Research Article

Solving the Maximum Weighted Clique Problem Based on Parallel Biological Computing Model

Zhaocai Wang,¹ Jiangfeng Qin,² Zuwen Ji,³ Dongmei Huang,¹ and Lei Li⁴

¹School of Information Sciences, Shanghai Ocean University, Shanghai 201306, China

²Guangxi Institute of Water Resources Research, Nanning 530023, China

³State Key Laboratory of Simulation and Regulation of River Basin Water Cycle, China Institute of Water Resources and Hydropower Research, Beijing 100048, China

⁴Department of Civil Engineering, Xi'an University of Architecture & Technology, Xi'an 710055, China

Correspondence should be addressed to Zuwen Ji; zuwenji70@163.com

Received 2 July 2014; Revised 15 September 2014; Accepted 20 September 2014

Academic Editor: L. W. Zhang

Copyright © 2015 Zhaocai Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The maximum weighted clique (MWC) problem, as a typical NP-complete problem, is difficult to be solved by the electronic computer algorithm. The aim of the problem is to seek a vertex clique with maximal weight sum in a given undirected graph. It is an extremely important problem in the field of optimal engineering scheme and control with numerous practical applications. From the point of view of practice, we give a parallel biological algorithm to solve the MWC problem. For the maximum weighted clique problem with m edges and n vertices, we use fixed length DNA strands to represent different vertices and edges, fully conduct biochemical reaction, and find the solution to the MVC problem in certain length range with $O(n^2)$ time complexity, comparing to the exponential time level by previous computer algorithms. We expand the applied scope of parallel biological computation and reduce computational complexity of practical engineering problems. Meanwhile, we provide a meaningful reference for solving other complex problems.

1. Introduction

DNA computing, as a comprehensive discipline, can use DNA biological technologies to solve complex practical engineering problems. In 1994, Adleman [1] made use of DNA molecule operations to solve the Hamiltonian path problem with n vertices in $O(n)$ time complexity; simultaneously, he also demonstrated the strong parallel ability of DNA computing. In 1995, Lipton [2] figured out the NP-complete satisfiability problem utilizing Adleman's biochemical experiment. Since then, DNA biological computing attracted more and more interest from different disciplinary scholars. DNA biological computing has three advantages: high parallelism, low energy consumption, and large memory capacity. Many research scholars, designing DNA procedures and algorithms, succeed in solving multifarious kinds of complicated NP-complete problems [3–21], which promoted development of DNA computing. In order to better apply DNA computing

theory to more practical engineering science broad, it is worth trying to solve more intractable problems using the DNA molecular computing. Furthermore, most previous works, relating to DNA computing, focused on how to solve the path search problems that the solutions are continuous head-to-tail ligation edge or vertex sets, so that the possible solutions can be relatively easily represented by DNA strands, while some practical engineering problems, such as maximum weighted clique problem, are discrete set problems without sequentially connected path. So how to represent discrete data on DNA strands is an important key to expand the applied scope of DNA computing.

The maximum weighted clique problem has a wide range of applications in optimal engineering scheme and computational mathematics. In this paper, DNA algorithm, based on the research foundation of Adleman [1] and Lipton [2], is used to get solution of the maximum weighted clique problem. The rest dissertation is organized as follows.

In Section 2, the parallel biological computing model is introduced with detailed description. Section 3 uses DNA molecular algorithm to solve the maximum weighted clique problem. Section 4 proves DNA algorithm correctness and feasibility and gets the computation complexity. We come to the conclusions in Section 5.

2. The Parallel Biological Computing Model

DNA is the material basis of biological genetics, which is strung together from deoxyribonucleotides. DNA is formed by four kinds of base composition. These bases are, respectively, called adenine (A), guanine (G), cytosine (C), and thymine (T). The permutation and combination of bases store genetic information. An important feature of DNA is that two single strands can form a double strand through complementary base pairing. Moreover, the pairing has high specificity: A can only match T; C can only be paired with G. The length of a DNA single strand is counted by the number of bases. For example, a single strand (ATTTCG) includes 5 bases; then it is called a 5 *mer*.

Based on Adleman [1] and Lipton's [2] research, DNA biological algorithm operations are described as follows. Corresponding biological operations can be used to get solution of the maximum weighted clique problem. In the parallel biological computing model, we can perform the following operations with given tubes which contain a list of DNA strands.

- (1) *Copy*(P_1, P_2): given a test tube P_1 , it can get another test tube P_2 with the same strands as P_1 .
- (2) *Merge*(P_1, P_2): given two test tubes P_1 and P_2 , it can get the compound strands P_1 and P_2 in P_1 and leave P_2 empty.
- (3) *Annealing*(P): given a test tube P , it can generate all feasible double strands in P by annealing. The products and residues are still stored in P after annealing.
- (4) *Separation*(P_1, Z, P_2): given a test tube P_1 and a list strands set Z , it can remove all single strands in Z from P_1 and get an another tube P_2 with the removed strands.
- (5) *Ligation*(T): given a tube P , it is used to ligate together the strands in P .
- (6) *Sort*(P_1, P_2, P_3): it picks out the shortest length strands into tube P_2 from tube P_1 , the longest strands into P_3 , and the surplus strands are still kept in P_1 .
- (7) *Denaturation*(P): given a test tube P , it can dissociate every double strand in P to couple of single strands.
- (8) *Read*(P): given a tube P , it can be used to describe each single strand in P .
- (9) *Append-tail*(P, z): given a test tube P and a single strand z , it can append z at back of each strand in the tube P .
- (10) *Discard*(P): given a test tube P , it discards the strands in tube P and leave P empty.

Since above operations are realized through the limited biological experimental procedures with DNA strands [18], we can reasonably conclude that each operation is in $O(1)$ time complexity.

3. Biological Algorithm for the Maximum Weighted Clique Problem

An undirected simple graph $G = (V, E, W)$ is a pair of vertex set $V = \{v_1, v_2, \dots, v_n\}$ with corresponding vertex positive weight value $\{w_1, w_2, \dots, w_n\}$ and edge set $E = \{e_{i,j} \mid 1 \leq i < j \leq n\}$. For a vertex subset $T_1 \subseteq V$, if $\forall v_i, v_j \in T_1$, v_i and v_j can be linked by edge $e_{i,j}$ in the graph, then T_1 is called a clique of the graph G , and simultaneously the clique weight is the sum of vertex weight in the T_1 . The solution of maximum weighted clique problem aims to seek a vertex clique T of graph G with maximal weight sum. For example, the undirected simple graph in Figure 1 is defined as the MVC problem.

In succession, the symbols A_k, B_k ($k = 1, 2, \dots, n$), 1, 0, $\bar{\#}$, $\#$ are composed by different single strands having same length, as t *mer*. Certainly, t would be best to choose a small integer which can be determined by the scale of the problem. Then in the following algorithms, we use DNA single strands symbols $A_k 0 B_k, A_k 1 B_k$ to indicate the vertex v_k , with strands symbol $A_k 1 B_k$ for vertex v_k in the vertex subset while $A_k 0 B_k$ for not. Simultaneously, the symbols $\#, \bar{\#}$ are the signal of division between different vertex subsets. We denote DNA singled strands y_k to encode the vertex v_k weight value with length of w_k *mer*. For distinguishing some edges belonging to the graph G or not, we meantime design DNA strings $B_i A_j$ in the tube R if $e_{i,j} \in E$. Let

$$\begin{aligned} T_1 &= \{\bar{\#}, \overline{A_k 0 B_k}, \overline{A_k 1 B_k} \mid k = 1, 2, \dots, n\}, \\ T_2 &= \{\# A_1, B_n \#, 0, 1, B_{k-1} A_k \mid k = 2, 3, \dots, n\}, \\ R &= \{B_i A_j \mid e_{i,j} \in E\}, \\ Y &= \{y_k \mid k = 1, 2, \dots, n\}. \end{aligned} \quad (1)$$

For a n -vertex graph, every vertex subset can be expressed by a n -bit binary value. The k th bit set to 1 means the vertex v_k in the subset; on the contrary, the k th bit set to 0 shows the vertex v_k out of the subset. Taking Figure 1, for example, the vertex subset $\{v_2, v_3, v_5\}$ can be expressed by the binary value 01101. Using the same method, we can represent the vertex subsets of a n -vertex simple graph as a series of n -bit binary numbers.

- (1) We generate all possible vertex subsets in graph G ;

- (1) *Merge*(T_1, T_2);
- (2) *Annealing*(T_1);
- (3) *Ligation*(T_1);
- (4) *Denaturation*($T_1, \{\# A_1\}, T_3$);
- (5) *Discard*(T_1);
- (6) *Denaturation*($T_3, \{B_n \#\}, T_1$).

FIGURE 1: An undirected simple graph G with 5 vertices and 8 edges.

After the above six manipulations, the single strands in tube T_1 mean all kinds of vertex subsets. For example, in Figure 1, we have single strands:

$$\#A_11B_1A_21B_2A_30B_3A_41B_4A_51B_5\# \in T_1 \quad (2)$$

which denote the vertex subset $\{v_1, v_2, v_4, v_5\}$ corresponding to binary value 11011. These operations can be executed with $O(1)$ time complexity since every operation can be finished in $O(1)$.

- (2) Every strand in tube T_1 denotes one kind of vertex subset. For the maximum weighted clique problem, solution is one kind of vertex subset that arbitrary two vertices in the subset can be connected by one edge included in the graph G . Therefore, we check whether all vertex subsets in T_1 are in line with the condition or not. If $e_{i,j} \notin E$, we discard the strands indicating that both vertices v_i and v_j are in the same subset. For example, in Figure 1, the singled strands $\#A_10B_1A_21B_2A_30B_3A_41B_4A_51B_5\#$ representing the vertex subset $\{v_2, v_4, v_5\}$ should be discarded for not including the edge $e_{2,4}$ in graph G to connect vertices v_2 and v_4 . We choose all possible vertex cliques in graph.

For $i = 1$ to $i = n$,

- (1) *Separation*($P, \{A_i1B_i\}, T_2$);
For $j = 1$ and $j \neq i$ to $j = n$
- (2) *Separation*($R, \{B_iA_j\}, T_3$);
- (3) If(*Detect*(T_3));
Then
- (4) *Discard*(T_3);
- (5) *Separation*($T_2, \{A_j1B_j\}, T_4$);
- (6) *Merge*(P, T_2);

End for

End for

Through the above operations, all the single strands in tube T_1 represent different vertex clique subsets.

Meanwhile, the algorithm includes two “For” clauses, this step is executed in $O(n^2)$ time complexity since each operation can be finished in $O(1)$.

- (3) The maximum weighted clique problem should be a maximal vertex clique subset in which arbitrary two vertices should be linked by certain edge of the graph G . So we select the maximal vertex subset from all kinds of vertex clique subsets. If the vertex v_k is included in the vertex subset, we append additional strand y_k at the end of previous subset strand in order to find the optimum solution strand. For the singled strand (representing the vertex subset $\{v_2, v_3, v_5\}$)

$$\#A_10B_1A_21B_2A_31B_3A_40B_4A_51B_5\# \in T_1, \quad (3)$$

we append strands $\{y_2, y_3, y_5\}$ at end of the previous strand to

$$\#A_10B_1A_21B_2A_31B_3A_40B_4A_51B_5\#y_2y_3y_5. \quad (4)$$

This step can be carried out as follows.

For $k = 1$ to $k = n$,

- (1) *Separation*(T_1, A_k1B_k, T_7);
- (2) *Append-tail*(T_7, y_k);
- (3) *Merge*(T_1, T_7);
- (4) *Discard*(T_7).

End for

This step includes one “For” clause; thus it can be finished in $O(n)$ time complexity.

- (4) We select single strands with the longest length from T_1 , which represent the solutions of maximum weighted clique problem. For example, in Figure 1, single strands in T_1 with the largest length are

$$\#A_10B_1A_20B_2A_31B_3A_41B_4A_51B_5\#y_3y_4y_5. \quad (5)$$

Consequently, solution of maximum weighted clique problem for Figure 1 is vertex subset $\{v_3, v_4, v_5\}$ with weight sum 15.

- (1) *Sort*(T_1, T_8, T_9);
- (2) *Read*(T_9).

4. The Feasibility and Computational Complexity of the Parallel Biological Computing Algorithm

Theorem 1. The maximum weighted clique problem for a n -vertex graph can be solved by the biological computing algorithm.

Proof. At first, we get all kinds of the vertex combinational subset in the test tube after Step (1). For the maximum weighted clique problem, if $e_{i,j} \notin E$, vertices v_i and v_j should be not in the same subset. Therefore, basic biological manipulations remove illegal combinations and seek legal

TABLE 1: Sequences chosen to represent $\#, A_k, B_k, y_k (k = 1, 2, \dots, n)$ in the example of Figure 1.

Symbol	3'-5' DNA sequence	Symbol	3'-5' DNA sequence
A_1	CCT	B_1	ACT
A_2	CAC	B_2	TAT
A_3	CTA	B_3	TAA
A_4	TTC	B_4	ATC
A_5	AAC	B_5	CTT
0	CAA	1	AAT
#	TAC	y_1	GG
y_2	GGG	y_3	GGGGG
y_4	GGGGGG	y_5	GGGG

ones from solution space strands through the Step (2). At Step (3), we append a series of “tails” y_k at the end of the strands which imply the vertex v_k included in the vertex subset. Owing to the length of strands $\|y_k\| = w_k$, the longest length strands in the pool mean the solutions of maximum weighted clique problem. Besides, we can search and get the solution at the last step. $\square$

Theorem 2. *The solutions of maximum weighted clique problem for a n -vertex graph can be solved in $O(n^2)$ time complexity using DNA molecules computing.*

Proof. The parallel biological computing algorithm can be entirely executed in finite time complexity such as Steps (1) and (4) in $O(1)$, Step (3) in $O(n)$, and simultaneously Step (2) in $O(n^2)$ time complexity. The total algorithm complexity T is as below:

$$\begin{aligned}
T(\text{Step (1)}) &= O(1); \\
T(\text{Step (2)}) &= O(n); \\
T(\text{Step (3)}) &= O(n^2); \\
T(\text{Step (4)}) &= O(1); \\
T &= T(\text{Step (1)}) + T(\text{Step (2)}) \\
&\quad + T(\text{Step (3)}) + T(\text{Step (4)}); \\
&= O(1) + O(n^2) + O(n) + O(1); \\
&= O(n^2).
\end{aligned} \tag{6}$$

In conclusion, we can get the solutions of maximum weighted clique problems with n -vertices in $O(n^2)$ time complexity. $\square$

Theorem 3. *Solution strands to the maximum weighted clique problem with n -vertices can be found in the finite length range.*

Proof. After Step (1), the singled strands in tube T_1 denote all possible vertex subsets. These strands can be described as follows:

$$\#A_1x_1B_1A_2x_2B_2 \cdots A_kx_kB_k \cdots A_nx_nB_n\# \quad x_k = 0 \text{ or } 1. \tag{7}$$

At Step (2), the single strands in T_1 mean all possible vertex clique subsets. We design the fixed length strands of $\#, A_k, B_k$, and x_k , for

$$\|A_k\| = \|B_k\| = \|\#\| = \|x_k\| = t \text{ mer}. \tag{8}$$

S is defined as strands assemblage after Step (3). Then S can be described:

$$\#A_1x_1B_1A_2x_2B_2 \cdots A_kx_kB_k \cdots A_nx_nB_n\#y_{i_1} \cdots y_{i_n}. \tag{9}$$

Appending the strands y_{i_j} or not is decided whether there exists vertex v_{i_j} information strands $A_{i_j}1B_{i_j}$ on the previous strands. Due to the fact that the number of vertex v_k in sunset is between 1 and n , so after “append” operation, the strands is also in a finite length range:

$$\begin{aligned}
\|S\| &= \|\#\| + \|A_1\| + \|x_1\| + \|B_1\| \\
&\quad + \cdots + \|A_k\| + \|x_k\| + \|A_k\| + \cdots \\
&\quad + \|A_n\| + \|x_n\| + \|A_n\| + \|\#\| \\
&\quad + \|y_{i_1}\| + \cdots + \|y_{i_n}\| \\
&= 2\|\#\| + \sum_{k=1}^n \|A_k\| + \sum_{k=1}^n \|x_k\| \\
&\quad + \sum_{k=1}^n \|B_k\| + \sum_{x_k=1}^n \|y_k\| \\
&= (3n+2)t + \sum_{x_k=1}^n \|y_k\|
\end{aligned} \tag{10}$$

$$\because 0 \leq \sum_{x_k=1}^n \|y_k\| \leq \sum_{k=1}^n \|y_k\|$$

$$\text{Let } \sum_{k=1}^n \|y_k\| = m$$

$$\therefore (3n+2)t \leq \|S\| \leq (3n+2)t + m.$$

For the maximum vertex clique problem, the length of solution strands is between $(3n+2)t$ and $(3n+2)t + m$. Therefore, we can get the solution in appropriate length range at Step (4). $\square$

TABLE 2: Sequences chosen to represent all kinds of vertex subsets for the example of Figure 1.

Vertex subset	Symbols sequence	3'-5' DNA sequence
$\{v_1\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACCA ATATCTACAATAATTCC AAATCAACCAACTTTAC
$\{v_2\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACAA TTATCTACAATAATTCC AAATCAACCAACTTTAC
$\{v_3\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACCA ATATCTAAATTAATTCC AAATCAACCAACTTTAC
$\{v_4\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACCA ATATCTACAATAATTCA ATATCAACCAACTTTAC
$\{v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACCA ATATCTACAATAATTCC AAATCAACAATCTTTAC
$\{v_1, v_2\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACAA TTATCTACAATAATTCC AAATCAACCAACTTTAC
$\{v_1, v_3\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACCA ATATCTAAATTAATTCC AAATCAACCAACTTTAC
$\{v_1, v_4\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACCA ATATCTACAATAATTCA ATATCAACCAACTTTAC
$\{v_1, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACCA ATATCTACAATAATTCC AAATCAACAATCTTTAC
$\{v_2, v_3\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACAA TTATCTAAATTAATTCC AAATCAACCAACTTTAC
$\{v_2, v_4\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACAA TTATCTACAATAATTCA ATATCAACCAACTTTAC
$\{v_2, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACAA TTATCTACAATAATTCCA AATCAACAATCTTTAC
$\{v_3, v_4\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACCA ATATCTAAATTAATTCA ATATCAACCAACTTTAC
$\{v_3, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACCA ATATCTAAATTAATTCC AAATCAACAATCTTTAC
$\{v_4, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACCA ATATCTACAATAATTCA ATATCAACAATCTTTAC
$\{v_1, v_2, v_3\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACAA TTATCTAAATTAATTCC AAATCAACCAACTTTAC
$\{v_1, v_2, v_4\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACAA TTATCTACAATAATTCA ATATCAACCAACTTTAC

TABLE 2: Continued.

Vertex subset	Symbols sequence	3'-5' DNA sequence
$\{v_1, v_2, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACAA TTATCTACAATAATTCC AAATCAACAATCTTTAC
$\{v_1, v_3, v_4\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACCA ATATCTAAATTAATTCA ATATCAACCAACTTTAC
$\{v_1, v_3, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACCA ATATCTAAATTAATTCC AAATCAACAATCTTTAC
$\{v_1, v_4, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACCA ATATCTACAATAATTCA ATATCAACAATCTTTAC
$\{v_2, v_3, v_4\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACAA TTATCTAAATTAATTCA ATATCAACCAACTTTAC
$\{v_2, v_3, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACAA ATTATCTAAATTAATTCC AAATCAACAATCTTTAC
$\{v_2, v_4, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACAA TTATCTACAATAATTCA ATATCAACAATCTTTAC
$\{v_3, v_4, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACCA ATATCTAAATTAATTCA ATATCAACAATCTTTAC
$\{v_1, v_2, v_3, v_4\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACAA TTATCTAAATTAATTCA ATATCAACCAACTTTAC
$\{v_1, v_2, v_3, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACAA TTATCTAAATTAATTCC AAATCAACAATCTTTAC
$\{v_1, v_2, v_4, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACAA TTATCTACAATAATTCA ATATCAACAATCTTTAC
$\{v_1, v_3, v_4, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACCA ATATCTAAATTAATTCA ATATCAACAATCTTTAC
$\{v_2, v_3, v_4, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTCAAACCTCACAA TTATCTAAATTAATTCA ATATCAACAATCTTTAC
$\{v_1, v_2, v_3, v_4, v_5\}$	$\#A_10B_1A_21B_2A_30B_3A_40B_4A_50B_5\#$	TACCCTAATACTCACAA TTATCTAAATTAATTCA ATATCAACAATCTTTAC

5. The Detailed Approach and Walkthrough of the Biological Computing Algorithm

Taking Figure 1 as example, we describe operation result of each step. Due to the fact that biological computing algorithm depends on basic biochemical DNA molecules reactions which may cause errors in the process, it is an important matter to make biological computing more reliable by means of the DNA molecular sequence design. To have a better performance in hybridization reactions, we follow

[22] to accomplish the sequence design. For the problem of Figure 1, the program generates 3-base random sequences to represent symbols $\#$, A_k , B_k , and y_k . If the generated DNA sequence fails to pass any of the constraints, the program will regenerate a new DNA sequence. If the constraints are satisfied, the new DNA sequences are accepted. If all the DNA strands satisfy the constraints, the program has then succeeded and these sequences would be the outputs. The corresponding vertex symbol sequences are shown in Table 1. In accordance with the above design, we can get all kinds of

TABLE 3: Symbols sequences chosen to represent the different vertex cliques for the example of Figure 1.

Vertex cliques	Symbols sequence
$\{v_1, v_2\}$	#A ₁ 1B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 0B ₄ A ₅ 0B ₅ #
$\{v_1, v_3\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 0B ₅ #
$\{v_1, v_4\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 0B ₅ #
$\{v_1, v_5\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 0B ₃ A ₄ 0B ₄ A ₅ 1B ₅ #
$\{v_2, v_3\}$	#A ₁ 0B ₁ A ₂ 1B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 0B ₅ #
$\{v_2, v_4\}$	#A ₁ 0B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 0B ₅ #
$\{v_2, v_5\}$	#A ₁ 0B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 0B ₄ A ₅ 1B ₅ #
$\{v_3, v_4\}$	#A ₁ 0B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 1B ₄ A ₅ 0B ₅ #
$\{v_3, v_5\}$	#A ₁ 0B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 1B ₅ #
$\{v_4, v_5\}$	#A ₁ 0B ₁ A ₂ 0B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 1B ₅ #
$\{v_1, v_2, v_3\}$	#A ₁ 1B ₁ A ₂ 1B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 0B ₅ #
$\{v_1, v_2, v_5\}$	#A ₁ 1B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 0B ₅ #
$\{v_1, v_3, v_5\}$	#A ₁ 1B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 0B ₄ A ₅ 1B ₅ #
$\{v_2, v_3, v_5\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 1B ₄ A ₅ 0B ₅ #
$\{v_3, v_4, v_5\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 1B ₅ #
$\{v_1, v_2, v_3, v_5\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 1B ₅ #

TABLE 4: Symbols sequences chosen to represent the vertex weighted cliques for the example of Figure 1.

Vertex set	Symbols sequence
$\{v_1, v_2\}$	#A ₁ 1B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 0B ₄ A ₅ 0B ₅ # $\gamma_1\gamma_2$
$\{v_1, v_3\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 0B ₅ # $\gamma_1\gamma_2$
$\{v_1, v_4\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 0B ₅ # $\gamma_1\gamma_4$
$\{v_1, v_5\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 0B ₃ A ₄ 0B ₄ A ₅ 1B ₅ # $\gamma_1\gamma_5$
$\{v_2, v_3\}$	#A ₁ 0B ₁ A ₂ 1B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 0B ₅ # $\gamma_2\gamma_3$
$\{v_2, v_4\}$	#A ₁ 0B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 0B ₅ # $\gamma_2\gamma_4$
$\{v_2, v_5\}$	#A ₁ 0B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 0B ₄ A ₅ 1B ₅ # $\gamma_2\gamma_5$
$\{v_3, v_4\}$	#A ₁ 0B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 1B ₄ A ₅ 0B ₅ # $\gamma_3\gamma_4$
$\{v_3, v_5\}$	#A ₁ 0B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 1B ₅ # $\gamma_3\gamma_5$
$\{v_4, v_5\}$	#A ₁ 0B ₁ A ₂ 0B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 1B ₅ # $\gamma_4\gamma_5$
$\{v_1, v_2, v_3\}$	#A ₁ 1B ₁ A ₂ 1B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 0B ₅ # $\gamma_1\gamma_2\gamma_3$
$\{v_1, v_2, v_5\}$	#A ₁ 1B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 0B ₅ # $\gamma_1\gamma_2\gamma_5$
$\{v_1, v_3, v_5\}$	#A ₁ 1B ₁ A ₂ 1B ₂ A ₃ 0B ₃ A ₄ 0B ₄ A ₅ 1B ₅ # $\gamma_1\gamma_3\gamma_5$
$\{v_2, v_3, v_5\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 1B ₄ A ₅ 0B ₅ # $\gamma_2\gamma_3\gamma_5$
$\{v_3, v_4, v_5\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 1B ₃ A ₄ 0B ₄ A ₅ 1B ₅ # $\gamma_3\gamma_4\gamma_5$
$\{v_1, v_2, v_3, v_5\}$	#A ₁ 1B ₁ A ₂ 0B ₂ A ₃ 0B ₃ A ₄ 1B ₄ A ₅ 1B ₅ # $\gamma_1\gamma_2\gamma_3\gamma_5$

symbol representations of vertex subsets in Table 2 after Step (1). Step (2) discards the inappropriate vertex combinatorial sequences and retains the vertex clique sequences in Table 3. At Step (4), we append the corresponding weighted sequences which are showed in Table 4. Through the “Sort” operation at Step (4), we find the optimal solution to the maximum weighted clique problem of Figure 1 in Table 5.

TABLE 5: DNA sequences chosen to represent the solution of the maximum weighted clique problem.

	3'-TACCCTCAAACCTACCAATAT
$\{v_3, v_4, v_5\}$	CTAAATTAATTCAATATCAACAATC
	TTTACGGGGGGGGGGGGGGG-5'

6. Conclusions

In this paper, we present a parallel computing algorithm to solve the maximum weighted clique problem based on biological operations. Due to the fact that DNA biological computing has some advantages including high parallelism, low energy consumption, and large memory capacity, comparing to electronic computers low speed and limited memory, the method of DNA computing has attracted more and more attention. Besides, compared with the previous algorithms, our proposed algorithm has the following features: (1) we utilize fixed length DNA strands to generate the solution strands of the problem, the algorithm actually has lower error rate in hybrid operations; (2) the time cost of algorithm and solution strands length increase in linear proportion with the expansion of instance scale. For an undirected simple n -vertex graph, the parallel biological computing algorithm executes in $O(n^2)$ time complexity for the maximum weighted clique problem, having lower computational complexity than previous algorithms in exponential level. Although operations in our paper are on the basis of a theoretical model, the capacity to executive complicated operations in algorithm could help us understand more about the nature of computing and promote the better and faster development of biocomputing, more conducive for us to solve complex practical engineering problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the Open Research Fund of State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin, China Institute of Water Resources and Hydropower Research (Grant no. 2014ZY05), and “12th Five-Year Plan” to Support Science and Technology Project (Grant no. 2012BAB04B02). The project was also supported by CNSF (Grant nos. 61272098, 51409050, and 51108376), SSF (Grant no. 2014JQ7231), and Doctor fund of Shanghai Ocean University (A-2400-12-0000351).

References

- [1] L. M. Adleman, “Molecular computation of solutions to combinatorial problems,” *Science*, vol. 266, no. 5187, pp. 1021–1024, 1994.
- [2] R. J. Lipton, “DNA solution of hard computational problems,” *Science*, vol. 268, no. 5210, pp. 542–545, 1995.

- [3] A. Fujiwara, K. Matsumoto, and W. Chen, "Procedures for logic and arithmetic operations with DNA molecules," *International Journal of Foundations of Computer Science*, vol. 15, no. 3, pp. 461–474, 2004.
- [4] W. X. Li, D. M. Xiao, and L. He, "DNA ternary addition," *Applied Mathematics and Computation*, vol. 182, no. 2, pp. 977–986, 2006.
- [5] F. Guarnieri, M. Fliss, and C. Bancroft, "Making DNA add," *Science*, vol. 273, no. 5272, pp. 220–223, 1996.
- [6] X. C. Liu, X. F. Yang, S. L. Li, and Y. Ding, "Solving the minimum bisection problem using a biologically inspired computational model," *Theoretical Computer Science*, vol. 411, no. 6, pp. 888–896, 2010.
- [7] G. Paun, G. Rozenberg, and A. Salomaa, *DNA Computing*, Springer, New York, NY, USA, 1998.
- [8] J. Y. Lee, S.-Y. Shin, T. H. Park, and B.-T. Zhang, "Solving traveling salesman problems with DNA molecules encoding numerical values," *BioSystems*, vol. 78, no. 1–3, pp. 39–47, 2004.
- [9] X. L. Wang, Z. M. Bao, J. J. Hu, S. Wang, and A. Zhan, "Solving the SAT problem using a DNA computing algorithm based on ligase chain reaction," *BioSystems*, vol. 91, no. 1, pp. 117–125, 2008.
- [10] Z. Wang, D. Xiao, W. Li, and L. He, "A DNA procedure for solving the shortest path problem," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 79–84, 2006.
- [11] A. Brenneman and A. Condon, "Strand design for biomolecular computation," *Theoretical Computer Science*, vol. 287, no. 1, pp. 39–58, 2002.
- [12] M. Y. Guo, W.-L. Chang, M. Ho, J. Lu, and J. N. Cao, "Is optimal solution of every NP-complete or NP-hard problem determined from its characteristic for DNA-based computing," *BioSystems*, vol. 80, no. 1, pp. 71–82, 2005.
- [13] Y. Wang, T. Yang, Y. Ma et al., "Mathematical modeling and stability analysis of macrophage activation in left ventricular remodeling post-myocardial infarction," *BMC Genomics*, vol. 13, supplement 6, article S21, 2012.
- [14] Y. Wang, H.-C. Han, J. Y. Yang, M. L. Lindsey, and Y. Jin, "A conceptual cellular interaction model of left ventricular remodelling post-MI: dynamic network with exit-entry competition strategy," *BMC Systems Biology*, vol. 4, supplement 1, article S5, 2010.
- [15] W.-L. Chang, "Fast parallel DNA-based algorithms for molecular computation: quadratic congruence and factoring integers," *IEEE Transactions on Nanobioscience*, vol. 11, no. 1, pp. 62–69, 2012.
- [16] Z. C. Wang, D. M. Huang, H. J. Meng, and C. P. Tang, "A new fast algorithm for solving the minimum spanning tree problem based on DNA molecules computation," *BioSystems*, vol. 114, no. 1, pp. 1–7, 2013.
- [17] Y. Xu and Z. Wang, "Genetic algorithm optimized CCEM for complex topology," *Mathematical Problems in Engineering*, vol. 2012, Article ID 383248, 14 pages, 2012.
- [18] W.-L. Chang, M. S.-H. Ho, and M. Guo, "Fast parallel molecular algorithms for DNA-based computation: factoring integers," *IEEE Transactions on Nanobioscience*, vol. 4, no. 2, pp. 149–163, 2005.
- [19] Z. C. Wang, Y. M. Zhang, W. H. Zhou, and H. F. Liu, "Solving traveling salesman problem in the Adleman-Lipton model," *Applied Mathematics and Computation*, vol. 219, no. 4, pp. 2267–2270, 2012.
- [20] T. Yang, Y. A. Chiao, Y. Wang et al., "Mathematical modeling of left ventricular dimensional changes in mice during aging," *BMC Systems Biology*, vol. 6, supplement 3, article S10, 2012.
- [21] Z. C. Wang, J. Tan, D. M. Huang, Y. C. Ren, and Z. W. Ji, "A biological algorithm to solve the assignment problem based on DNA molecules computation," *Applied Mathematics and Computation*, vol. 244, pp. 183–190, 2014.
- [22] R. S. Braich, C. Johnson, P. W. K. Rothmund, D. Hwang, N. Chelyapov, and L. M. Adleman, "Solution of a satisfiability problem on a gel-based DNA computer," in *Proceedings of the 6th International Conference on DNA Computation (DNA '00)*, vol. 2054 of *Lecture Notes in Computer Science*, pp. 27–42, 2000.