secondary structure of DNA sequence; *GC* content and melting temperature are used to maintain uniform chemical characteristics. Furthermore, in the paper, we also will introduce the minimum free energy to constrain the thermodynamic stability of DNA sequence.

2.1 Objective Functions

(1) *Continuity*

Continuity calculates the number of same bases in a single-stranded DNA. If there are same bases occurring continuously in a sequence, it will weaken the strands stability. The formulations are defined as follows [8]:

$$F_{Con}(\Sigma)=\sum_{i=1}^{m}\sum_{j=1}^{n-t+1}\sum_{\alpha\in\{A,T,C,G\}}T(C_{\alpha}(x_i,j),t)^2 \tag{1}$$

$$C_{\alpha}(x_i,j)=\begin{cases}c, & \text{if there is } c \text{ such that } x_i\neq\alpha, x_i^{j+k}=\alpha,\\ & \text{for } 1\le k\le c, x_i^{j+k+1}\neq\alpha\\ 0, & \text{otherwise}\end{cases} \tag{2}$$

where x_i $(1\le i\le m)$ denotes the DNA sequences with length n, m is the cardinality of a set of DNA sequences; x_i^j is the j-th base in DNA sequence x_i, and t is the target. If $i>j$, $T(i,j)$ is i, otherwise is 0.

(2) *Hairpin Structure*

Hairpin structure calculates the probability of a single-stranded DNA to form a secondary structure. Hairpin evaluation function is shown as follows [8]:

$$F_{\text{Hairpin}}(\Sigma)=\sum_{i=1}^{m}\sum_{r}^{(n-2*pinlen)}\sum_{c=pinlen+\lceil r/2\rceil}^{(n-pinlen-\lfloor r/2\rfloor)}Hairpin(x_i,c) \tag{3}$$

where r is the minimum length to form hairpin ring, *pinlen* denotes the minimum length of the stem. A hairpin structure is formed at position c for the sequence x_i. $Hairpin(x_i,c)$ is 1, when reverse-complement distance of two sequence which sequence x_i is folded around the c-th base is more than *pinlen*/2, otherwise is 0.

(3) *Similarity*

To keep each sequence as unique as possible, *Similarity* is used to measures the similarity of two given sequences in the same direction. The evaluation function $F_{Similarity}(\Sigma)$ of the similarity measure is defined by Eq. (4).

$$F_{Similarity}(\Sigma)=\sum_{i=1}^{i=m}\sum_{\substack{1\le j\le m\\ j\neq i}}\max_{0\le g\le n}\max_{0\le k\le n+g-1}S(x_i(-)^g x_i,\sigma^k(x_j)) \tag{4}$$

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, please refer to [8].

(4) *H-measure*

H-measure calculates how many nucleotides are complementary to prevent cross-hybridization of two sequences including position shift. The evaluation function is defined as follows.

$$F_{H-measure}(\Sigma)=\sum_{i=1}^{i=m}\sum_{\substack{1\le j\le m\\ j\neq i}}\max_{0\le g\le n}\max_{0\le k\le n+g-1}C(x_i(-)^g x_i,\sigma^k(x_j^R)) \tag{5}$$

where x_j^R is the reverse sequences of sequence x_i, $\sigma^k(x_j^R)$ is the number of corresponding places where two nucleotides are the same.

(5) *Melting Temperatures*

Melting temperature is the temperature at which half of a double stranded DNA starts to break into its single stranded form. In the paper, the nearest neighbor mode [37] is used to calculate melting temperature, and the evaluation function $F_{Tm}(\Sigma)$ is defined as follows.

$$F_{Tm}(\Sigma)=\sum_{i=1}^{m}\left(\frac{\Delta H^{\circ}(x_i)}{\Delta S^{\circ}(x_i)+R\ln(C_T/4)}-273.15\right) \tag{6}$$

where $\Delta H^{\circ}(x_i)$ is the enthalpy of the generated sequence x_i, $\Delta S^{\circ}(x_i)$ is the entropy of the generated sequence x_i, $Tm_{tar}(x_i)$ is the target melting temperature, R is the gas constant; C_T is salt concentration.

2.2 Constraints

(1) *GC Content*

The *GC* content is the percentage of G base and C base in a DNA sequence. It is an important criterion for keeping the uniform chemical properties of DNA sequences. The formulation of *GC* is shown in Eq. (7).

$$GC(x_i)=\frac{\#G+\#C}{|x_i|} \tag{7}$$

where $\#C$, $\#G$ are the amount of C and G in sequences, respectively, and $|x_i|$ is the amount of bases for DNA sequence x_i.

(2) *Minimum Free Energy*

It is known that the secondary structure is more stable for DNA sequence with small free energy. The minimum free energy is the minimum value among free energies of all possible secondary structures of a sequence. In the paper, we will use the nearest-neighbor model [37] to calculate the minimum free energy. The formulation of GC is shown as follows.

$$\triangle G^o(x_i) = \sum_j n_j \triangle G^o(j) + \triangle G^o(init\ w/term\ G \cdot C) + \triangle G^o(init\ w/term\ A \cdot T) + \triangle G^o(sym) \quad (8)$$

where $\triangle G^o(j)$ is the standard free energy changes for the 10 possible Watson-Crick nearest-neighbor. n_j is the number occurrences of each nearest neighbor j, and $\triangle G^o(sym)$ is 0.43 kcal/mol if the duplex is self-complementary, otherwise, it is zero.

2.3 Fitness Function

DNA sequences design optimization is a multi-objective optimization problem with constraints. In the paper, we formulate the objective function as a minimum problem, and use the weight-sum approach to deal with each objective function. The fitness function can be described as follows.

$$\begin{aligned} &\text{Fitness}(\Sigma) = \sum_i w_i F_i(\Sigma) \\ &S.t. \\ &\quad GC_{\min} \le GC(x_j) \le GC_{\max} \quad 1 \le j \le m \\ &\quad \triangle G^o(x_j) \le \triangle G^o_{\max} \quad 1 \le j \le m \end{aligned} \quad (9)$$

where $i \in \{Similarity, H-measure, Continuity, Tm, Hairpin\}$; $GC_{\max}$, $GC_{\min}$ are the maximum and minimum value of GC content, respectively; $\triangle G^o_{\max}$ is the given maximum free energy; w_i is the weight of for each objective. For simplicity, we set each weight to be one.

3. A Dynamic Membrane Evolutionary Algorithm for DNA Sequences Design

P systems with active membranes are a very hot research topic in membrane computing, and the corresponding membrane algorithms have been used widely to solve various optimization problems [32, 33]. In the paper, the structure of the dynamic membrane algorithm with the fusion and division rules is shown in Figure 1.

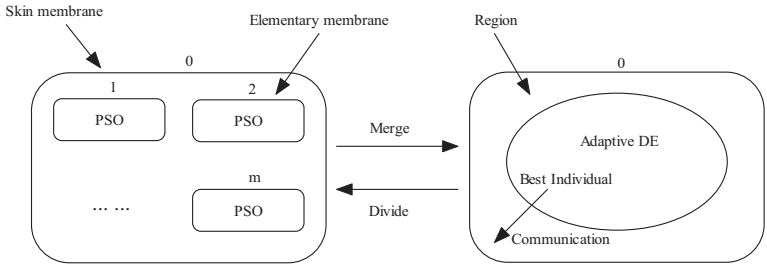

Figure 1. The structure of dynamic membrane algorithm

In this structure, the elementary membranes $1, 2, \cdots, m$ are embedded in the skin membrane 0, and contain a multiset of objects and a set of evolutionary rules and communication rules. In the computing process, all elementary membranes may be merged into one membrane. Moreover, the merging membrane also may be divided into the elementary membranes $1, 2, \cdots, m$. For more details about P systems with active membranes, please refer to [32].

The procedure of improved membrane evolutionary algorithm is described as follows.

Step 1: Specify one level membrane structure $[_0 [_1]_1, [_2]_2, \cdots, [_m]_m]_0$ with m regions contained in the skin membrane denoted by 0 is constructed, and is shown in Figure 1. Randomly generate N_s initial string object in each elementary membrane.

Step 2: In region from 1 to m, the particle swarm optimization (PSO) based on Gaussian distribution will be implemented simultaneously. The flow chart of the particle swarm optimization is shown in Figure 2.

In PSO, proper parameters of the acceleration constants C_1 and C_2 are crucial to enhance the search ability of PSO during the optimization process. However, since the different optimization problem has different values for the acceleration constants, it is not an easy task to select the optimal values. In the paper, Gaussian probability distribution [38] is introduced to generate the accelerating coefficients of PSO. The equation of basic PSO is modified as follows.

$$v_{i,j}^{k+1} = |Randn()| \times (pbest_{i,j}^k - x_{i,j}^k) + |randn()| \times (gbest_j^k - x_{i,j}^k) \tag{10}$$

$$x_{i,j}^{k+1} = x_{i,j}^k + v_{i,j}^{k+1} \tag{11}$$

where $Randn()$ and $|randn()|$ are positive random numbers generated by using $abs(N(0,1))$; $pbest_{i,j}^k$ represents the best location in the search space ever visited by particle i, and $gbest_j^k$ is

the best location discovered so far; $v_{i,j}^{k}$ and $x_{i,j}^{k}$ are the velocity and the current position of the j-th dimension in the i-th particle at the k-th iteration.

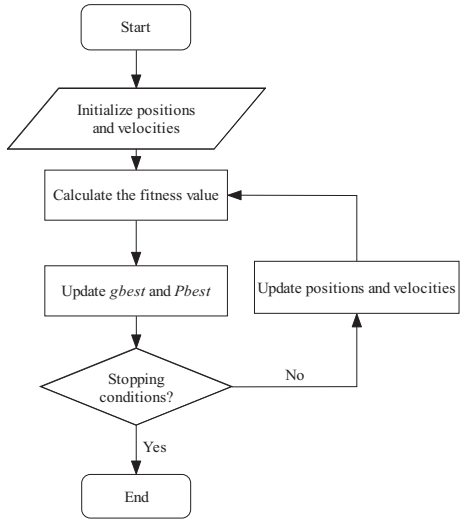

Figure 2. The flow chart of the particle swarm optimization

Step 3: Implement the merging operation, all elementary membranes are merged into one elementary membrane m_{one}, and the strings of all elementary membranes enter the membrane m_{one}.

Step 4: In membrane m_{one}, the adaptive differential evolution (ADE) will be used to update the strings object. The pseudocode of standard DE is shown as follows.

In standard DE, F and CR are constant, but the performance of DE is sensitive to the choice of control parameter F. There are several attempts to dynamically adjust the parameter F in [34, 35], and have achieve good results. In the paper, we will introduce the self-adaptive method to control the parameters CR and F [36] and the equations are modified as follows.

$$CR_i(t) = CR_{i1}(t) + N(0,1) \times (CR_{i2}(t) - CR_{i3}(t)) \tag{12}$$

$$F_i(t) = F_{i4}(t) + N(0,0.5) \times (F_{i5}(t) - F_{i6}(t)) \tag{13}$$

where $i_1, i_2, i_3 \in \{1, 2, \cdots, N\}$ are random and mutually different integers; the parameters F and CR are generated for each variable from a normal distribution. Each individual i has its own CR and F. The parameters CR and F are firstly initialized for each individual in the population from the uniform distribution $U(0,1)$ and the normal distribution $N(0.5, 0.15)$, respectively.

```
Generate initial population P = (x_1, x_2, ··· , x_N) of random individuals
Do
    For each individual i in the population
        Generate three random different integers, r_1, r_2 and r_3 ∈ {1, 2, ··· , N}
        Generate a random integer j_rand ∈ {1, ··· , n}
        For each dimension parameter j
            If rand_j(0, 1) < CR or j = j_rand
                x'_{i,j} = x_{i,r3} + F*(x_{i,r1} - x_{i,r2})
            Else
                x'_{i,j} = x_{i,j}
            End If
        End For
        If Fitness(x'_i) ⩽ Fitness(x_i)
            x_i = x'_i
        End If
    End For
While (The termination condition)
```

where n is the number of objective parameters and N is the population size, $x_{i,j}$ is the j-th decision variable of the i-th individual in the population; $rand()$ stands for the uniform random number in [0, 1], and j_{rand} is a randomly chosen index; $F > 0$ is the scale factor, and $CR \in [0,1]$ is the probability of reproduction.

In the paper, if the variable value $x'_{i,j}$ violates the boundary constraint, the violated variable value is reflected back from the violated boundary using the following rule [39].

$$x'_{i,j} = \begin{cases} x_{\min,j} & \text{if } (rand() \le 0.5) \wedge (x'_{i,j} < x_{\min,j}) \\ x_{\max,j} & \text{if } (rand() \le 0.5) \wedge (x'_{i,j} > x_{\max,j}) \\ 2 \times x_{\min,j} - x'_{i,j} & \text{if } (rand() > 0.5) \wedge (x'_{i,j} < x_{\min,j}) \\ 2 \times x_{\max,j} - x'_{i,j} & \text{if } (rand() > 0.5) \wedge (x'_{i,j} > x_{\max,j}) \end{cases} \tag{14}$$

Step 5: Calculate the fitness of each string object by fitness function.

Step 6: Implement the communication rules, a copy of the best strings selected in the membrane m_{one} is sent to the skin membrane, and the current best strings are saved in the skin membrane.

Step 7: The algorithm checks if the stopping conditions are achieved. If the stopping condition is met, then output results; otherwise go to Step 8.

Step 8: The membrane m_{one} is divided into the same structure with the m elementary membranes, and the currently best strings and N_s-1 strings with the worst fitness will be sent to each elementary membrane in turn by the send-in communication rules, then go back to Step 2.

In the paper, the stopping condition is the maximum number of iterations. The algorithm will stop if the maximum number of iterations is reached, and output the results.

4. Simulation Results

4.1 Algorithm Parameters

The dynamic membrane evolutionary algorithm based on ADE/PSO for solving DNA sequence design is executed with Matlab 7.0. The parameters of the algorithm used in our example are shown in Table 1. For hairpins, we assumed that hairpin formation requires at least six base-pairings and a six base loop.

Table 1. Parameters used in our algorithm

Parameters	Initial Value	Meaning
$iter_{\max}$	1000	The maximum number of iterations
m	20	The number of the elementary membrane
N_s	8	The string size in region from 1 to m
C_T	0.1	The salt concentration
t	2	The threshold value of continuity
$\triangle G^o_{\max}$	-25	The given maximum free energy
R	1.987	The gas constant

4.2 Results and Analyses

In this subsection, the results of the proposed algorithm are compared with existing approaches, taken from conventional evolutionary algorithm (CEA) [8], quantum chaotic swarm evolutionary algorithm (QCSEA) [15], and conventional membrane evolutionary algorithm

(CMEA) [19]. For each comparison, 10 runs have been performed by DMEA algorithm and the best performance is calculated by the fitness function.

First, the DMEA is compared with results given in [8], which were obtained by using the conventional evolutionary algorithm (CEA) with multiple-point crossover, single-point mutation, and roulette wheel selection. In [8], the population size was 1000, maximum generation was 1000, crossover rate was 0.9, and mutation rate was 0.05. Results of the two algorithms are compared in Table 2 and Figure 3. Where the white columns denote the averages calculated from our sequences, and the grey columns denote the averages calculated from CEA.

Table 2. Comparison results of the sequences in CEA algorithm and our sequences

DNA Sequences (5' →3')	Continuity	Hairpin	H-measure	Similarity	T*m*	*GC* (%)
Our Sequences						
TGAGTTGGAACTTGGCGGAA	0	0	70	52	55.4044	50
CAGCATGTTAGCCAGTACGA	0	0	60	55	53.5214	50
TTGAGTCCGCGTGGTTGGTC	0	0	63	53	58.8391	60
AATTGACACTCTGATTCCGC	0	0	68	58	51.9350	45
CATACATTGCATCAACGGCG	0	0	67	53	54.3063	50
ATACACGCACCTAGCCACAC	0	0	59	50	55.9207	55
GTTCCACAACAGGTCTAATG	0	3	61	53	49.7760	45
CEA Sequences						
AGGCGAGTATGGGGTATATC	16	0	66	48	47.6070	50
ATCGTACTCATGGTCCCTAC	9	0	64	54	47.8464	50
CCTGTCAACATTGACGCTCA	0	3	66	57	50.6204	50
CGCTCCATCCTTGATCGTTT	9	0	62	58	50.4628	50
CTTCGCTGCTGATAACCTCA	0	3	68	54	49.8103	50
GAGTTAGATGTCACGTCACG	0	3	67	51	48.3995	50
TTATGATTCCACTGGCGCTC	0	0	61	58	50.1205	50

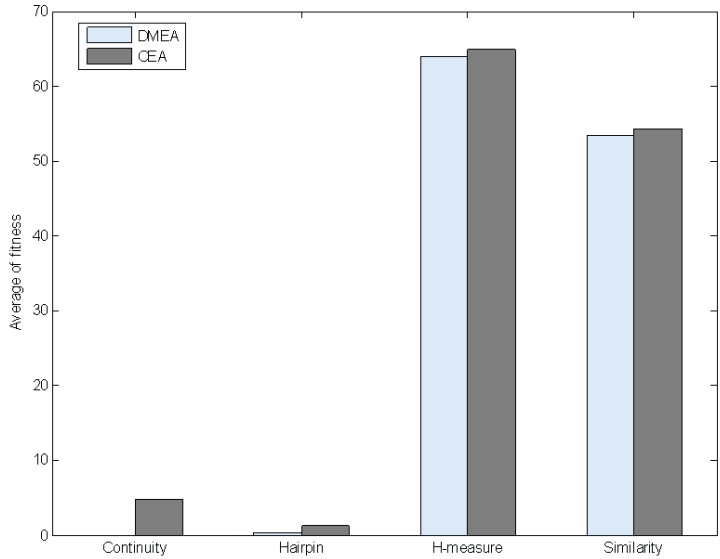

Figure 3. Average objective values comparison between CEA and DMEA

From Figure 3, the sequences generated by our algorithm outperform the sequences designed by CEA algorithm. Our sequences show much lower average values in *Continuity*, *Hairpin*, *H-measure* and *Similarity*. This implies that the sequences made by DMEA have better performance to avoid the secondary structure and reduce the probability to hybridize with the non-complementary sequences.

Then, we compared DMEA with [15]. In [15], a hybrid quantum chaotic swarm evolutionary algorithm (QCSEA) was used to design good sequences. The set of seven DNA sequences whose length is 20-mer and corresponding fitness values are listed in Table 3. The comparison results are shown in Figure 4.

Table 3. Comparison results of the sequences in QCSEA algorithm and our sequences

DNA Sequences (5' →3')	Continuity	Hairpin	H-measure	Similarity	T*m*	*GC* (%)
Our Sequences						
TATACTCTGATATGGTCGTG	0	0	63	52	46.7627	40
AGTGTCAAGTCACGGTGCGT	0	0	65	56	57.7731	55
ACTTCAGGACTCCACGCCTT	0	0	65	56	56.5048	55
CGCGCCGTCATCCTAATGAC	0	0	68	57	57.4978	60
TCCTTCTGGTCCAGGTACAT	0	0	59	57	52.7424	50
GTCCTATCCAACCAACTGCG	0	0	60	55	54.4618	55
GCACTCAAGTTGTAAGCTAG	0	0	66	53	49.9098	45
QCSEA Sequences						
AACAATGAATGGGCAGGAGT	9	3	54	56	52.9306	45
CAGGACTAAACAATTCCAAA	18	3	53	60	46.9346	35
CACATTACGCCAAGGATACC	0	0	54	53	52.2051	50
GACCGCAAGACAGAAGAGAA	0	0	48	61	53.3654	50
ACCGACGTCCGTAACTGACC	0	0	59	54	57.7230	60
ACATGAGATCAACCTGCGCA	0	0	54	56	55.6584	50
TAAGAGAATGCCAGAATAAG	0	0	50	60	45.5851	35

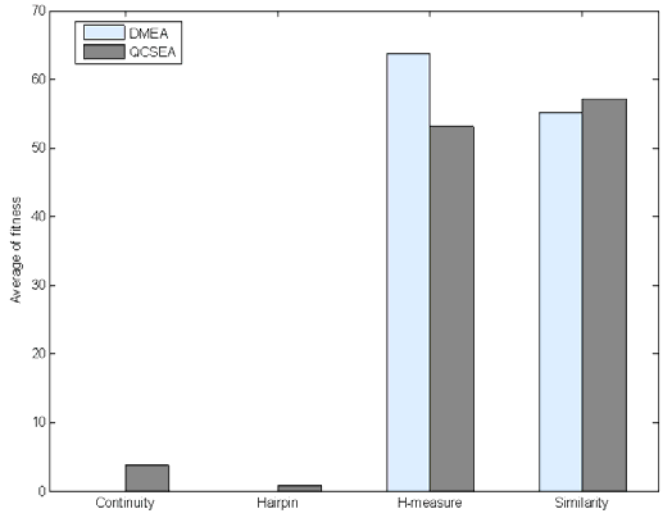

Figure 4. Average objective values comparison between QCSEA and DMEA

From Figure 4, it is found that the sequences designed from QCSEA algorithm achieved a lower *H-measure* fitness level. However, the proposed DMEA performs better than the QCSEA according to the average of fitness values of *Continuity*, *Similarity*, and *Hairpin*. In addition, the range of melting temperatures (from 46.7627 to 57.7731) is better than QCSEA algorithm (from 45.5851 to 57.7230), which has more advantage in the thermodynamic characteristics.

Finally, we compared our algorithm with [19]. In [19], a conventional membrane evolutionary algorithm (CMEA) was proposed to solve the DNA sequences optimization problem. The generated sequences and their evaluated values of two algorithms are listed in Table 4, and the corresponding performance comparison of DMEA algorithm with CMEA method is also shown in Figure 5.

Table 4. Comparison results of the sequences in CMEA algorithm and our sequences

DNA Sequences (5' →3')	Continuity	Hairpin	H-measure	Similarity	T*m*	*GC* (%)
Our Sequences						
ACACCTCCTTCCTTCTAACC	0	0	46	58	51.7253	50
ACTTCTTCCTGAAGTCTGCC	0	0	57	55	52.7729	50
TAACTGGCTATAGTACCGCG	0	0	61	48	52.3525	50
TCGAATGAGCCAACGGAATT	0	0	65	49	53.4295	45
ACTCTCCTTCTCGTCTCTCC	0	0	40	60	53.2088	55
TCCTGTTCTTATCTCTCGCC	0	0	45	58	52.1097	50
CATTGGTCATGTTCCTCACC	0	0	56	56	52.1764	50
CMEA 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

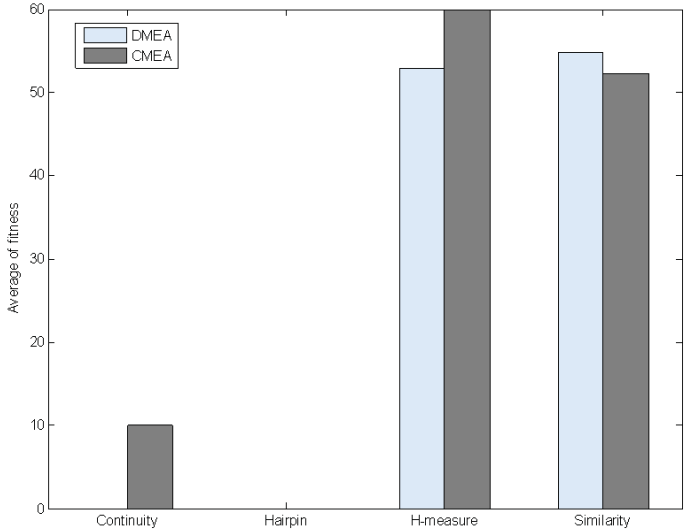

Figure 5. Average objective values comparison between CMEA and DMEA

From Figure 5, our sequences show much lower values in terms of *Continuity* and *H-measure*, except for *Similarity*. Our algorithm has the same performance for *Hairpin*. Moreover, the range of *melting temperatures* (from 51.7253 to 53.4295) is smaller than CMEA (from 46.0839 to 57.4337). This implies the sequences made by DMEA have much better thermodynamic characteristics.

5. Conclusion

The DNA encoding is a key problem, and its quantity and quality directly affect the computing efficiency and solution extractions. In this paper, the dynamic membrane evolutionary algorithm was proposed to design good DNA sequences for reliable DNA computing. The algorithm used particle swarm optimization to update the string object for each elementary membrane, and implemented differential evolution based on self-adaptive method to improve the global search ability. The results of simulation experiments show that the proposed algorithm is valid and outperforms other evolutionary algorithms. However, we have some further works to do. The improved membrane algorithm will be used to solve other optimization hard problems.

References

[1] L. M. Adleman, Molecular computation of solutions to combinatorial problems, *Science* **266** (1994) l021-1024.

[2] R. S. Braich, N. Chelyapov, C. Johnson, P. W. K. Rothemund, L. Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer, *Science* **296** (2002) 499-502.

[3] K. Zimmermann, Efficient DNA sticker algorithms for NP-complete graph problems, *Comput. Phys. Commun.* **144** (2002) 397-309.

[4] Q. Ouyang, P. D. Kaplan, S. Liu, A. Libchaber, DNA solution of the maximal clique problem, *Science* **278** (1997) 446-449.

[5] X. Zhou, K. L. Li, M. Goodman, A. Sallam, A novel approach for the classical Ramsey number problem on DNA-based supercomputing, *MATCH Commun. Math. Comput. Chem.* **66** (2011) 347-370.

[6] M. Arita, S. Kobayashi, DNA sequence design using templates, *New Generation Computing* **20** (2002) 263-277.

[7] R. Penchovsky, J. Ackermann, DNA library design for molecular computation. *J. Comput. Biol.* **10** (2003) 215-229.

[8] S. Y. Shin, I. H. Lee, D. Kim, B. T. Zhang, Multi-objective evolutionary optimization of DNA sequences for reliable DNA computing, *IEEE T. Evolut. Comput.* **9** (2005) 143-158.

[9] N. K. Khalid, Z. Ibrahim, T. B. Kurniawan, DNA sequence optimization based on continuous particle swarm optimization for reliable DNA computing and DNA nanotechnology, *J. Comput. Sci.* **4** (2008) 942-950.

[10] W. Wang, X. D. Zheng, Q. Zhang, J. Xu, The optimization of DNA encodings based on GA/SA algorithm, *Prog. Nat. Sci*. **6** (2007) 739-744.

[11] Q. R. Qiu, P. Mukre, M. Bishop, D. Burns, Q. Wu, Hardware acceleration for thermodynamic constrained DNA code generation, in: M. H. Garzon, H. Yan (Eds.), *DNA Computing*, Springer, Berlin, 2008, pp. 201-210.

[12] K. Zhang, J. Xu, X. T. Geng, J. H. Xiao, L. Q. Pan, Improved taboo search algorithm for designing DNA sequences, *Prog. Nat. Sci.* **18** (2008) 623-627.

[13] S. Kawashimo, H. Ono, K. Sadakane, M. Yamashita, Dynamic neighborhood searches for thermodynamically designing DNA sequence, in: M. H. Garzon, H. Yan (Eds.), *DNA Computing*, Springer, Berlin, 2008, pp. 130-139.

[14] X. C. Zhang, Y. F. Wang, G. Z. Cui, Y. Niu, J. Xu, Application of a novel IWO to the design of encoding sequences for DNA computing, *Comput. Math. Appl.* **57** (2009) 2001-2008.

[15] J. H. Xiao, J. Xu, Z. H. Chen, K. Zhang, L. Q. Pan, A hybrid quantum chaotic swarm evolutionary algorithm for DNA encoding, *Comput. Math. Appl.* **57** (2009) 1949-1958.

[16] Q. Zhang, B. Wang, X. P. Wei, Evaluation the different combinatorial constraints in DNA computing based on minimum free energy, *MATCH Commun. Math. Comput. Chem*. **65** (2011) 291-308.

[17] Q. Zhang, B. Wang, X. P. Wei, X. Y. Fang, C. J. Zhou, DNA word set design based on minimum free energy, *IEEE T. Nanobiosci.* **9** (2010) 273-277.

[18] Q. Zhang, B. Wang, On the bounds of DNA coding with H-distance, *MATCH Commun. Math. Comput. Chem.* **66** (2011) 371-380.

[19] J. H. Xiao, X. Y. Zhang, J. Xu, A membrane evolutionary algorithm for DNA sequence design in DNA computing, *Chinese Sci. Bull.* **57** (2012) 698-706.

[20] G. H. Paun, Computing with membranes, *Technical Report, Finland: Turku Center for Computer Science*, 1998.

[21] A. Alhazov, C. Martin-Vide, L. Q. Pan, Solving a PSPACE-complete problem by recognizing P systems with restricted active membranes, *Fund. Inform.* **58** (2003) 67-77.

[22] L. Q. Pan, C. Martin-Vide, Solving multidimensional 0-1 knapsack problem by P systems with input and active membranes, *J. Parallel Distr. Com.* **65** (2005) 1578-1584.

[23] Y. Y. Niu, L. Q. Pan, M. J. Perez-Jimenez, M. R. Font, A tissue P systems based uniform solution to tripartite matching problem, *Fund. Inform.***109** (2011) 1-10.

[24] L. Q. Pan, A. Alhazov, Solving HPP and SAT by P systems with active membrane and separation rules, *Acta Inform.* **43** (2006) 131-145.

[25] M. Garcia-Arnau, D. Manrique, A. Rodriguez-Paton, P. Sosik, A P system and a constructive membrane-inspired DNA algorithm for solving the maximum clique problem, *Biosystems* **2** (2007) 1-11.

[26] 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, Orlando*, 2004, pp.109-112.

[27] T. Y. Nishida, An approximate algorithm for NP-complete optimization problems exploiting P-systems, *The Brainstorming Workshop on Uncertainty in Membrane Computing, Palma de Mallorca,* 2004, pp.185-192.

[28] A. Leporati, D. Pagani, A membrane algorithm for the min storing problem, in: H. J. Hoogeboom, G. Paun, G. Rozenberg, A. Salomaa (Eds.), *Membrane Computing*, Springer, Berlin, 2006, pp. 443-462.

[29] L. Huang, X. X. He, N. Wang, Y. Xie, P systems based multi-objective optimization algorithm, *Prog. Nat. Sci.* **17** (2007) 458-465.

[30] G. X. Zhang, M. Gheorghe, C. Z. Wu. A quantum-inspired evolutionary algorithm based on P systems for knapsack problem. *Fund. Inform.* **87** (2008) 93-116.

[31] G. X. Zhang, H. N. Rong, Real-observation quantum-inspired evolutionary algorithm for a class of numerical optimization problems, in: Y. Shi, G. D. van Albada, J. Dongara, P. M. A. Sloot, *Computational Science*, Springer, Berlin, 2007, pp. 989-996.

[32] X. X. Liu, G. X. Zhang, H. W. Liu, M. Gheorghe, F. Ipate, An improve membrane algorithm for solving time-frequency atom decomposition, in: G. Paun, M. J. Perez-Jimenez, A. Riscos-Nunez, G. Rozenberg, A. Salomaa (Eds.), *Membrane Computing*, Springer, Berlin, 2010, pp. 371-384.

[33] L. Zhong, W. F. Luo, An improved membrane algorithm for solving time-consuming water quality retrieval, *SPIE* **8005** (2011) 8005091-8005097.

[34] J. Liu, J. Lampinen, A fuzzy adaptive differential revolution algorithm, *Soft. Comput*. **9** (2005) 448-462.

[35] Z. Y. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search. *The 2008 IEEE Congress on Evolutionary Computation, Hong Kong*, 2008, pp.1111-1116.

[36] A. Salman, A. P. Engelbrecht, M. G. H. Omran, Empirical analysis of self-adaptive differential evolution, *Eur. J. Oper. Res.* **183** (2007) 785-804.

[37] L. Santa, A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics, *Proc. Nat. Acad. Sci. USA* **95** (1998) 1460-1465.

[38] R. A. Krohling, L. S. Coelho, Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems, *IEEE Trans.* **36** (2006) 1407-1416.

[39] J. Brest, V. Zumer, M. S. Maucec, Self-adaptive differential evolution algorithm in constrained real-parameter optimization, *The 2006 Congress on Evolutionary Computation, Vancouver*, 2006, pp. 215-222.