Journal of
Computational and Theoretical Nanoscience
Vol. 4, 1257–1262, 2007

An Improved Hopfield Neural Network Algorithm for Computational Codeword Design

Yan-Feng Wang[1, 2, *], Guang-Zhao Cui[1], Xun-Cai Zhang[2], and Bu-Yi Huang[1]

[1] *School of Electrical and Electronic Engineering, Zhengzhou University of Light Industry, 450002 Zhengzhou, Henan, China*
[2] *Research Institute of Biomolecular Computer, Huazhong University of Science and Technology, 430074 Wuhan, Hubei, China*

Designing computational codeword is crucial in DNA computing. However, this is a bothersome task as too many constraints need to be satisfied in terms of definiting of encoding problem. This paper proves that the problem of finding the maximum number of computational codeword in a randomly generated set of DNA sequences is not only NP-hard, but it can also be mapped onto the solution of a graph of maximum clique problem. Thus, utilizing meta-heuristic algorithm to find an optimal or near optimal solution and predestinating whether or not the computational codeword in randomly generated set are required for the following controllable computation. Here we present an improved Hopfield neural network algorithm to solve this problem. The simulation results show that the proposed method is useful for a user to select an appropriate set of candidate DNA sequences to filter and obtain good computational codeword.

Keywords: DNA Computing, Hopfield Neural Network, Codeword Design, Maximum Clique.

1. INTRODUCTION

DNA computing is a bio-inspired computing paradigm, which always used to solve hard combinatorial optimization problem due to its useful properties, such as its massive parallelism and huge memory capacity. Because DNA computing relies heavily on biochemical reactions and is restricted by technological difficulties, it may result in undesirable computations. Therefore, designing codeword for controllable computation is crucial in DNA computing. However, this is a bothersome task as too many constraints need to be satisfied in terms of definiting the encoding problem.

In general, single-stranded DNA sequences that are structure free are always utilized to perform DNA computing. When faced with a large-scale computational problem, an exponentially increasing amount of computational codeword is required. Therefore, the total number of computational codeword, namely, the maximal set of encoding DNA sequences, which can guarantee desirable computations should be surveyed to estimate whether or not it is enough for mapping the instances of an algorithmic problem. To our knowledge there is currently no accredited method to solve this problem. For this reason, an efficient algorithm that can solve the maximum number of computational codeword problem (MNCCP) would be valuable.

*Author to whom correspondence should be addressed.

This paper is organized as follows. Section 2 introduces some necessary backgrounds, such as the encoding problem, the design strategies and criteria of computational codeword; moreover, how to map MNCCP onto the solution of the graph of maximum clique problem (MCP) is presented, too. Since the Hopfield neural network (HNN) has been demonstrated to be effective in solving MCP, and then we also introduce the basic HNN in brief. Section 3 presents the neural representation of MNCCP and proposes an efficient algorithm to solve this problem. Computational results are presented and conclusions are drawn in Sections 4 and 5, respectively.

2. BACKGROUNDS

2.1. The Encoding Problem

The encoding problem of DNA computing can be simply defined as mapping the instances of an algorithmic problem in a systematic manner onto specific DNA molecules such that the certain chemical reactions avoid sources of error, and at the same time, the resulting products contain, with a high degree of reliability, enough DNA molecules to encode the answers to the problem's instances and enable a successful extraction.[1, 2]

In other words, *good computational codeword* should ensure:[3]

(1) certain chemical reactions are specific hybridizations;

(2) the controllable PCR reaction can amplify the resulting products; and
(3) the resulting products are reliable and can be extracted successfully.

2.2. Design Strategies and Criteria

To ensure a chemical reaction is controllable, some constraints have been proposed in terms of definiting the encoding problem, such as similarity, H-measure, secondary structure, continuity, free energy, melting temperature, GC content, polymerase chain reaction (PCR) protocol, and so on.[4] In our method, we only consider three constraints: primary structure, melting temperature and free energy, the detailed reasons for which will be explained in Section 5. Note that as the sequences that satisfy the above three constraints, we define *computational codeword*.

2.2.1. Primary Structure

The primary structure refers to the sequence of bases. To satisfy the definition of the encoding problem, the primary structure of each single-stranded DNA sequence that is generated randomly should have:[5]
(1) No repeat structure in the sequence;
(2) No large GC rich or deficient regions;
(3) No nucleotide stretches; and
(4) No dimer potential.

P_i is denoted as the primary structure properties of DNA sequence v_i, which is defined as

$$P_i = \begin{cases} 1, & \text{if } v_i \in \{(1) \cap (2) \cap (3) \cap (4)\}, \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

2.2.2. Melting Temperature

Melting temperature (T_m) is an important factor for the efficiency of a reaction. The accurate prediction of T_m is particularly critical in the case of PCR. Large errors in the T_m estimation can lead to the amplification of non-specific products or to an inappropriate hybridization performance.

Since oligonucleotides up to about 50 bases in length are always utilized for performing DNA computing, here we consider the method and thermodynamic parameters provided by SantaLucia as having demonstrated that it has a good performance in predicting the experimental T_m of short single-stranded DNA sequences. Please see Ref. [6] for a more detailed discussion of the thermodynamic parameters in T_m calculation.

To reduce the probability of non-specific hybridizations, the melting temperature should be uniform. T_{ij} is denoted as the melting temperature between sequences v_i and v_j, which is defined as

$$T_{ij} = \begin{cases} 1, & \text{if } |T_m(v_i) - T_m(v_j)| \le 5, \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

2.2.3. Free Energy

Free energy is perhaps the most important thermodynamic quantity that relates enthalpy and entropy. Many physical properties of DNA sequences depend directly or indirectly on free energy.

The change in free energy (ΔG) for a chemical reaction indicates whether it will be thermodynamically favourable at a given temperature. An exothermic reaction produces heat ($\Delta G < 0$), while an endothermic reaction requires heat ($\Delta G > 0$). The sign of ΔG determines whether or not the reaction is spontaneous, as well as its direction. Therefore, ΔG is the driving force for any chemical reaction.

For a candidate single-stranded DNA sequence, let ΔG be the free energy released when specific hybridization occurs between any randomly generated single-stranded DNA sequence and its complement. To ensure reliable results, we hope $|\Delta G^{sp}|$ is as large as possible where ΔG^{sp} is defined as

$$\Delta G^{sp} = \begin{cases} 1, & \text{if } |\Delta G^{sp}| \ge 20, \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

2.3. MNCCP Is NP-Hard

Let $G(V, E)$ be an arbitrary undirected graph, where $V = \{v_i \mid i = 1, 2, \ldots, n\}$ is the set of vertices and $E \subseteq V \oplus V$ is the set of edges, namely, $E = \{a_{ij} \mid i, j = 1, 2, \ldots, n\}$. Two distinct vertices v_i and v_j are called adjacent if they are connected by an edge. The adjacent matrix of G is the Boolean matrix $A(a_{ij})$, where $a_{ij} = 1$, if $(v_i, v_j) \in E$ and $a_{ij} = 0$ otherwise. A clique of G is a subset $C \subseteq V$ where every pair of vertices is adjacent. A clique is called maximal if it is not a subset of another clique, and the highest-cardinality maximal clique is called maximum. The maximal clique problem requires finding a maximum clique in a given graph G. Formally,

Max Clique
Instance: Graph $G(V, E)$.
Solution: A clique in G, i.e., a subset $C \subseteq V$ such that every pair of vertices in C is adjacent.
Measure: Cardinality $|C|$ of the clique.

The MCP has been proven to be a NP-hard problem.[7]

Constructing an arbitrary undirected graph G, if let n vertices in G representing n DNA sequences, an edge between two vertices v_i and v_j, represent the corresponding two DNA sequences x_i and x_j that satisfies the above three criteria, and is defined as

$$a_{ij} = T_{ij} \cdot P_i \cdot \Delta G^{sp} = 1 \quad (4)$$

Therefore MNCCP problem can be mapped onto the MCP in graph G.

As the problem is NP-hard, efficient procedures providing high quality solutions in reasonable runtimes are very

useful. At present, most of meta-heuristics algorithms, such as simulated annealing algorithms, genetic algorithms, ant colony optimization algorithms and artificial neural networks algorithms for NP-hard problems work reasonably well on many cases. Although these approaches do not obtain the optimum solutions, we can obtain suboptimum solutions by restricting bounds. For the MNCCP discussed here, we consider it is efficient if the number of computational codeword obtained by using the meta-heuristics algorithms is enough for biological experiments. Since the HNN has been demonstrated to be effective in solving the MCP, we will address the MNCCP by using HNN algorithm in this paper.

2.4. The Basic HNN

Hopfield presented the energy function approach to solve several optimization problems, including the travelling salesman problem (TSP), analog to digital conversion, signal processing problems and linear programming problems.[8] His results have encouraged a number of researchers to apply this network to different problems, such as object recognition, graph recognition and economic dispatch problems.

In general, the Hopfield network with n neurons is represented in the following differential equation.

$$\frac{dx_i(t)}{dt} = \sum_{j=1}^{n} w_{ij} \cdot y_j(t) + I_i(t), \quad i = 1, 2, \ldots, n \tag{5}$$

where x_i and y_i are the input and output of neuron i at time t, respectively; w_{ij} is the weight of the connection from neuron j to neuron i; and I_i is the extra bias current of neuron i.

There are two kinds of modes to update the internal potential x_i of neuron:

(1) Time-independent mode, in which the internal potential of neuron at $t+1$ does not directly depend on its value at t.

$$x_i(t+1) = \frac{dx_i(t)}{dt} \tag{6}$$

(2) Time-dependent mode, in which the internal potential of neuron at $t+1$ depends on its value at time t.

$$x_i(t+1) = x_i(t) + \frac{dx_i(t)}{dt} \tag{7}$$

The neuron state y_i (output) is updated from x_i using a non-linear function called neuron model.

The following two neuron models have been used for optimization problems:

(1) The McCulloch-Pitts neuron model

$$y_i = \begin{cases} 1, & \text{if } x_i > 0, \\ 0, & \text{otherwise} \end{cases} \quad i = 1, 2, \ldots, n \tag{8}$$

(2) The Sigmoid model

$$y_i = \frac{1}{1 + e^{(-x_i/T)}} \tag{9}$$

3. METHODS

3.1. Statement of the Problem

To provide a solution to an undirected graph MCP with n vertices, we construct a Hopfield network with n neurons, such that output $Y = (y_1, y_2, \ldots, y_n)$ is stable, and corresponds to a clique C in graph G. In which case, $Y = (y_1, y_2, \ldots, y_n)$ can be defined as

$$y_i = \begin{cases} 1, & \text{if } v_i \in C, \\ 0, & \text{if } v_i \notin C \end{cases} \quad i = 1, 2, \ldots, n \tag{10}$$

Objective Function. Obviously, the maximum vertices of C in graph G can be denoted as

$$J = -\sum_{i=1}^{n} y_i \to \min \tag{11}$$

Constraint Conditions. The following cases should be considered based on whether or not vertices are in C.[9, 10]

(1) If v_i, $v_j \in C$, namely, $v_i = 1$, $v_j = 1$, then there must exit one edge between v_i and v_j, namely, $a_{ij} = 1$.

(2) If $v_i \in C$; $v_j \notin C$, namely, $v_i = 1$, $v_j = 0$, then there may exit one edge between v_i and v_j or not, namely, $a_{ij} = 1$ or $a_{ij} = 0$.

(3) If v_i, $v_j \notin C$, namely, $v_i = 0$, $v_j = 0$, then there may exit one edge between v_i and v_j or not, namely, $a_{ij} = 1$ or $a_{ij} = 0$.

To summarize the above three cases, we can obtain a stable output Y, and correspond to any one clique C in graph G such that Y satisfies

$$y_i y_j (1 - a_{ij}) = 0 \quad i, j = 1, 2, \ldots, n, \; i \neq j \tag{12}$$

Based on formula (11) and (12), we can then construct the energy function of Hopfield network, represented as follows.

$$E = -A \sum_{i=1}^{n} y_i + B \sum_{i=1}^{n} \sum_{j=1,\, j \neq i}^{n} y_i y_j (1 - a_{ij}) \tag{13}$$

where A and B are the weight coefficients of object function and constraint, respectively, and can be confirmed by experiments.

This energy function can be rewritten into the standard energy function of the Hopfield network:

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1,\, j \neq i}^{n} w_{ij} y_i y_j - \sum_{i=1}^{n} I_i y_i \tag{14}$$

Where the weights and the thresholds of the Hopfield network become:

$$\begin{cases} w_{ij} = -2B(1 - a_{ij})(1 - \delta_{ij}), \\ I_i = A \end{cases} \quad i = 1, 2, \ldots, n \tag{15}$$

In Eq. (15), the notation δ_{ij} is 1 if $i = j$, 0 otherwise.

3.2. An Improved HNN

Wang et al.[11, 12] proposed a better solution to the MCP, which we shall refer to as the "improved HNN" algorithm. In this algorithm, the internal dynamics of the HNN is modified to efficiently increase exchange of information between neurons, and permit temporary increases in the energy function in order to avoid local minima. Simulation results on the p-random graph and the k-random clique graph have demonstrated that it is good in approximating MCP compared with that of Funabiki's Binary Neural Network[13] and RaCLIQUE.[14]

The detailed description of the improved HNN is as follows.

The updating mode of the internal potential has been modified as such:

$$x_i(t+1) = \alpha_i(y_i, t) \cdot x_i(t) + \frac{dx_i(t)}{dt} \tag{16}$$

Table I. Two sets of DNA sequence generated randomly.

Number	Set 1	Number	Set 2
1-01	GACCTAGCGGTTCTACATAC	2-01	AATTCGCCTTTAGGCACGGG
1-02	ATCTTTCCGTCTCAGCCTTG	2-02	*GTCCACCTATAGATTGCTAG*
1-03	*GCTCCGATTAAGAACTCCGA*	2-03	CTTGAGAGGCCGTGACATAG
1-04	GTGATCGTAGCTAGGGCAA	2-04	CCAAATACGTGAACTGCAGG
1-05	CCTCTGGTAATTCCAACACT	2-05	GCAGCGCACAGTCATCATAG
1-06	TTAACCTCTCCGTGCCACTT	2-06	ACGAGTAGCCCATTGAGATG
1-07	*CATTTAGTGCCGCTCGTTCA*	2-07	*ATGGGTTCTTGGAGTGCCTT*
1-08	CGCTATAGTTCGTGACCGGT	2-08	GCTCCATAAGCTTAGCCTAA
1-09	*CCTTGCGATAGGTATGTGAT*	2-09	CGGACTTTATTGACCACTAC
1-10	ACGTTCCTGGGCCCGTTAAA	2-10	TTTATGGGCCTTATAACGGG
1-11	CCCAATTCCCGATGCTGTGT	2-11	*GCGTATGGCGATGCGTGATT*
1-12	AGGTATCGCCTTCTATTGCG	2-12	ATCACTAGAGAAAGCACCCG
1-13	TGCGTCCAAACCTTGAAGGT	2-13	AGTAGGTTGCCACGACACGT
1-14	*GTATCTCTAGTCTGGTGAGT*	2-14	AGGGCGCGTTTGTCTCCAAT
1-15	CTTGGTACCGAATAGAGCCG	2-15	*AATGTTACGAGGTATCCTGG*
1-16	CCGCGAACACAGGACTTTAT	2-16	TTGGTAGGAACTGGTGAGCC
1-17	TTCCTCAGTGGAACCTTAGA	2-17	ATCCATCCCATCTTAGCTGT
1-18	ATGCATAGCGAGGTCTTATC	2-18	GACCCTCTGAGCTGTTTGCT
1-19	CAATCAGCTGAATCCCACAC	2-19	*CCCTGTGACTTGATCTTTGT*
1-20	ATAATTCTACGAGGCCGGCA	2-20	TGAACTCTGTAGAACGACCT
1-21	CGTAGTGACGGATGGTTCAC	2-21	GTGTTGTAGCGTATATGAGC
1-22	GGACTTATTGCACGCTTGAC	2-22	*GCAAATAGCCCGAGACACAA*
1-23	*GAGTAGATTGAGCTGACCCG*	2-23	TAGCGCCTGGTTGCAGATTG
1-24	TCTTACTTTCGGTCGGCTGT	2-24	TCGGCAATATCAACCGTAGA
1-25	AGCGACCTTATGTAACCCAG	2-25	AGGGTGAAGCGCATCAGATA
1-26	ACTCTCCTAAAGACGGTCTC	2-26	ATGAAGCAGAAGGAATGTCC
1-27	*CGTAGCCAACAGGTTTCGTG*	2-27	*CATAATCCATTCTCAGTCGG*
1-28	CTGCCTGTAACAGCAACTTG	2-28	CTTGGATCCACATCTAGTCC
1-29	GTTTATAACCACCTACCTCC	2-29	GGTCACCTGGAGTTATTGTT
1-30	AGACGCGATTCTGTGGAGCT	2-30	GCCTAATTCGTCTGGTCAGA
1-31	CAGAGAGTTGGCCTGACCTT	2-31	ACAACGATCGGTTCGCCGTT
1-32	ATCATTAGGTTCTGCTCCCT	2-32	GATTACGACAGACGCTTAAG
1-33	*TACTATCTCCCTCTAAAGGC*	2-33	ACTTCTCGTTGCGACCCGAA
1-34	AACATGAATTCATGGGCGCA	2-34	CATTTCGAGATCCGCACTTA
1-35	*CGAAGTGTCATTGCTTCTTC*	2-35	GACAGAGCTGCAATGTAAGC
1-36	TTGTTAGACCACCCTTGTGC	2-36	*GCTATCTCTCCTTATTTCCG*
1-37	AAAGGGTCAAACCGGGTAGT	2-37	TGTTGCGGTACGAGTTGTCT
1-38	*GGCTTGAATGCTATTACGTG*	2-38	TACGTGGACTGTCCTCAACA
1-39	GTAAGTGGTGCACGGTGCAA	2-39	*GCGGACGATTACATAGTTCG*
1-40	CCTACTAGCTCTGGAGTTGT	2-40	CCAAACGGTTCTAAAGCGTG
1-41	*CGCCTAATATCCATTAACGC*	2-41	TGTGAGTGATTGAGGGCACT
1-42	CATAGATACGTTGAGTGGGC	2-42	AGCATTACCTTCGCTCACTG
1-43	CCGGTGGCAACATATTCTCA	2-43	GACGTCTACGTTACCGGATG
1-44	TCTCGGAGACTCATCGTGTA	2-44	AGGCTGTCATGTTTGACCGC
1-45	ATAATTTGACTCTCGGGAGC	2-45	*CAGAAGCTGGTACAATGACG*
1-46	*TCGACTTAACTTTGCCCACC*	2-46	CAGGAAGATTCGCGGCATCA
1-47	*CACAACCCAAGTTATCATGC*	2-47	AATTACGGTGACTGGCTAGT
1-48	AGGGCGAGTGCTATATCACA	2-48	*TCTCATCTGTTCCCTCATGA*
1-49	ACCGCCATTGATATCGTACT	2-49	AAATACTCGGAGTGGGAGCA
1-50	TCATACTGGCGCCCTTGATC	2-50	GGCCAGGAATTAAATATCCC

where $\alpha_i(y_i, t)$ represents the stabilization of neuron i, which is used to control the internal potential change in any neuron, and is defined as follows:

$$\alpha_i(y_i, t) = 1 - e^{-((1/2 - y_i(t))^2/\varepsilon)(t/\lambda)} \tag{17}$$

And clearly, $0 \leq \alpha_i(y_i, t) \leq 1$, where y_i is the state of neuron i, t is the updating iteration, and ε and λ are constants that decide the neuron growth speed and network convergent speed. To ensure the network converges to a stable state as soon as possible, ε and λ are selected around 1 and at or near the maximum number of iteration steps allowed by a user, respectively.

3.3. Algorithm

Thus, we obtain the program for MNCCP based on the improved HNN algorithm, which consists of the following.

Step 1. Set parameters A, B, T, ε and λ, and set iteration step $t = 1$;

Step 2. Randomly initialize the internal potential for x_i $(i = 1, 2, \ldots, n)$;

Step 3. Update the neuron state y_i $(i = 1, 2, \ldots, n)$ using Eq. (9);

Step 4. Set *loop_time* $= 1$;

Step 5. Loop until *loop_time* $\geq n$, where n is the number of vertices;

(a) Randomly select a neuron i;

(b) Use Eqs. (17, 16) to update the internal potential x_i of neuron i;

(c) Use Eq. (9) to update neuron state y_i;

(d) Increase the *loop_time* by 1;

Step 6. Increase the t by 1;

Step 7. If the system reaches equilibrium state, go to Step 8; otherwise return to Step 4;

Step 8. Calculate the maximum clique using the stable state of the network.

4. COMPUTATIONAL RESULTS

This improved Hopfield network algorithm has been experimented using C++ programming language to randomly generate sets of DNA sequences. We let $A = B = 1$ in the experiment, and gave ε and λ the value of 0.1 and 135, respectively. The temperature parameter T in Eq. (9) was set to 0.64. For this paper, we only randomly selected two sets of 50 DNA sequences (as shown in Table I) to show the simulation results (as shown in Fig. 1).

In Table I, the sequences marked in italic type represent the computational codeword and the corresponding vertices in the maximal clique of graph (a) and graph (b), respectively, in Figure 1. The simulation results show that there are 12 and 11 computational codewords available in set 1 and set 2, respectively.

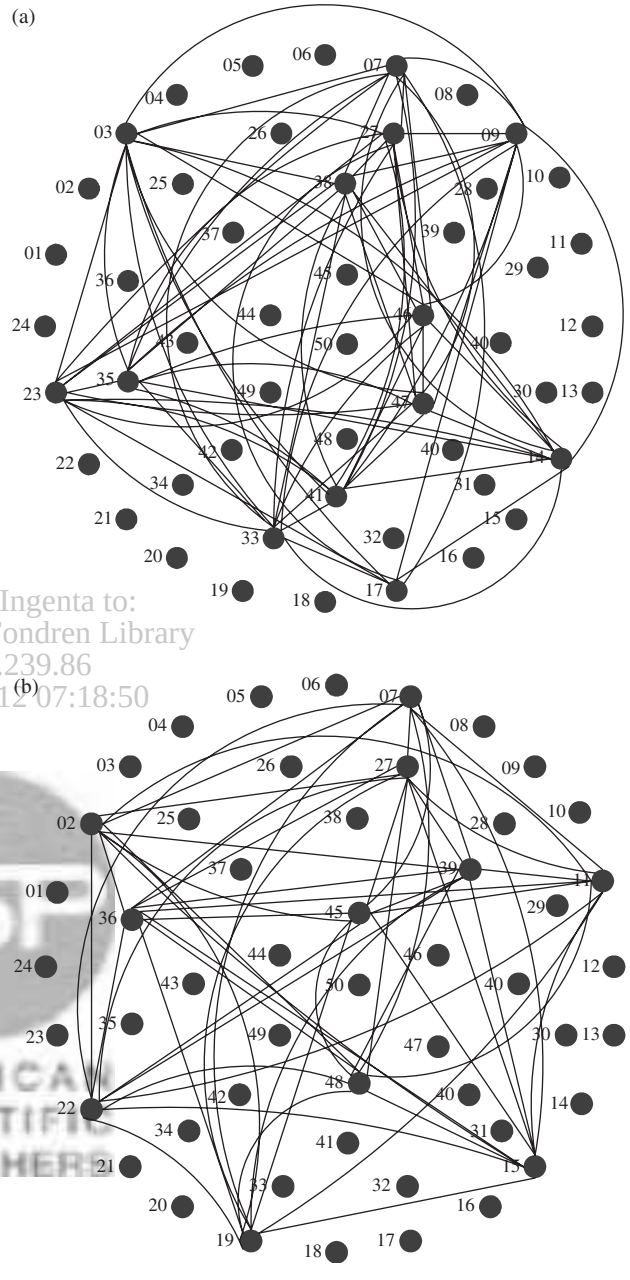

Fig. 1. The computational codeword in Table I and their corresponding maximal clique graphs. Graph (a) corresponds to set 1, and the graph with maximal clique {03, 07, 09, 14, 23, 27, 33, 35, 38, 41, 46, 47} corresponds to the computational codeword in set 1. Graph (b) corresponds to set 2, and the graph with maximal clique {02, 07, 11, 15, 19, 22, 27, 36, 39, 45, 48} corresponds to the computational codeword in set 2.

5. DISCUSSION

In this paper, we have proposed an efficient algorithm to resolve the MNCCP problem based on the improved Hopfield neural network, and shown its effectiveness by simulation experiments. The simulation results show that the proposed method is useful for a user to select an appropriate set of candidate DNA sequences to

filter and obtain good computational codeword for DNA computing.

It should be mentioned that considering only the melting temperature (T_m) and the free energy (ΔG) may in fact be enough for designing computational codeword, because the two parameters govern the kinetics of hybridization and ligation. Besides the above three constraints, the factor that should be considered likewise might be the shifting free energy between single-stranded DNA sequences. If this is utilized by concatenating single-stranded DNA sequence to perform DNA computing, the free energy should also be considered for the set of long strands obtained by concatenating these computational codeword. Even if all the factors are considered, a clique cannot be obtained, because the radio of computational codeword to candidate DNA sequences is so small. This is the reason why we only considered primary structure, melting temperature and free energy of single-stranded DNA sequence in this paper.

Acknowledgments: This work is supported by the National Natural Science Foundation of China (60573190), Natural Science Foundation of Henan Province (0511011600, 0611052400), and Natural Science Foundation of Zhejiang Province (Y106654).

References

1. M. H. Garzon and R. J. Deaton, *Natural Computing* 3, 253 **(2004)**.
2. M. H. Garzon, R. J. Deaton, P. Neathery, R. C. Murphy, D. R. Franceschetti, and E. Stevens, Jr., *Proc. 3rd DIMACS Workshop DNA Based Comput.* **(1997)**, pp. 230–237.
3. Y. F. Wang, G. Z. Cui, B. Y. Huang, and X. C. Zhang, *J. Dynamics of Continuous, Discrete and Impulsive Systems, Series B.* S1, 3410 **(2006)**.
4. S.-Y. Shin, I.-H. Lee, D. Kim, and B.-T. Zhang, *IEEE Trans. Evol. Comput.* 9, 143 **(2005)**.
5. F. J. Burpo, A critical review of PCR primer design. Algorithms and cross-hybridization case study, Technical report, Department of Chemical Engineering, Stanford University **(2001)**.
6. J. SantaLucia, Jr., *Proc. Nat. Acad. Sci. U.S.A.* 95, 1460 **(1998)**.
7. R. M. Karp, Reducibility Among Combinatorial Problems, Complexity of Computer Computations, Plenum Press, New York **(1972)** pp. 85–103.
8. J. J. Hopfield and D. W. Tank, *Science* 233, 625 **(1986)**.
9. J. Xu and Z. Bao, Neural Networks and Graph Theory, Science in China Series F-Information Sciences **(2002)**, Vol. 45, pp. 1–24.
10. J.-Y. Zhang, J. Xu, and Z. Bao, *Journal of Electronics* (in Chinese) 18, 122 **(1996)**.
11. R. L. Wang, Z. Tang, and Q. P. Cao, *IEEJ Trans. EIS* 123, 362 **(2003)**.
12. R. L. Wang, Z. Tang, and Q. P. Cao, *Neural Computation* 15, 1605 **(2003)**.
13. N. Funabiki and S. Nishikawa, *IEICE Trans. Fundamentals* E79-A, 452 **(1996)**.
14. A. Jagota, *IEEE Trans. Neural Networks*, 6, 724 **(1995)**.

Received: 27 April 2007. Accepted: 11 May 2007.