Available online at www.sciencedirect.com

BioSystems 80 (2005) 71–82

www.elsevier.com/locate/biosystems

Is optimal solution of every NP-complete or NP-hard problem determined from its characteristic for DNA-based computing[☆]

Minyi Guo[a,b,*], Weng-Long Chang[c], Machael Ho[c], Jian Lu[b], Jiannong Cao[d]

[a] *Department of Computer Software, The University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan*
[b] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, PR China*
[c] *Department of Information Management, Southern Taiwan University of Technology, Tainan County, Taiwan*
[d] *Department of Computing, Hong Kong Polytechnic University, Hong Kong*

Received 1 July 2004; received in revised form 15 October 2004; accepted 16 October 2004

Abstract

Cook's Theorem [Cormen, T.H., Leiserson, C.E., Rivest, R.L., 2001. Introduction to Algorithms, second ed., The MIT Press; Garey, M.R., Johnson, D.S., 1979. Computer and Intractability, Freeman, San Fransico, CA] is that if one algorithm for an NP-complete or an NP-hard problem will be developed, then other problems will be solved by means of reduction to that problem. Cook's Theorem has been demonstrated to be correct in a general *digital electronic* computer. In this paper, we first propose a DNA algorithm for solving the *vertex-cover problem*. Then, we demonstrate that if the size of a reduced NP-complete or NP-hard problem is equal to or less than that of the vertex-cover problem, then the proposed algorithm can be directly used for solving the reduced NP-complete or NP-hard problem and Cook's Theorem is correct on DNA-based computing. Otherwise, a new DNA algorithm for optimal solution of a reduced NP-complete problem or a reduced NP-hard problem should be developed from the characteristic of NP-complete problems or NP-hard problems.

Keywords: Molecular computing; DNA-based computing; NP-complete problems; NP-hard problems; Vertex-cover problem; Cook's Theorem.

1. Introduction

Adleman wrote the first paper in which it was demonstrated that deoxyribonucleic acid (DNA) strands could be applied for figuring out solutions to an instance of the NP-complete Hamiltonian path problem (HPP) (Adleman, 1994). Lipton wrote the second paper in which it was shown that the Adleman techniques could also be used to solve the NP-complete satisfiability (SAT) problem (the first NP-complete problem) (Lipton, 1995). Adleman and co-workers proposed *sticker* for enhancing the Adleman–Lipton model (Roweis et al., 1999).

In this paper, we use *sticker* to construct solution space of DNA library sequences for the *vertex-cover problem*. Simultaneously, we also apply DNA opera-

[☆] This work is supported in part by China National 973 Program under Grant 2002CB312002 and China NSFC (60273034, 60233010).

[*] Corresponding author. Tel.: +81 2423 72557; fax: +81 2423 72744.
E-mail address: minyi@u-aizu.ac.jp (M. Guo).

0303-2647/$ – see front matter © 2004 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2004.10.003

tions in the Adleman–Lipton model to develop a DNA algorithm. The main result of the proposed DNA algorithm shows that the vertex-cover problem is solved with biological operations in the Adleman–Lipton model from the solution space of stickers. Furthermore, if the size of a reduced NP-complete or a reduced NP-hard problem is equal to or less than that of the vertex-cover problem, then the proposed algorithm can be directly used for solving the reduced NP-complete, or NP-hard problem.

The rest of this paper is organized as follows. In Section 2, the Adleman–Lipton model is introduced and the comparison is made with other models. In Section 3, the first DNA algorithm is proposed for solving the vertex-cover problem from solution space of sticker in the Adleman–Lipton model. In Section 4, the experimental result of simulated DNA computing is also given. Conclusions are drawn in Section 5.

2. DNA model of computation

In Section 2.1, a summary of DNA structure and the Adleman–Lipton model is described in detail. In Section 2.2, the comparison of the Adleman–Lipton model with other models is also introduced.

2.1. The Adleman–Lipton model

A DNA is a *molecule* that plays the main role in DNA based computing (Paun et al., 1998). In the biochemical world of large and small *molecules, polymers* and *monomers*, DNA is a polymer, which is strung together from monomers called *deoxyribonucleotides*. The monomers used for the construction of DNA are deoxyribonucleotides which each deoxyribonucleotide containing three components: a *sugar*, a *phosphate* group and a *nitrogenous* base. This sugar has five carbon atoms—for the sake of reference there is a fixed numbering of them. Because the base also has carbons, to avoid confusion the carbons of the sugar are numbered from 1′ to 5′ (rather than from 1 to 5). The phosphate group is attached to the 5′ carbon, and the base is attached to the 1′ carbon. Within the sugar structure there is a hydroxyl group attached to the 3′ carbon.

Distinct nucleotides are detected only with their bases, which come in two sorts: purines and pyrimidines (Sinden, 1994; Paun et al., 1998). Purines include *adenine* and *guanine*, abbreviated A and G. Pyrimidines contain *cytosine* and *thymine*, abbreviated C and T. Because nucleotides are only distinguished from their bases, they are simply represented as A, G, C, or T nucleotides, depending upon the sort of base that they have. The structure of a nucleotide is illustrated (in a very simplified way) in Fig. 1. In Fig. 1, **B** is one of the four possible bases (A, G, C, or T), **P** is the phosphate group and the rest (the “stick”) is the sugar base (with its carbons enumerated 1′ through 5′).

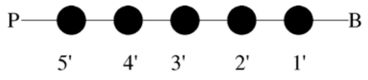

Fig. 1. A schematic representation of a nucleotide.

Nucleotides can link together in two different ways (Sinden, 1994; Boneh et al., 1996; Paun et al., 1998). The first method is that the 5′-phosphate group of one nucleotide is joined with 3′-hydroxyl group of the other forming a *phos-phodiester* bond. The resulting molecule has the 5′-phosphate group of one nucleotide, denoted as 5′ end, and the 3′-**OH** group of the other nucleotide available, denoted as 3′ end, for bonding. This gives the molecule the *directionality*, and we can talk about the direction of 5′ end to 3′ end or 3′ end to 5′ end. The second way is that the base of one nucleotide interacts with the base of the other to form a hydrogen bond. This bonding is the subject of the following restriction on the base pairing: A and T can pair together, and C and G can pair together—no other pairings are possible. This pairing principle is called the Watson–Crick complementarity (named after James D. Watson and Francis H.C. Crick who deduced the famous double helix structure of DNA in 1953, and won the Nobel Prize for the discovery).

A DNA strand is essentially a sequence (polymer) of four types of nucleotides detected by one of four bases they contain. Two strands of DNA can form (under appropriate conditions) a double strand, if the respective bases are the Watson–Crick complements of each other—A matches T and C matches G; also 3′ end matches 5′ end. The length of a single stranded DNA is the number of nucleotides comprising the single strand. Thus, if a single stranded DNA includes 20 nucleotides, then we say that it is a 20 mer (it is a polymer containing 20 monomers). The length of a double stranded DNA (where each nucleotide is base paired) is counted in the number of base pairs. Thus, if we make a double stranded DNA from a single stranded 20 mer, then the length of the double stranded DNA is 20 base pairs, also written 20 bp (for more discussion of the relevant biological background refer to Sinden, 1994; Boneh et al., 1996; Paun et al., 1998).

In the Adleman–Lipton model (Adleman, 1994; Lipton, 1995), splints were used to correspond to the edges of a particular graph the paths of which represented all possible binary numbers. A s it stands, their construction indiscriminately builds all splints that lead to a complete graph. This is to say that hybridization has higher probabilities of errors. Hence, Adleman et al. (Roweis et al., 1999) proposed the sticker-based model, which was an abstract model of molecular computing based on DNAs with a random access memory and a new form of encoding the information, to enhance the Adleman–Lipton model.

The DNA operations in the Adleman–Lipton model are described below (Adleman, 1994; Lipton, 1995; Boneh et al., 1996; Adleman, 1996). These operations will be used for figuring out solutions of the vertex-cover problem.

The Adleman–Lipton model

A (test) tube is a set of molecules of DNA (i.e., a multi-set of finite strings over the alphabet {A, C, G, T}). Given a tube, one can perform the following operations:

(1) *Extract*. Given a tube P and a short single strand of DNA, S, produce two tubes $+(P, S)$ and $-(P, S)$, where $+(P, S)$ is all of the molecules of DNA in P, which contain the strand S as a sub-strand and $-(P, S)$ is all of the molecules of DNA in P, which do not contain the short strand S.
(2) *Merge*. Given tubes P_1 and P_2, yield $\cup(P_1, P_2)$, where $\cup(P_1, P_2) = P_1 \cup P_2$. This operation is to pour two tubes into one, with no change of the individual strands.
(3) *Detect*. Given a tube P, say 'yes' if P includes at least one DNA molecule, and say 'no' if it contains none.
(4) *Discard*. Given a tube P, the operation will discard the tube P.
(5) *Read*. Given a tube P, the operation is used to describe a single molecule, which is contained in the tube P. Even if P contains many different molecules each encoding a different set of bases, the operation can give an explicit description of exactly one of them.

2.2. Other related work and comparison with the Adleman–Lipton model

Based on solution space of *splint* in the Adleman–Lipton model, their methods (Narayanan and Zorbala, 1998; Chang and Guo, 2002a, 2002b, 2002c, 2002d) could be applied towards solving the traveling salesman problem, the dominating-set problem, the vertex-cover problem, the clique problem, the independent-set problem, the 3-dimensional matching problem and the set-packing problem. Those methods for the problems show exponentially increasing volumes of DNA and linearly increasing time.LaBean et al. (2000) proposed an $n1.89^n$ volume, $O(n^2 + m^2)$ time molecular algorithm for the 3-coloring problem and a 1.51^n volume, $O(n^2m^2)$ time molecular algorithm for the independent set problem, where n and m are, subsequently, the number of vertices and the number of edges in the problems resolved. Fu (1997) presented a polynomial-time algorithm with a 1.497^n volume for the 3-SAT problem, a polynomial-time algorithm with a 1.345^n volume for the 3-coloring problem and a polynomial-time algorithm with a 1.229^n volume for the independent set. Though the size of those volumes (Fu, 1997; LaBean et al., 2000) is lower, constructing those volumes is more difficult and the time complexity to the methods is very higher.

Quyang et al. (1997) showed that restriction enzymes could be used to solve the NP-complete clique problem. The maximum number of vertices that they can process is limited to 27 because the size of the pool with the size of the problem exponentially increases (Quyang et al., 1997). Shin et al. (1999) presented an encoding scheme for decreasing error rate in hybridization. The method (Shin et al., 1999) could be employed towards ascertaining the traveling salesman problem for representing integer and real values with fixed-length codes. Arita et al. (1997) and Morimoto et al. (1999), respectively, proposed new molecular experimental techniques and a solid-phase method to find a Hamiltonian path. Amos (1997) proposed parallel filtering model for resolving the Hamiltonian path problem, the sub-graph isomorphism problem, the 3-

vertex-colorability problem, the clique problem and the independent-set problem. Those methods (Arita et al., 1997; Morimoto et al., 1999; Amos, 1997) have lower error rate in real molecular experiments.

In the literature (Reif et al., 2000; LaBean et al., 2000; LaBean and Reif, 2001), the methods for DNA-based computing by self-assembly require to use DNA nanostructures, called tiles, that have efficient chemistries, expressive computational power, and convenient input and output (I/O) mechanisms. DNA tiles have very lower error rate in self-assembly. Garzon and Deaton (1999) introduced a review of the most important advances in molecular computing.

Adleman and co-workers (Roweis et al., 1999) proposed sticker-based model to enhance error rate in hybridization in the Adleman–Lipton model. Their model could be used for determining solutions to an instance of the set cover problem. Perez-Jimenez and Sancho-Caparrini (2001) employed sticker-based model (Roweis et al., 1999) to solve knapsack problems. In our previous work, Chang and Guo (2004) and Chang et al. (2003) also employed the sticker-based model and the Adleman–Lipton model to deal with the dominating-set problem and the set-splitting problem for decreasing error rate of hybridization.

3. Using sticker for solving the vertex-cover problem in the Adleman–Lipton model

In Section 3.1, the vertex-cover problem is described. Applying sticker to constructing solution space of DNA sequences for the vertex-cover problem is introduced in Section 3.2. In Section 3.3, one DNA algorithm is proposed to resolve the vertex-cover problem. In Section 3.4, the complexity of the proposed algorithm is offered. In Section 3.5, the range of application to famous Cook's Theorem is described in molecular computing.

3.1. *Definition of the vertex-cover problem*

Assume that a graph G can be represented as $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is a set of vertices in G and $E = \{(v_a, v_b) | v_a$ and v_b are, respectively, vertices in $V\}$ is a set of edges in G. $|V| = n$ is the number of vertex in V and $|E| = m$ is the number of edge in E.

Mathematically, a *vertex cover* of a graph G is a subset $V^1 \subseteq V$ of vertices such that for each edge (v_a, v_b) in E, at lease one of v_a and v_b belongs to V^1 (Cormen et al., 2001; Garey and Johnson, 1979). The vertex-cover problem is to find a minimum-size vertex cover from G. The problem has been shown to be a NP-complete problem (Garey and Johnson, 1979).

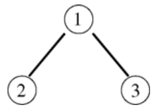

Fig. 2. The graph G of our problem.

The graph in Fig. 2 denotes such a problem. In Fig. 2, the graph G contains three vertices and two edges. The minimum-size vertex cover for G is $\{v_1\}$. Hence, the size of the vertex-cover problem in Fig. 2 is one. It is indicated from (Garey and Johnson, 1979) that finding a minimum-size vertex cover is an NP-complete problem, and it can be formulated as a search problem.

3.2. *Using sticker for constructing solution space of DNA sequence for the vertex-cover problem*

The first step in the Adleman–Lipton model is to yield solution space of DNA sequences for those problems solved. Next, basic biological operations are used to remove illegal solution and find legal solution from solution space. Thus, the first step of solving the vertex-cover problem is to generate a test tube, which includes all of the possible vertex covers. Assume that an n-digit binary number corresponds to each possible vertex cover to any n-vertex graph, G. Also suppose that V^1 is a vertex cover for G. If the ith bit in an n-digit binary number is set to 1, then it represents that the corresponding vertex is in V^1. If the ith bit in an n-digit binary number is set to 0, then it represents that the corresponding vertex is out of V^1.

By this way, all of the possible vertex covers in G are transformed into an ensemble of all n-digit binary numbers. Hence, with the way above, Table 1 denotes the solution space for the graph in Fig. 2. The binary number 000 in Table 1 represents that the corresponding vertex cover is empty. The binary numbers 001, 010 and 011 in Table 1 represent that those corresponding vertex covers are $\{v_1\}$, $\{v_2\}$ and $\{v_2, v_1\}$, respectively. The binary numbers 100, 101 and 110 in Table 1 represent that those corresponding vertex covers, subse-

Table 1
The solution space for the graph in Fig. 2

3-Digit binary number	The corresponding vertex-cover
000	Ø
001	$\{v_1\}$
010	$\{v_2\}$
011	$\{v_2, v_1\}$
100	$\{v_3\}$
101	$\{v_3, v_1\}$
110	$\{v_3, v_2\}$
111	$\{v_3, v_2, v_1\}$

quently, are $\{v_3\}$, $\{v_3, v_1\}$ and $\{v_3, v_2\}$. The binary number 111 in Table 1 represents that the corresponding vertex cover is $\{v_3, v_2, v_1\}$. Though there are eight 3-digit binary numbers for representing eight possible vertex covers in Table 1, not every 3-digit binary number corresponds to a *legal vertex* cover. Hence, in next section, basic biological operations are used to develop an algorithm for removing illegal vertex covers and finding legal vertex covers.

To implement this way, assume that an unsigned integer X is represented by a binary number x_n, x_{n-1}, ..., x_1, where the value of x_1 is 1 or 0 for $1 \leq i \leq n$. The integer X contains 2^n kinds of possible values. Each possible value represents a vertex cover for any n-vertex graph, G. Hence, it is very obvious that an unsigned integer X forms 2^n possible vertex cover. A bit x_i in an unsigned integer X represents the ith vertex in G. If the ith vertex is in a vertex cover, then the value of x_i is set to 1. If the ith vertex is out of a vertex cover, then the value of x^i is set to 0.

To represent all possible vertex covers for the vertex-cover problem, *sticker* (Roweis et al., 1999; Braich et al., 1999) is used to construct solution space for that problem solved. For every bit, x_i, two distinct 15 base value sequences are designed. One represents the value 1 and another represents the value 0 for x_i. For the sake of convenience of presentation, assume that x_i^1 denotes the value of x_i to be 1 and x_i^0 defines the value of x_i to be 0. Each of the 2^n possible vertex covers is represented by a library sequence of $15 \times n$ bases consisting of the concatenation of one value sequence for each bit. DNA molecules with library sequences are termed library strands and a combinatorial pool containing library strands is termed a library. The probes used for separating the library strands have sequences complementary to the value sequences.

It is pointed out from Roweis et al. (1999) and Braich et al. (1999) that errors in the separation of the library strands are errors in the computation. Sequences must be designed to ensure that library strands have little secondary structure that might inhibit intended probe-library hybridization. The design must also exclude sequences that might encourage unintended probe-library hybridization. To help achieve these goals, sequences were computer-generated to satisfy the following constraint (Braich et al., 1999):

(1) Library sequences contain only As, Ts and Cs.
(2) All library and probe sequences have no occurrence of 5 or more consecutive identical nucleotides; i.e., no runs of more than 4 A's, 4 T's, 4 C's or 4 G's occur in any library or probe sequences.
(3) Every probe sequence has at least four mismatches with all 15 base alignment of any library sequence (except for with its matching value sequence).
(4) Every 15 base subsequence of a library sequence has at least four mismatches with all 15 base alignment of itself or any other library sequence.
(5) No probe sequence has a run of more than seven matches with any eight base alignment of any library sequence (except for with its matching value sequence).
(6) No library sequence has a run of more than seven matches with any eight base alignment of itself or any other library sequence.
(7) Every probe sequence has 4, 5 or 6 Gs in its sequence.

Constraint (1) is motivated by the assumption that library strands composed only of As, Ts and Cs will have less secondary structure than those composed of As, Ts, Cs and Gs (Kalim, 1998). Constraint (2) is motivated by two assumptions: first, that long homopolymer tracts may have unusual secondary structure and second, that the melting temperatures of probe-library hybrids will be more uniform if none of the probe-library hybrids involve long homopolymer tracts. Constraints (3) and (5) are intended to ensure that probes bind only weakly where they are not intended to bind. Constraints (4) and (6) are intended to ensure that library strands have a low affinity for themselves. Constraint (7) is intended to ensure that intended probe-library pairings have uniform melting temperatures.

The Adleman program (Braich et al., 1999) is modified for generating those DNA sequences to satisfy the constraints above. For example, for representing the three vertices in the graph in Fig. 2, the DNA sequences generated are:

- x_1^0 = AAAACTCACCCTCCT;
- x_2^0 = TCTAATATAATTACT;
- x_3^0 = ATTCTAACTCTACCT;
- x_1^1 = TTTCAATAACACCTC;
- x_2^1 = ATTCACTTCTTTAAT; and
- x_3^1 = AACATACCCCTAATC.

Therefore, for every possible vertex cover to the graph in Fig. 2, the corresponding library strand is synthesized by employing a mix-and-split combinatorial synthesis technique (Cukras et al., 1998). Similarly, for any n-vertex graph, all of the library strands for representing every possible vertex cover could be also synthesized with the same technique.

3.3. *The DNA algorithm for solving the vertex-cover problem*

The following DNA algorithm is proposed to solve the vertex-cover problem:

Algorithm 1. Solving the vertex-cover problem.

```
(1) Input ($T_0$), where the tube $T_0$ includes solution space
    of DNA sequences to encode all of the possible ver-
    tex covers for any $n$-vertex graph, $G$, with those tech-
    niques mentioned in Section 3.2.
(2) For $k = 1$ to $m$, where $m$ is the number of edges in
    $G$.
     Assume that $e_k$ is $(v_i, v_j)$, where $e_k$ is one edge in $G$
    and $v_i$ and $v_j$ are vertices in $G$. Also suppose that bits
    $x_i$ and $x_j$, respectively, represent $v_i$ and $v_j$.
     (a) $\theta^1 = +(T_0, x_i^1)$ and $\theta = -(T_0, x_i^1)$.
     (b) $\theta^2 = +(\theta, x_j^1)$ and $\theta^3 = -(\theta, x_j^1)$.
     (c) $T_0 = \cup(\theta^1, \theta^2)$
EndFor
(3) For $i = 0$ to $n - 1$
     For $j = I$ down to 0
       (a) $T_{j+1}^{ON} = +(T_j, x_{i+1}^1)$ and $T_j = -(T_0, x_{i+1}^1)$.
       (b) $T_{j+1} = \cup(T_{j+1}, T_{j+1}^{ON})$.
     EndFor
EndFor
(4) For $k = 1$ to $n$
     (a) If (detect ($T_k$) = 'yes') then
     (b) Read ($T_k$) and terminate the algorithm.
     EndIf
EndFor
```

Theorem 3.1. *From those steps in Algorithm 1, the vertex-cover problem for any n-vertex graph G can be solved.*

Proof 1. In Step 1, a test tube of DNA strands that encode all 2^n possible input bit sequences $x_n, \ldots, x_1$, is generated. It is very clear that the test tube includes all 2^n possible vertex covers for any n-vertex graph, G.

From the definition of vertex cover (Cormen et al., 2001; Garey and Johnson, 1979), Step 2(a) applies "extraction" operation from the tube T_0 to form two test tubes: θ^1 and θ. The first tube θ^1 contains all of the strands that have $x_i = 1$. The second tube θ consists of all of the strands that have $x_i = 0$. It is very clear from the definition of vertex cover that the tube θ represents those sets which do not include the vertex v_i. Next, Step 2(b) uses "extraction" operation from the tube θ to form two new test tubes: θ^2 and θ^3. The test tube θ^2 includes all of the strands that have $x_i = 0$ and $x_j = 1$. The test tube θ^3 consists of all of the strands that have $x_i = 0$ and $x_j = 0$. It is indicated from the definition of vertex cover that the tubes θ^1 and θ^2 contain the strands, which satisfy the definition of vertex cover. Therefore, Step 2(c) applies "merge" operation to pour the tubes θ^1 and θ^2 into the tube T_0. After Steps 2(a)–2(c) are repeated to execute m times, the tube T_0 includes the strands, which represent those legal vertex covers.

When each time of the outer loop in Step 3 is executed, the number of execution for the inner loop is $(i + 1)$ times. The first time of the outer loop is executed, the inner loop is only executed one time. Therefore, Step 3(a) and 3(b) will also be executed one time. Step 3a uses "extraction" operation to form two test tubes: T_1^{ON} and T_0. The first tube T_1^{ON} contains all of the strands that have $x_1 = 1$. The second tube T_0 consists of all of the strands that have $x_1 = 0$. That is to say that

the first tube encodes every vertex cover including the first vertex and the second tube represents every vertex cover not including the first vertex. Hence, Step 3(b) applies "merge" operation to pour the tube T_1^{ON} into the tube T_1. After repeat to execute Steps 3(a) and 3(b), it finally produces n new tubes. The tube T_k for $n \geq k \geq 1$ encodes those vertex covers that contain k vertices.
Because the vertex-cover problem is to find a minimum-size vertex-cover, the tube T_1 first is detected with "detection" operation in Step 4(a). If it returns "yes", then the tube T_1 contains those vertex covers, which size is minimum. Therefore, Step 4(b) uses "read" operation to describe sequence of a molecular in the tube T_1 and the algorithm is terminated. Otherwise, repeat to execute Step 4(a) until a minimum-size vertex cover is found in the tube detected.

The graph in Fig. 1 is used to show the power of Algorithm 1. It is pointed out from Step 1 in Algorithm 1 that the tube T_0 is filled with eight library stands with the techniques mentioned in Section 3.2, representing eight possible vertex covers for the graph in Fig. 1. All of the edges in the graph in Fig. 1 are (v_1, v_2) and (v_1,v_3). So, the number of execution to Steps 2(a)–2(c) in Algorithm 1 is two times. From the first execution of Step 2a in Algorithm 1, two tubes are generated. The first tube, θ^1, includes those vertex covers: $\{v_1\}$, $\{v_2, v_1\}$, $\{v_3, v_1\}$ and $\{v_3, v_2, v_1\}$ and the second tube, θ, also contains those vertex covers: Ø, $\{v_2\}$, $\{v_3\}$ and $\{v_3, v_2\}$. Next, from the first execution of Step 2(b) in Algorithm 1, two tubes are yielded. The first tube, θ^2, includes those vertex covers: $\{v_2\}$ and $\{v_3, v_2\}$ and the second tube, θ^3, also contains those vertex covers: Ø and $\{v_3\}$. The tubes, θ^1 and θ^2, consist of the strands which satisfy the definition of vertex cover. Hence, Step 2(c) of Algorithm 1 pours the tube θ^1 and θ^2 into the tube T_0. Therefore, the tube T_0 now includes those vertex covers: $\{v_1\}$, $\{v_2, v_1\}$, $\{v_3, v_1\}$, $\{v_3, v_2, v_1\}$, $\{v_2\}$ and $\{v_3, v_2\}$. The same processing can be applied to deal with other edge (v_1, v_3). After every edge is processed, the remaining strands in the tube T_0 represent the legal vertex covers. That is to say that the tube T_0 contains those vertex covers: $\{v_1\}$, $\{v_2, v_1\}$, $\{v_3, v_1\}$, $\{v_3, v_2\}$ and $\{v_3, v_2, v_1\}$.

Because the number of vertex in the graph in Fig. 1 is three, the number of execution to the outer loop in Step 3 in Algorithm 1 is three times. The number of execution to the inner loop in Step 3 in Algorithm 1 is dependent on the value of the loop variable in the outer loop. After the execution of the first time to Step 3(a) and Step 3(b) is finished, the tube T_1 contains those vertex covers: $\{v_1\}$, $\{v_2, v_1\}$, $\{v_3, v_1\}$ and $\{v_3, v_2, v_1\}$ and the tube T_0 includes only the vertex cover: $\{v_3, v_2\}$. After repeating Steps 3(a) and 3(b), it finally produces three new tubes. The three tubes T_1, T_2 and T_3, respectively, include $\{\{v_2, v_1\}, \{v_3, v_1\}, \{v_3, v_2\}\}$ and $\{\{v_3, v_2, v_1\}\}$.

Because the tube T_1 is not empty, "detection" operation for detecting the tube T_1 in Step 4(a) in Algorithm 1 returns "yes". Therefore, Step 4(b) in Algorithm 1 reads the answer from the tube T_1. Thus, a minimum-size vertex cover for the graph in Fig. 1 is $\{v_1\}$. □

3.4. *The complexity of the proposed DNA algorithm*

The following theorems describe time complexity of Algorithm 1, volume complexity of solution space in Algorithm 1, the number of the tube used in Algorithm 1 and the longest library strand in solution space in Algorithm 1.

Theorem 3.2. *The vertex-cover problem for any undirected n-vertex graph G with m edges can be solved with* O(n^2)*biological operations in the Adleman–Lipton model, where n is the number of vertices in G and m is at most equal to* ($n(n-1)/2$).

Proof 2. Algorithm 1 can be applied for solving the vertex-cover problem for any undirected n-vertex graph G. Algorithm 1 includes three main steps. Step 2 is mainly used to determine legal vertex covers and to remove illegal vertex covers from all of the 2^n possible library strands. From Algorithm 1, it is very obvious that Steps 2(a) and 2(b) take $2 \times m$ "extraction" operations and Step 2(c) takes m "merge" operations. Step 3 is mainly applied to figure out the number of element in every legal vertex cover. It is indicated from Algorithm 1 that Step 3(a) takes ($n(n-1)/2$) "extraction" operations and Step 3(b) takes ($n(n-1)/2$) "merge" operations. Step 4 is used to find a minimum-size vertex cover from legal vertex cover. It is pointed out from Algorithm 1 that Step 4(a) at most takes n "detection" operations and Step 4(b) takes one "read" operation. Hence, from the statements mentioned above, it is at once inferred that the time complexity of Algorithm 1 is $O(n^2)$ biological operations in the Adleman–Lipton model. □

Theorem 3.3. *The vertex-cover problem for any undirected n-vertices graph G with m edges can be solved with sticker to construct* $O(2^n)$ *strands in the Adleman–Lipton model, where n is the number of vertices in G.*

Proof 3. Refer to Theorem 3.2. □

Theorem 3.4. *The vertex-cover problem for any undirected n-vertices graph G with m edges can be solved with* $O(n)$ *tubes in the Adleman–Lipton model, where n is the number of vertices in G.*

Proof 4. Refer to Theorem 3.2. □

Theorem 3.5. *The vertex-cover problem for any undirected n-vertices graph G with m edges can be solved with the longest library strand,* $O(15 \times n)$, *in the Adleman–Lipton model, where n is the number of vertices in G.*

Proof 5. Refer to Theorem 3.2. □

3.5. Range of application to Cook's Theorem in DNA computing

Cook's Theorem (Cormen et al., 2001; Garey and Johnson, 1979) is that if one algorithm for one NP-complete problem will be developed, then other problems will be solved by means of reduction to that problem. Cook's Theorem has been demonstrated to be correct in a general digital electronic computer. Assume that a collection C is $\{c_1, c_2, \cdots, c_m\}$ of clauses on a finite set U of variables, $\{u_1, u_2, \cdots, u_n\}$, such that —$c_x$— is equal to 3 for $1 \le x \le m$. The 3-satisfiability problem (3-SAT) is to find whether there is a truth assignment for U that satisfies all of the clauses in C. The simple structure for the 3-SAT problem makes it one of the most widely used problems for other NP-completeness results (Cormen et al., 2001). The following theorems are used to describe the range of application for Cook's Theorem in molecular computing

Theorem 3.6. *Assume that any other*NP-*complete problems or any other*NP-*hard problems can be reduced to the vertex-cover problem with a polynomial time algorithm in a general electronic computer. If the size of a reduced*NP-*complete problem or a reduced*NP-*hard problem is not equal to or less than that of the vertex-cover problem, then a new DNA algorithm for optimal solution of the reduced*NP-*complete problem or the reduced*NP-*hard problem should be developed from the characteristic of*NP-*complete problems or*NP-*hard problems.*

Proof 6. We transform the 3-SAT problem to the vertex-cover problem with a polynomial time algorithm (Cormen et al., 2001). Suppose that U is $\{u_1, u_2, \cdots, u_n\}$ and C is $\{c_1, c_2, \cdots, c_m\}$. U and C are any instance for the 3-SAT problem. We construct a graph $G = (V, E)$ and a positive integer $K \le |V|$ such that G has a vertex cover of size K or less if and only if C is satisfiable.

For each variable u_i in U, there is a truth-setting component $T_i = (V_i, E_i)$, with $V_i = \{u_i, u_i^1\}$ and $E_i = \{\{u_i, u_i^1\}\}$, that is, two vertices joined by a single edge. Note that any vertex cover will have to contain at least one of u_i and u_i^1 in order to cover the single edge in E_i. For each clause c_j in C, there is a satisfaction testing component $S_j = (V_j^1, E_j^1)$, consisting of three vertices and three edges joining them to form a triangle:

$$V_j^1 = \{a_j[j], a_2[j], a_3[j]\}$$

$$E_j^1 = \{\{a_j[j], a_2[j]\}, \{a_1[j], a_3[j]\}, \{a_2[j], a_3[j]\}\}$$

Note that any vertex cover will have to contain at least two vertices from V_j^1 in order to cover the edges in E_j^1. The only part of the construction that depends on which literals occur in which clauses is the collection of communication edges. These are best viewed from the vantage point of the satisfaction testing components. For each clause c_j in C, assume that the three literals in c_j is denoted as x_j, y_j and z_j. Then the communication edges emanating from S_j are given by:

$$E_j^2 = \{\{a_1[j], x_j\}, \{a_2[j], y_j\}, \{a_3[j], z_j\}\}.$$

The construction of our instance to the vertex-cover problem is completed by setting $K = n + (2 \times m)$ and $G = (V, E)$, where

$$V = \left(\bigcup_{i=1}^{n} E_i\right) \cup \left(\bigcup_{j=1}^{m} V_j^1\right)$$

and

$$E = \left(\bigcup_{i=1}^{n} E_i\right) \cup \left(\bigcup_{j=1}^{m} E_j^1\right) \cup \left(\bigcup_{j=1}^{m} V_E^2\right).$$

Therefore, the number of vertex and the number of edge in G are, respectively, $((2 \times n) + (3 \times m))$ and $(n + (6 \times m))$. This implies that Algorithm 1 is used to solve the reduced 3-SAT problem, then it will take $O(4 \times n^2 + 12 \times m \times n + 9 \times n^2)$ biological operations with $O(2^{2n+3m})$ DNA strands. However, from the characteristic of the 3-SAT problem, Lipton's algorithm will only take O(m) biological operations with $O(2^n)$ DNA strands. Therefore, it is inferred that if the size of a reduced NP-complete problem or a reduced NP-hard problem is not equal to or less than that of the vertex-cover problem, then a new DNA algorithm for optimal solution of the reduced NP-complete problem or the reduced NP-hard problem should be developed from the characteristic of NP-complete problems or NP-hard problems.

From Theorem 3.3–3.6, if the size of a reduced NP-complete problem or a reduced NP-hard problem is equal to or less than that of the vertex-cover problem, then Algorithm 1 can be directly used for solving the reduced NP-complete or the reduced NP-hard problem. Otherwise, a new DNA algorithm for optimal solution of a reduced NP-complete problem or a reduced NP-hard problem should be developed according to the characteristic of NP-complete or NP-hard problems. □

4. Experimental results of simulated DNA computing

We finished the modification of the Adleman program (Braich et al., 1999). This modified program is applied to generate DNA sequences for solving the vertex-cover problem. Because the source code of the two functions *srand48*() and *drand48*() was not found in the *original* Adleman program, we use the standard function *srand*() in C++ builder 6.0 to replace the function *srand48*() and added the source code to the function *drand48*(). We also added subroutines to the Adleman program for simulating biological operations in the Adleman–Lipton model in Section 2. We add subroutines to the Adleman program to simulate Algorithm 1 in Section 3.3.

Table 2
Sequences chosen to represent the vertices in the graph in Fig. 2

Vertex	DNA sequence
x_3^0	5′-ATTCTAACTCTACCT-3′
x_2^0	5′-TCTAATATAATTACT-3′
x_1^0	5′-AAAACTCACCCTCCT-3′
x_3^1	5′-AACATACCCCTAATC-3′
x_2^1	5′-ATTCACTTCTTTAAT-3′
x_1^1	5′-TTTCAATAACACCTC-3′

The Adleman program is used for constructing each 15-base DNA sequence for each bit of the library. For each bit, the program is applied for generating two 15-base random sequences (for 1 and 0) and checking to see if the library strands satisfy the seven constraints in Section 3.2 with the new DNA sequences added. If the constraints are satisfied, the new DNA sequences are greedily accepted. If the constraints are not satisfied then mutations are introduced one by one into the new block until either (A) the constraints are satisfied and the new DNA sequences are then accepted or (B) a threshold for the number of mutations is exceeded and the program has failed and then it exits, printing the sequence found so far. If n-bits that satisfy the constraints are found then the program has succeeded and it outputs these sequences.

Consider the graph in Fig. 2. The graph includes three vertices: v_1, v_2 and v_3. DNA sequences generated by the Adleman program modified were shown in Table 2. This program, respectively, took one mutation, one mutation and 10 mutations to make new DNA sequences for v_1, v_2 and v_3. With the nearest neighbor parameters, the Adleman program was used to calculate the enthalpy, entropy and free energy for the binding of each probe to its corresponding region on a library strand. The energy was shown in Table 3. Only G really matters to the energy of each bit. For example, the

Table 3
The energy for the binding of each probe to its corresponding region on a library strand

Vertex	Enthalpy energy (H)	Entropy energy (S)	Free energy (G)
x_3^0	105.2	277.1	22.4
x_2^0	104.8	283.7	19.9
x_1^0	113.7	288.7	27.5
x_3^1	112.6	291.2	25.6
x_2^1	107.8	283.5	23
x_1^1	105.6	271.6	24.3

Table 4
DNA sequences chosen represent all possible vertex covers

5′-ATTCTAACTCTACCTTCTAATATAATTACTAAAACTCACCCTCCT-3′ 3′-TAAGATTGAGATGGAAGATTATATTAATGATTTTGAGTGGGAGGA-5′
5′-ATTCTAACTCTACCTTCTAATATAATTACTTTTCAATAACACCTC-3′ 3′-TAAGATTGAGATGGAAGATTATATTAATGAAAAGTTATTGTGGAG-5′
5′-ATTCTAACTCTACCTATTCACTTCTTTAATAAAACTCACCCTCCT-3′ 3′-TAAGATTGAGATGGATAAGTGAAGAAATTATTTTGAGTGGGAGGA-5′
5′-ATTCTAACTCTACCTATTCACTTCTTTAATTTTCAATAACACCTC-3′ 3′-TAAGATTGAGATGGATAAGTGAAGAAATTAAAAGTTATTGTGGAG-5′
5′-AACATACCCCTAATCTCTAATATAATTACTAAAACTCACCCTCCT-3′ 3′-TTGTATGGGGATTAGAGATTATATTAATGATTTTGAGTGGGAGGA-5′
5′-AACATACCCCTAATCTCTAATATAATTACTTTTCAATAACACCTC-3′ 3′-TTGTATGGGGATTAGAGATTATATTAATGAAAAGTTATTGTGGAG-5′
5′-AACATACCCCTAATCATTCACTTCTTTAATAAAACTCACCCTCCT-3′ 3′-TTGTATGGGGATTAGTAAGTGAAGAAATTATTTTGAGTGGGAGGA-5′
5′-AACATACCCCTAATCATTCACTTCTTTAATTTTCAATAACACCTC-3′ 3′-TTGTATGGGGATTAGTAAGTGAAGAAATTAAAAGTTATTGTGGAG-5′

delta G for the probe binding a ‘1’ in the first bit is, thus, estimated to be 24.3 kcal/mol and the delta G for the probe binding a ‘0’ is estimated to be 27.5 kcal/mol.

The program simulated a mix-and-split combinatorial synthesis technique (Cukras et al., 1998) to synthesize the library strand to every possible vertex cover. Those library strands are shown in Table 4, and represent eight possible vertex covers: Ø, $\{v_1\}$, $\{v_2\}$, $\{v_2, v_1\}$, $\{v_3\}$, $\{v_3, v_1\}$, $\{v_3, v_2\}$ and $\{v_3, v_2, v_1\}$, respectively. The program is also applied to figure out the average and standard deviation for the enthalpy, entropy and free energy over all probe/library strand interactions. The energy is shown in Table 5. The standard deviation for delta G is small because this is partially enforced by the constraint that there are 4, 5 or 6 Gs (the seventh constraint in Section 3.2) in the probe sequences.

The Adleman program is employed for computing the distribution of the types of potential mishybridizations. The distribution of the types of potential mishybridizations is the absolute frequency of a probe-strand match of length k from 0 to the bit length 15 (for DNA sequences) where probes are not supposed to match the strands. The distribution is, subsequently, 106, 152, 183, 215, 216, 225, 137, 94, 46, 13, 4, 1, 0, 0, 0 and 0. It is pointed out from the last four zeros that there are 0 occurrences where a probe matches a strand at 12, 13, 14 or 15 places. This shows that the third constraint in Section 3.2 has been satisfied. Clearly, the number of matches peaks at 5 (225). That is to say that there are 225 occurrences where a probe matches a strand at 5 places.

The results for simulation of Step 2–4 were, respectively, shown in Tables 6–10. From the tube T_1, the answer was found to be $\{v_1\}$.

Table 5
The energy over all probe/library strand interactions

	Enthalpy energy (H)	Entropy energy (S)	Free energy (G)
Average	108.283	282.633	23.7833
Standard deviation	3.58365	6.63867	2.41481

Table 6
DNA sequences generated by Step 2 represent legal vertex covers

5′-ATTCTAACTCTACCTTCTAATATAATTACTTTTCAATAACACCTC-3′
5′-ATTCTAACTCTACCTATTCACTTCTTTAATTTTCAATAACACCTC-3′
5′-AACATACCCCTAATCTCTAATATAATTACTTTTCAATAACACCTC-3′
5′-AACATACCCCTAATCATTCACTTCTTTAATAAAACTCACCCTCCT-3′
5′-AACATACCCCTAATCATTCACTTCTTTAATTTTCAATAACACCTC-3′

Table 7
DNA sequence generated by Step 3 represents that vertex cover only containing one vertex

5′-ATTCTAACTCTACCTTCTAATATAATTACTTTTCAATAACACCTC-3′

Table 8
DNA sequences generated by Step 3 represent those vertex covers including two vertices

5′-ATTCTAACTCTACCTATTCACTTCTTTAATTTTCAATAACACCTC-3′
5′-AACATACCCCTAATCTCTAATATAATTACTTTTCAATAACACCTC-3′
5′-AACATACCCCTAATCATTCACTTCTTTAATAAAACTCACCCTCCT-3′

Table 9
DNA sequence generated by Step 3 represents that vertex cover containing three vertices

5′-AACATACCCCTAATCATTCACTTCTTTAATTTTCAATAACACCTC-3′

Table 10
DNA sequence generated by Step 4 represents the minimum-size vertex cover

5′-ATTCTAACTCTACCTTCTAATATAATTACTTTTCAATAACACCTC-3′

5. Conclusions

Cook's Theorem is that if one algorithm for an NP-complete or an NP-hard problem will be developed, then other problems will be solved by means of reduction to that problem. Cook's Theorem has been demonstrated to be right in a general digit electronic computer. In this paper, we showed that, from Theorem 3.3 to 3.6, if the size of a reduced NP-complete problem is equal to or less than that of the vertex-cover problem, then Cook's Theorem is right in molecular computing. Otherwise, a new DNA algorithm for optimal solution of a reduced NP-complete problem or a reduced NP-hard problem should be developed from the characteristic of NP-complete problems or NP-hard problems.

Chang and Guo (2002b, 2002d) applied splints to constructing solution space of DNA sequence for solving the vertex-cover problem in the Adleman–Lipton. This causes that hybridization has higher probabilities for errors. Adleman and co-workers (Roweis et al., 1999) proposed *sticker* to decrease probabilities of errors to hybridization in the Adleman–Lipton. The main result of the proposed algorithms shows that the vertex-cover problem is solved with biological operations in the Adleman–Lipton model from solution space of sticker. Furthermore, this work represents clear evidence for the ability of DNA based computing to solve NP-complete problems.

Currently, there still are lots of NP-complete problems not to be solved because it is very difficult to use basic biological operations for replacing mathematical operations. We are not sure whether molecular computing can be applied for dealing with every NP-complete problem. Therefore, in the future, our main work is to solve other unsolved NP-complete problems with the Adleman–Lipton model and the sticker model, or develop a new model.

References

Sinden, R.R., 1994. DNA Structure and Function. Academic Press.

Adleman, L., 1994. Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024.

Lipton, R.J., 1995. DNA solution of hard computational problems. Science 268, 542–545.

Quyang, Q., Kaplan, P.D., Liu, S., Libchaber, A., 1997. DNA solution of the maximal clique problem. Science 278, 446–449.

Arita, M., Suyama, A., Hagiya, M., 1997. A heuristic approach for Hamiltonian path problem with molecules. In: Proceedings of Second Genetic Programming (GP-97), pp. 457–462.

Morimoto, N., Arita, M., Suyama, A., 1999. Solid phase DNA solution to the Hamiltonian path problem. In: Series in Discrete Mathematics and Theoretical Computer Science, vol. 48, pp. 93–206.

Narayanan, A., Zorbala, S., 1998. DNA algorithms for computing shortest paths. In: Koza, J.R., et al. (Eds.), Genetic Programming 1998: Proceedings of the Third Annual Conference, pp. 718–724.

Shin, S.-Y., Zhang, B.-T., Jun, S.-S., 1994. Solving traveling salesman problems using molecular programming. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), vol. 2, pp. 994–1000.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., 2001. Introduction to Algorithms, second ed. The MIT Press.

Garey, M.R., Johnson, D.S., 1979. Computer and Intractability. Freeman, San Fransico, CA.

Boneh, D., Dunworth, C., Lipton, R.J., Sgall, J., 1996. On the computational power of DNA. In: Discrete Applied Mathematics, vol. 71, pp. 79–94 (special issue on computational molecular biology).

Adleman, L.M., 1996. On constructing a molecular computer. DNA based computers. In: Lipton, R., Baum, E. (Eds.), Series in Discrete Mathematics and Theoretical Computer Science American Mathematical Society, pp. 1–21.

Amos, M., 1997. DNA computation, Ph.D. Thesis, Department of Computer Science, The University of Warwick.

Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N.V., Goodman, M.F., Rothemund, P.W.K., Adleman, L.M., 1999. Sticker Based Model for DNA Computation, Princeton University, In: Landweber, L., Baum, E. (Eds.), In: Proceedings of the second annual workshop on DNA Computing, Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, pp. 1–29.

Perez-Jimenez, M.J., Sancho-Caparrini, F., 2001. Solving Knapsack Problems in a Sticker Based Model. In: Proceedings of the Seventh Annual Workshop on DNA Computing, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society.

Paun, G., Rozenberg, G., Salomaa, A., 1998. DNA Computing: New Computing Paradigms. Springer–Verlag, New York.

Chang, W.-L., Guo, M., 2002a. Solving the dominating-set problem in Adleman–Liptons Model. In: The Third International Conference on Parallel and Distributed Computing, Applications and Technologies, Japan, pp. 167–172.

Chang, W.-L., Guo, M., 2002b. Solving the clique problem and the vertex-cover problem in Adleman–Lipton's Model. In: IASTED International Conference on Networks, Parallel and Distributed Processing, and Applications, Japan, pp. 431–436.

Chang, W.-L., Guo, M., 2002c. Solving NP-complete problem in the Adleman–Lipton Model. In: The Proceedings of 2002 International Conference on Computer and Information Technology, Japan, pp. 157–162.

Chang, W.-L., Guo, M., 2002d. Solving the 3-dimensional matching problem and the set packing problem in Adleman–Lipton's Model. In: IASTED International Conference on Networks, Parallel and Distributed Processing, and Applications, Japan, pp. 455–460.

Fu, B., 1997. Volume bounded molecular computation, Ph.D. Thesis, Department of Computer Science, Yale University.

Braich, R.S., Johnson, C., Rothemund, P.W.K., Hwang, D., Chelyapov, N., Adleman, L.M., 1999. Solution of a satisfiability problem on a gel-based DNA computer. In: Proceedings of the sixth International Conference on DNA Computation in the Springer–Verlag Lecture Notes in Computer Science Series.

Kalim, M., Restricted genetic alphabet for DNA computing, In: Eric, B., Baum, Landweber, L.F., 1998. DNA Based Computers II: DIMACS Workshop, June 10–12, 1996, volume 44 of DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, Providence, RI, pp. 243–246.

Cukras, A.R., Faulhammer, D., Lipton, R.J., Landweber, L.F., 1998. Chess games: a model for RNA-based computation. In: Proceedings of the fourth DIMACS Meeting on DNA Based Computers, University of Pennsylvania, pp. 27–37.

Chang, W.-L., Guo, M., 2004. Using sticker for solving the dominating-set problem in the Adleman–Lipton Model. IEICE Trans. Inf. Syst. E-87D (7), 1782–1788.

Reif, J.H., LaBean, T.H., Seeman, 2000. Challenges and applications for self-assembled DNA-nanostructures. In: Proceedings of the Sixth DIMACS Workshop on DNA Based Computers, Leiden, Holland.

LaBean, T.H., Winfree, E., Reif, J.H., 2000. Experimental progress in computation by self-assembly of DNA tilings. Theor. Comput. Sci. 54, 123–140.

LaBean, T.H., Reif, J.H., 2001. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. Nature 407, 493–496.

Garzon, M.H., Deaton, R.J., 1999. Biomolecular Computing and Programming. IEEE Trans. Evolut. Comput. 3, 236–250.

Chang, W.-L., Guo, M., Ho, M., 2003. Solving the set-splitting problem in sticker-based model and the Adleman–Lipton Model. In: The 2003 International Symposium on Parallel and Distributed Processing and Applications, Aizu City, Japan, LNCS 2745, pp. 185–196.