Available online at www.sciencedirect.com
SCIENCE DIRECT®
Parallel Computing 30 (2004) 1109–1125
PARALLEL COMPUTING
www.elsevier.com/locate/parco

Fast parallel molecular solution to the dominating-set problem on massively parallel bio-computing

Minyi Guo [a,*], Michael (Shan-Hui) Ho [b,1], Weng-Long Chang [b,1]

[a] *Department of Computer Software, The University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan*
[b] *Department of Information Management, Southern Taiwan University of Technology, Tainan County, 710, R.O.C. Taiwan*

Received 5 April 2004; revised 1 July 2004; accepted 15 July 2004
Available online 25 September 2004

Abstract

This paper shows how to use DNA strands to construct solution space of molecules for the *dominating-set problem* and how to apply biological operations to solve the problem from the solution space of molecules. In order to achieve this, we have proposed some DNA based parallel algorithms using the operations in Adleman–Lipton model, together with the analysis of the computational complexity for DNA parallel algorithms.

Keywords: Parallel biological computing; DNA-based computing; NP-complete problem; Dominating-set problem; DNA based parallel algorithms

* Corresponding author. Tel.: 0242 37 2557; fax: 0242 37 2744.
E-mail addresses: minyi@u-aizu.ac.jp (M. Guo), michael@mail.stut.edu.tw (Michael (Shan-Hui) Ho), changwl@mail.stut.edu.tw (W.-L. Chang).
[1] Tel.: +886 6 2533131x4300; fax: +886 6 2541621.

0167-8191/$ - see front matter
doi:10.1016/j.parco.2004.07.014

1. Introduction

Through advances in molecular biology [1,2], it is now possible to produce 10^{18} or more DNA strands in tube. Those 10^{18} or more DNA strands can also be applied for representing 10^{18} or more bits of information. Biological operations can be used to simultaneously operate 10^{18} or more bits of information. Or we can say that 10^{18} or more data processors can be executed in parallel. Hence, it becomes obvious that biological computing can provide a very huge parallelism for dealing with problems in the real world. Especially, the problems from the NP-complete class are well-known to be exponentially more difficult than evaluating determinants whose entries are merely numerical. It is difficult to solve these kinds of problems even if very massive supercomputers are used when the problem sizes become large.

On the other hand, DNA computers have the full potential of the high performance computing technology. One test tube can be viewed as a processing unit like standard computer architecture. Furthermore, DNA algorithms using biological operations have natural parallelism because DNA strands are separated (melted, annealed) in test tubes in parallel.

Feynman [29] first proposed molecular computation in 1961, but his idea was not implemented by experiment for a few decades. In 1994 Adleman [2] succeeded to solve an instance of the Hamiltonian path problem in a test tube, just by handling DNA strands. Lipton [3] demonstrated that the Adleman techniques could be used to solve the satisfiability problem (the first NP-complete problem). Adleman and coworkers [14] proposed *sticker* for enhancing the Adleman–Lipton model.

In this paper, we use *sticker* to construct a solution space of DNA for the *dominating-set problem*. Simultaneously, we also apply DNA operations in the Adleman–Lipton model to develop a DNA algorithm. The results of the proposed algorithm show that the *dominating-set problem* is resolved with biological operations in the Adleman–Lipton model for solution space of sticker. Furthermore, this work presents clear evidence of the ability of DNA based computing to solve NP-complete problems.

The paper is organized as follows. Section 2 introduces the Adleman–Lipton model in detail then this model is compared with other models. Section 3 introduces a DNA algorithm for solving the dominating-set problem for the solution space of sticker. In Section 4, the experimental results of simulated DNA computing are given. Conclusions and future researching works are drawn in Section 5.

2. DNA model of computation

2.1. The Adleman–Lipton model

It is cited from [16] that DNA (*DeoxyriboNucleic Acid*) is the *molecule* that plays the main role in DNA based computing. 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 construc-

tion of DNA are deoxyribonucleotides, where each deoxyribonucleotide contains three components: *sugar*, *phosphate* group, and *nitrogenous* base. Sugar has five carbon atoms—for the sake of reference there is a fixed numbering to them. The base also has carbons, so to avoid confusion the carbons of the sugar are numbered 1′ to 5′ (rather than 1–5). 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.

Due to [1,16], distinct nucleotides are detected only from their bases, which come in two types: *purines* and *pyrimidines*. Purines include *adenine*and *guanine*, abbreviated A and G. Pyrimidines contain *cytosine* and *thymine*, abbreviated C and T. Since nucleotides are only distinguished from their bases, they are simply represented as A, G, C, or T, depending upon the type of base they have. The structure of a nucleotide, cited in [16], is simplified in Fig. 1. In this figure, B is one of the four possible bases (A, G, C, or T), P is the phosphate group, and the rest (of the "stick") is the sugar base (with its carbons enumerated 1′ through 5′).

It was indicated from [1,11,16] that nucleotides could link together in two ways. Firstly the 5′-phosphate group of one nucleotide is joined with the 3′-hydroxyl group of another forming a *phosphodiester* 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 is available for bonding, denoted as 3′ end. This gives the molecule *direction*, and we can talk about the direction of 5′ end to the 3′ end or 3′ end to the 5′ end. The second way is that the base of one nucleotide interacts with the base of another to form a *hydrogen* bond. This bonding is based on 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 J.D. Watson and F.H.C. Crick who deduced the famous double helix structure of DNA in 1953, and won the Nobel Prize for its discovery).

According to [1,11,16], a DNA strand is essentially a sequence (polymer) of four types of nucleotides detected by one of the bases they contain. Two single strands of DNA under appropriate conditions can form a double strand, if the respective bases are the Watson–Crick complements of each other—A matches T, and C matches G; also the 3′ end matches the 5′ end. The length of a single stranded DNA is the number of nucleotides comprising a single strand. Thus, if a single stranded DNA includes 20 nucleotides, we can 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 [1,11,16].)

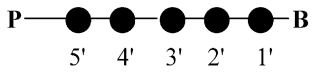

Fig. 1. A schematic representation of a nucleotide.

In the Adleman–Lipton model [2,3], *splints* were used to construct the corresponding edges of a particular graph of paths, which represented all possible binary numbers. As it stands, their construction indiscriminately builds all splints that lead to a complete graph. This is to say that hybridization has a higher probability of errors. Hence, Adleman and coworkers [14] proposed the sticker-based model, which was an abstract of molecular computing based on DNA with a random access memory as well as a new form of encoding the information

The DNA operations in the Adleman–Lipton model [2,3,11,12] are described below. These operations will be used for figuring out solutions of the dominating-set 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, produces 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 in the individual strands.
3. *Detect*. Given a tube P, we have 'yes' if P includes at least one DNA molecule, and we have '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. Comparison of the Adleman–Lipton model with other models

Techniques in the Adleman–Lipton model could be used to solve the NP-complete Hamiltonian path problem and satisfiability (SAT) problem by linearly increasing time and exponentially increasing volumes of DNA [2,3]. Quyang et al. [4] showed that restriction enzymes could be used to solve the NP-complete clique problem (MCP). The maximum number of vertices that can be processed is limited to 27 because the size of the pool with the size of the problem exponentially increases [4]. Arito et al. [5] described new molecular experimental techniques for searching a Hamiltonian path. Morimoto et al. [6] offered a solid-phase method to finding a Hamiltonian path. Narayanan and Zorbala [7] proved that the Adleman–Lipton model was extended towards solving the traveling salesman problem. Shin et al. [8] presented an encoding scheme that applies fixed-length codes for representing integer and real values. Their method could also be employed towards solving the traveling salesman problem. Amos [13] proposed a parallel filtering model for resolving the Hamiltonian path problem, the sub-graph isomorphism problem, the 3-ver-

tex-colorability problem, the clique problem and the independent-set problem. In our previous work, Chang and Guo [17–20,25] proved how the DNA operations for solution space of *splint* in the Adleman–Lipton model could be employed for developing DNA algorithms to resolve the dominating-set problem, the vertex cover problem, the clique problem, the independent-set problem, the three-dimensional matching problem, the set-packing problem, the set cover problem and the problem of exact cover by 3-sets.

Roweis et al. [14] proposed sticker-based model to enhance the Adleman–Lipton model. Their model could be used for determining solutions to the set cover problem. Perez-Jimenez et al. [15] employed sticker-based model [14] to resolve knapsack problems. Fu [21] proposed new algorithms to resolve 3-SAT, 3-Coloring and the independent set. In our previous work, Chang et al. [26–28] also employed the sticker-based model and the Adleman–Lipton model for dealing with the subset-sum problem, Cook's theorem [9,10] and the set-splitting problem for decreasing the error rate of hybridization.

3. Using sticker for solving the dominating-set problem in the Adleman–Lipton model

3.1. Definition of the dominating-set problem

Mathematically, a *dominating set* of a graph $G = (V, E)$, where V is the set of the vertex and E is the set of the edge, is a subset $V^1 \subseteq V$ of vertices such that for all $u \in V - V^1$ there is a $v \in V^1$ for which $(u, v) \in E$ [9,10]. The dominating-set problem is to find a *minimum size* dominating set in G. This has been proved to be a NP-complete problem [10].

The dominating-set problem asks: Given a network consisting of n vertices and m edges, how many vertices are in a *minimum size* dominating set? The graph includes three vertices and two edges as shown in Fig. 1, where each circle in the figure represents a vertex and the arc connecting two circles represents an edge. The *minimum size* dominating set for the graph in Fig. 2 is $\{v_1\}$. Hence, the size of the dominating-set problem in this graph is one. It is indicated from [10] that finding a minimum-size dominating-set is a NP-complete problem, so it can be formulated as a "search" problem. The dominating set problem is widely used in network routing, town

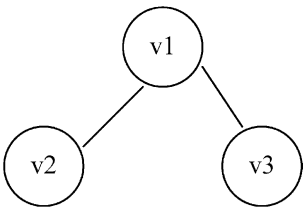

Fig. 2. Graph of our problem.

planning, and other real applications. Thus we use the problem as an example to show how powerful DNA computers are for solving NP-complete problem.

3.2. Using sticker for constructing solution space of DNA sequence for the dominating-set problem

In the Adleman–Lipton model, their main idea is to first generate solution space of DNA sequences for those problems resolved. Then, basic biological operations are used to select legal solutions and to remove illegal solutions from the solution space. Therefore, the first step of resolving the dominating-set problem is to produce a test tube, which contains all possible dominating sets. Assume that an undirected graph $G = (V, E)$, where V is the set of the vertices and E is the set of the edges. $|V|$ represents the number of the vertices in V and $|E|$ represents the number of the edges in E. Assume that $|V| = n$ and $|E| = m$. Assume that an n-digit binary number represents each possible dominating set within G. Also suppose that V^1 is a dominating set of G. If the ith bit in an n-digit binary number is set to 1, then it represents that the ith vertex in V^1 is also in G and not in $V - V^1$. If the ith bit in an n-digit binary number is set to 0, then it represents that the corresponding vertex is not in V^1 but is found in $V - V^1$. By doing this, all of the possible dominating sets in G are transformed into an ensemble of all n-digit binary numbers. Hence, Table 1 denotes the solution space for the graph in Fig. 1. The binary number 000 in Table 1 shows that the corresponding dominating set is empty. The binary numbers, 001, 010 and 011, in Table 1 shows that those corresponding dominating sets are $\{v_1\}$, $\{v_2\}$ and $\{v_2, v_1\}$ respectively. The binary numbers, 100, 101 and 110, in Table 1 shows that the corresponding dominating sets, subsequently, are $\{v_3\}$, $\{v_3, v_1\}$ and $\{v_3, v_2\}$. The binary number 111 in Table 1 shows that the corresponding dominating set is $\{v_3, v_2, v_1\}$. Though there are eight 3-digit binary numbers for representing eight possible dominating sets in Table 1, not every 3-digit binary number corresponds to a *legal* dominating set. Hence, in next subsection, basic biological operations are used to develop an algorithm for removing illegal dominating sets and determining legal dominating sets.

To implement this, assume that an unsigned value X is represented by a binary number $x_n, x_{n-1}, \ldots, x_1$, where the value of x_j is 1 or 0 for $1 \leqslant j \leqslant n$. The range of the value for X is from zero to 2^n–1. This is to say that it is formed by 2^n kinds

Table 1
Solution space for the graph in Fig. 1

3-Digit binary number	The corresponding dominating set
000	$\varnothing$
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	$\{v_3, v_2, v_1\}$

of possible values. Each possible value represents a dominating set for a graph G. Therefore an unsigned value X forms 2^n possible dominating sets. A bit x_i in an unsigned integer X represents the ith vertex in G. If the ith vertex is in a dominating set, then the value of x_i is set to 1. If the ith vertex is out of a dominating set, then the value of x_i is set to 0.

To represent all possible dominating sets for the dominating-set problem, *sticker* [14,22] is used to construct solution space for the problem solved. For every bit, x_i, two *distinct* 15 base value sequences were designed. One represents the value, 1, for x_i and the second represents the value, 0, to x_i. For the sake of convenience in our 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 zero. Each of the 2^n possible dominating sets was 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 [14,22] 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 [22].

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 As, 4 Ts, 4 Cs or 4 Gs occur in any library or probe sequences.
3. Every probe sequence has at least 4 mismatches with all 15 base alignments of any library sequence (except for its matching value sequence).
4. Every 15 base subsequence of a library sequence has at least 4 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 eigth base alignment of any library sequence (except for 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 [23]. Constraint (2) is motivated by two assumptions: first, those 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 [22] was modified for generating those DNA sequences to satisfy the constraints above. For example, for representing the three vertices in the graph in Fig. 1, the DNA sequences generated were:

$$x_1^0 = \text{AAAACTCACCCTCCT},\ x_2^0 = \text{TCTAATATAATTACT},$$
$$x_3^0 = \text{ATTCTAACTCTACCT},\ x_1^1 = \text{TTTCAATAACACCTC},$$
$$x_2^1 = \text{ATTCACTTCTTTAAT and } x_3^1 = \text{AACATACCCCTAATC}$$

Therefore, for every possible dominating set of the graph in Fig. 1, the corresponding library strand was synthesized by employing a mix-and-split combinatorial synthesis technique [24]. Similarly, for any n-vertex graph, all of the library strands for representing every possible dominating set could also be synthesized using the same technique.

3.3. The DNA algorithm for solving the dominating-set problem

The following DNA algorithm is proposed to solve the dominating-set problem.

Algorithm 1. Solving the dominating-set problem.

(1) Input (T_0), where tube T_0 includes solution space of DNA sequences to encode all of the possible dominating sets for any n-vertex graph, G, with those techniques mentioned in Section 3.2.
(2) Forall $i = 1$ to n, where n is the number of vertices in G.
  (a) $T_0 = +(T_0, x_i^1)$ and $R = -(T_0, x_i^1)$.
  (b) For each vertex V_j is adjacency to V_i.
    (c) $S = +(R, x_j^1)$ and $R = -(R, x_j^1)$.
    (d) $T_0 = \cup(T_0, S)$.

  EndFor.
  (e) Discard the tube R.

 EndForall.
(3) Forall $i = 0$ to $n - 1$.
 For $j = i$ down to 0.
  (a) $T_{j+1}^{\text{ON}} = +(T_j, x_{i+1}^1)$ and $T_j = -(T_j, x_{i+1}^1)$.
  (b) $T_{j+1} = \cup(T_{j+1}, T_{j+1}^{\text{ON}})$.
 EndFor
 EndForall
(4) For k = 1–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 dominating-set problem for any n-vertex graph can be solved.*

Proof. 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. The tube includes all 2^n possible dominating sets for any n-vertex graph, G. □

According to the definition of dominating set [9,10], Step 2(a) applies "extraction" to form two test tubes: T_0 and R. The first tube T_0 contains all of the strands that have $x_i = 1$. The second tube R consists of all of the strands that have $x_i = 0$. From the definition of dominating set, tube R represents sets, $V - V^1$ which do not include the vertex V_i. If there is no vertex adjacent to V_i, then Step 2(e) will discard tube R. This means that all of the illegal dominating sets in R will be removed. Otherwise, Steps 2(c) and 2(d) will be executed z times, where z is the number of vertices adjacent (directly connected by an edge) to V_i. Each time Step 2(c) is executed, it uses extraction to form two new test tubes: S and R. Tube S includes all of the strands that have $x_i = 0$ and $x_j = 1$. Tube R consists of all of the strands that have $x_i = 0$ and $x_j = 0$. It is indicated from the definition of the dominating set that tube S contains the strands, which represent those legal dominating sets. Therefore, Step 2(d) applies "merge" to pour tube S into tube T_0. After Steps 2(c)–(d) are repeated z times, tube T_0 includes the strands, which satisfy $(V_i, V_j) \in E$, where $V_i \in V^1$ and $V_j \in V - V^1$. Tube R contains the strands, which do not satisfy $(V_i, V_j) \in E$, where $V_i \in V^1$ and $V_j \in V - V^1$. Hence, Step 2(e) discards tube R. For other vertices in G, similar processing is also finished. Therefore, the remaining strands in tube T_0 represent legal dominating sets.

Each time outer loop of Step 3 is executed; the number of executions for the inner loop is $(i + 1)$ times. The first time the outer loop is executed; the inner loop is only executed once. Therefore, Steps 3(a) and 3(b) will also be executed once. Step 3(a) uses "extraction" to form two 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 the first tube encodes every dominating set including the first vertex and the second tube represents every dominating set not including the first vertex. Hence, Step 3(b) applies "merge" to pour tube T_1^{ON} into tube T_1. After repeating execution of Steps 3(a) and (b), it finally produces n *new* tubes. Tube T_k for $n \geqslant k \geqslant 1$ encodes those dominating sets that contain k vertices.

Because the dominating-set problem is to find a minimum-size dominating set, tube T_1 is detected with the "detection operation" from Step 4(a). If it returns a "yes", then tube T_1 contains dominating sets in which the number of vertices is minimum. Therefore, Step 4(b) uses "read operation" to describe the 'sequence' of a molecular in tube T_1 and terminates the algorithm. Otherwise, continue to repeat execution of Step 4(a) until a minimum-size dominating set is found in the tube detected.

The graph in Fig. 1 is used to show the power of Algorithm 1. From Step 1 of Algorithm 1 tube T_0 is filled with eight library strands using the techniques mentioned in Section 3.2. The eight library strands corresponds to eight possible dominating sets for the graph in Fig. 1. All of the edges in the graph of Fig. 1 are (V_1, V_2) and (V_1, V_3). The number of the vertices in the graph for Fig. 1 is three, so the number

of executions for Step 2 of Algorithm 1 is three times. According to the first execution of Step 2(a) of Algorithm 1, two tubes are generated. The first tube, T_0 includes those dominating sets: $\{V_1\}$, $\{V_2, V_1\}$, $\{V_3, V_1\}$ and $\{V_3, V_2, V_1\}$ and the second tube, R, contains dominating sets: $\varnothing$, $\{V_2\}$, $\{V_3\}$ and $\{V_3, V_2\}$. Because the number of the vertices adjacent to vertex V_1 is two, the number of executions for Step 2(b) of Algorithm 1 is two times. Due to the first execution of Step 2(c) of Algorithm 1, two tubes are yielded. The first tube S, includes dominating sets: $\{V_2\}$ and $\{V_3, V_2\}$ and the second tube R, contains dominating sets: $\varnothing$ and $\{V_3\}$. Tube S contains the legal dominating sets from definition of the dominating set. Hence, Step 2(d) of Algorithm 1 pours tube S into tube T_0. Now tube T_0 includes dominating sets: $\{V_1\}$, $\{V_2, V_1\}$, $\{V_3, V_1\}$, $\{V_3, V_2, V_1\}$, $\{V_2\}$ and $\{V_3, V_2\}$. According to the second execution of Step 2(c) of Algorithm 1, two tubes are produced. The first tube S, only consists of the dominating set: $\{V_3\}$ and the second tube R, only contains the dominating set: $\varnothing$. It is obvious from the definition of dominating set that tube S includes the legal dominating set. Hence, Step 2(d) of Algorithm 1 again pours tube S into tube T_0. Now tube T_0 includes dominating sets: $\{V_1\}$, $\{V_2, V_1\}$, $\{V_3, V_1\}$, $\{V_3, V_2, V_1\}$, $\{V_2\}$, $\{V_3, V_2\}$ and $\{V_3\}$. Since tube R contains the binary number 000, this does not satisfy the definition of a dominating set, the strand in R represents an illegal dominating set. Therefore, Step 2(e) of Algorithm 1 discards the tube R. The same processing can be applied to deal with the other two vertices V_2 and V_3 After every vertex is processed, the remaining strands in tube T_0 represent the legal dominating sets. That is to say that tube T_0 contains dominating sets: $\{V_1\}$, $\{V_2, V_1\}$, $\{V_3, V_1\}$, $\{V_3, V_2\}$ and $\{V_3, V_2, V_1\}$.

Because the number of vertices in the graph for Fig. 1 is three, the number of executions for the outer loop in Step 3 of Algorithm 1 is three times. The number of executions for the inner loop in Step 3 of Algorithm 1 is dependent on the value of the loop variable in the outer loop. After execution of Step 3(a) and (b) the first time tube T_1 contains dominating sets: $\{V_1\}$, $\{V_2, V_1\}$, $\{V_3, V_1\}$ and $\{V_3, V_2, V_1\}$ and tube T_0 only includes dominating set: $\{V_3, V_2\}$. After Step 3(a) and (b), it produces three *new* tubes. The three tubes T_1, T_2 and T_3 respectively, include $\{V_1\}$, $\{V_2, V_1\}$, $\{V_3, V_1\}$, $\{V_3, V_2\}\}$ and $\{\{V_3, V_2, V_1\}\}$.

Because tube T_1 is not empty, the "detection operation" for detecting tube T_1 in Step 4(a) in Algorithm 1 returns "yes". Therefore, Step 4(b) in Algorithm 1 reads the answer from tube T_1. Thus, a minimum-size dominating set for the graph in Fig. 1 is $\{V_1\}$.

3.4. The complexity of the proposed DNA algorithm

Theorem 3.2. *The dominating-set problem for any undirected n-vertex graph G can be solved with $O(n^2)$ biological operations in the Adleman–Lipton model.*

Proof. Algorithm 1 can be applied for solving the dominating-set problem for any undirected n-vertex graph G. Algorithm 1 includes three main steps. Step 2 is mainly used to determine the legal dominating sets and to remove any illegal dominating sets from all of the 2^n possible library strands. From Algorithm 1, it is very obvious that Step 2(a) takes n extraction operations and Step 2(e) takes n discarded opera-

tions. Since every vertex at most has $(n-1)$ adjacent vertices. Therefore, from Algorithm 1 Step 2(c) takes $n \times (n-1)$ extraction operations and Step 2(d) takes $n \times (n-1)$ merge operations. Step 3 is used to figure out the number of elements in every legal dominating set. It is indicated from Algorithm 1 that Step 3(a) takes $(n \times (n-1)/2)$ extraction operations and Step 3(b) takes $(n \times (n-1)/2)$ merge operations. Step 4 is used to find a minimum-size dominating set from legal dominating sets. It is pointed out from Algorithm 1 that Step 4(a) at most takes n detection operations and Step 4(b) takes one reading 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 dominating set problem for any undirected n-vertices graph G can be solved with sticker to construct $O(2^n)$ strands in the Adleman–Lipton model.*

Proof. Refer to Theorem 3.2. □

Theorem 3.4. *The dominating set problem for any undirected n-vertices graph G can be solved with $O(n)$ tubes in the Adleman–Lipton model.*

Proof. Refer to Theorem 3.2. □

Theorem 3.5. *The dominating set problem for any undirected n-vertices graph G can be solved with the longest library strand, $O(15 \times n)$, in the Adleman–Lipton model.*

Proof. Refer to Theorem 3.2. □

4. Experimental results of simulated DNA computing

We modified the Adleman program [22] using a Pentium II, 200 MHz CPU and 64 MB of main memory. The modified program was applied to generating DNA sequences for solving the dominating-set problem of any n-vertex graph. Because the source code of the two functions *srand*48( ) and *drand*48( ) was not found in the *original* Adleman program, we used the standard function *srand*( ) in C++ builder 6.0 to replace the function *srand*48( ) and added the source code to the function *drand*48( ). We also added subroutines to the Adleman program for simulating biological operations in the Adleman–Lipton model in Section 2. We added subroutines to the Adleman program to simulate Algorithm 1 in Section 3.3.

The Adleman program was used to construct each 15-base DNA sequence for every bit of the library. For each bit, the program generates two 15-base random sequences ('1' and '0') 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 so 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. 1. The graph includes three vertices: V_1, V_2 and V_3. DNA sequences generated by the modified Adleman program are shown in Table 2. This program took one mutation, one mutation and ten mutations to make the 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 used is shown in Table 3. Only G really matters to the energy of each bit. For example, delta G for the probe binding of '1' in the first bit is 24.3 kcal/mol and the delta G for the probe binding of '0' is 27.5 kcal/mol.

The program simulated a mix-and-split combinatorial synthesis technique [24] to synthesize the library strand to every possible dominating set. These library strands are shown in Table 4 and represent eight possible dominating sets: $\varnothing$, $\{V_1\}$, $\{V_2\}$, $\{V_2, V_1\}$, $\{V_3\}$, $\{V_3, V_1\}$, $\{V_3, V_2\}$ and $\{V_3, V_2, V_1\}$. The program also figured out the average and standard deviation for the enthalpy, entropy and free energy over all of the probe/library strand interactions. The energy levels are shown in Table 5. The standard deviation for delta G is small because it 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.

Table 2
Sequences chosen to represent the vertices in the graph in Fig. 1

Vertex	$5' \rightarrow 3'$ DNA sequence
x_3^0	ATTCTAACTCTACCT
x_2^0	TCTAATATAATTACT
x_1^0	AAAACTCACCCTCCT
x_3^1	AACATACCCCTAATC
x_2^1	ATTCACTTCTTTAAT
x_1^1	TTTCAATAACACCTC

Table 3
The energy for binding each probe to its corresponding region of 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 dominating sets

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′

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

The Adleman program was employed for computing the distribution of the different 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 was, 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. Hence, the number of matches peaks at 5 (225). That is to say that there are 225 occurrences where a probe matches a strand at five places.

It is indicated from the number of the vertices in the graph of Fig. 1, the number of simulations for Step 2 is three times. In Step 2(a) of the first simulation, two different arrays were used to store the results generated by Step 2(a). In Steps 2(c) and 2(d) of simulation, legal solutions were remained and merged in arrays. In Step 2(e) of simulation, all of the illegal solutions produced by Step 2(c) were discarded. Repeat the execution of Step 2 of simulation until all of the vertices in the graph in Fig. 1 were processed. Thus, the result generated by Step 2 is shown in Table 6.

Table 6
DNA sequences generated by Step 2 represent legal dominating sets

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

The goal of Step 3 in Algorithm 1 is to compute all of the legal dominating sets, which include how many vertices. Due to the number of the vertices shown in Fig. 1, the number of simulation for Steps 3(a) and 3(b) is six times. In Step 3(a) of the first simulation, two different arrays were used to store the results generated by Step 3(a). This means that the dominating sets in tube, T_1^{ON}, contain V_1 and the dominating sets in tube, T_0 do not include V_1. In Step 3(b) of the first simulation, those dominating sets in T_1^{ON} were merged into the tube T_1. Repeating the execution of Step 3 of the simulation until all of the vertices in the graph in Fig. 1 were processed. Therefore, the results generated by Step 3 are shown in Tables 7–9. The goal of Step 4 is to find a minimum-size dominating set. Because the number of the vertices is three, Step 4(a) was simulated no more than three times. The step was simulated by explicitly examining whether every tube generated in Step 3 was empty. The first

Table 7
DNA sequence represents that dominating set only containing one vertex

5′-ATTCTAACTCTACCTTCTAATATAATTACTTTTCAATAACACCTC-3′

Table 8
DNA sequences represent those dominating sets including two vertices

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

Table 9
DNA sequence represents that dominating set containing three vertices

5′-AACATACCCCTAATCATTCACTTCTTTAATTTTCAATAACACCTC-3′

Table 10
DNA sequence represents the minimum-size dominating set

5′-ATTCTAACTCTACCTTCTAATATAATTACTTTTCAATAACACCTC-3′

simulation for Step 4(a) generated a resulted value, *yes*, since the tube T_1 is not empty. Therefore Step 4(b) of simulation, the minimum-size dominating set from the tube T_1 was shown in Table 10.

5. Conclusions and future work

The present method for solving the dominating-set problem is based on biological operations in the Adleman–Lipton model and the solution space of stickers in the sticker-based model and thus is similar to the proposed method based on the solution space of splints to solving the same problem [17]. The proposed algorithm has three advantages from the Adleman–Lipton model and the sticker-based model. First, the proposed algorithm actually has a lower rate of errors for hybridization because we modified the Adleman program to generate good DNA sequences for constructing the solution space of stickers to the dominating-set problem. Only simple and fast biological operations in the Adleman–Lipton model were employed to solve the problem. Secondly, those biological operations in the Adleman–Lipton model had been performed in a fully automated manner in their lab. The full automation manner is essential not only for the speedup of computation but also for error-free computation. Thirdly, in the proposed algorithm the number of tubes, the longest length of DNA library strands and the number of DNA library strands, respectively, are $O(n)$, $O(15 \times n)$ and $O(2^n)$ strands. This implies that the proposed algorithm can be easily performed in a fully automated manner in a lab. Furthermore, the present algorithm generates 2^n library strands, which satisfies the seven constraints in Section 3.2, which corresponds to 2^n possible dominating sets. This allows the present algorithm to be applied to a larger instance of the dominating-set problem.

Currently, there are lots of NP-complete problems that cannot be solved because it is very difficult to support basic biological operations using mathematical operations. We are not sure whether molecular computing can be applied to dealing with every NP-complete problem. Therefore, in the future, our main work is to solve other NP-complete problems that were unresolved with the Adleman–Lipton model and the sticker model.

References

[1] R.R. Sinden, DNA Structure and Function, Academic Press, New York, 1994.
[2] L. Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (November) (1994) 1021–1024.
[3] R.J. Lipton, DNA solution of hard computational problems, Science 268 (1995) 542–545.
[4] Q. Quyang, P.D. Kaplan, S. Liu, A. Libchaber, DNA solution of the maximal clique problem, Science 278 (1997) 446–449.
[5] M. Arita, A. Suyama, M. Hagiya, A heuristic approach for Hamiltonian path problem with molecules, in: Proceedings of 2nd Genetic Programming (GP-97), 1997, pp. 457–462.

[6] N. Morimoto, M. Arita, A. Suyama, Solid phase DNA solution to the Hamiltonian path problem, DIMACS (Series in Discrete Mathematics and Theoretical Computer Science) 48 (1999) 93–206.
[7] A. Narayanan, S. Zorbala, DNA algorithms for computing shortest paths, in: J.R. Koza et al. (Eds.), Genetic Programming 1998: Proceedings of the Third Annual Conference, 1998, pp. 718–724.
[8] S.-Y. Shin, B.-T. Zhang, S.-S. Jun, Solving traveling salesman problems using molecular programming, in: Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), vol. 2, 1999, pp. 994-1000.
[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest. Introduction to algorithms, MIT Press, Cambridge, UK, ISBN 0-262-03141-8.
[10] M.R. Garey, D.S. Johnson, Computer and intractability, Freeman, San Fransico, CA, 1979.
[11] D. Boneh, C. Dunworth, R.J. Lipton, J. Sgall, On the computational power of DNA, in: Discrete Applied Mathematics, Special Issue on Computational Molecular Biology, vol. 71, 1996, pp. 79–94.
[12] L.M. Adleman, On constructing a molecular computer, DNA Based Computers, in: R. Lipton, E. Baum (Eds.), DIMACS series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1996, pp. 1–21.
[13] M. Amos, DNA Computation, Ph.D. Thesis, Department of Computer Science, The University of Warwick, 1997.
[14] S. Roweis, E. Winfree, R. Burgoyne, N.V. Chelyapov, M.F. Goodman, P.W.K. Rothemund, L.M. Adleman, A sticker based model for DNA computation, in: L. Landweber, E. Baum (Eds.), 2nd Annual Workshop on DNA Computing, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Princeton University, 1999, pp. 1–29.
[15] M.J. Perez-Jimenez, F. Sancho-Caparrini, Solving knapsack problems in a sticker based model, in: 7th Annual Workshop on DNA Computing, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 2001.
[16] G. Paun, G. Rozenberg, A. Salomaa, DNA Computing: New Computing Paradigms, Springer-Verlag, New York, 1998, ISBN: 3-540-64196-3.
[17] W.-L. Chang, M. Guo, Solving the dominating-set problem in Adleman–Lipton's Model, in: The Third International Conference on Parallel and Distributed Computing, Applications and Technologies, Japan, 2002, pp. 167–172.
[18] W.-L. Chang, M. Guo, Solving the clique problem and the vertex cover problem in Adleman–Lipton's model, in: IASTED International Conference, Networks, Parallel and Distributed Processing, and Applications, Japan, 2002, pp. 431–436.
[19] W.-L. Chang, M. Guo, Solving NP-complete problem in the Adleman–Lipton Model, in: The Proceedings of 2002 International Conference on Computer and Information Technology, Japan, 2002, pp. 157–162.
[20] W.-L. Chang, M. Guo, Resolving the 3-dimensional matching problem and the set packing problem in Adleman–Lipton's model, in: IASTED International Conference, Networks, Parallel and Distributed Processing, and Applications, Japan, 2002, pp. 455–460.
[21] B. Fu, Volume Bounded Molecular Computation, Ph.D. Thesis, Department of Computer Science, Yale University, 1997.
[22] R.S. Braich, C. Johnson, P.W.K. Rothemund, D. Hwang, N. Chelyapov, L.M. Adleman. Solution of a satisfiability problem on a gel-based DNA computer, in: Proceedings of the 6th International Conference on DNA Computation in the Springer-Verlag Lecture Notes in Computer Science series.
[23] K. Mir, A restricted genetic alphabet for DNA computing, in: E.B. Baum, L.F. Landweber (Eds.), DNA Based Computers II: DIMACS Workshop, June 10–2, 1996, volume 44 of DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, Providence, RI, 1998, pp. 243–246.
[24] A.R. Cukras, D. Faulhammer, R.J. Lipton, L.F. Landweber, Chess games: a model for RNA-based computation, in: Proceedings of the 4th DIMACS Meeting on DNA Based Computers, Held at the University of Pennsylvania, June 16–19, 1998, pp. 27–37.
[25] W.-L. Chang, M. Guo, Solving the set cover problem and the problem of exact cover by 3-sets in the AdlemanLipton model, Biosystems 72 (3) (2003) 263–275.
[26] W.-L. Chang, M.(S.-H.) Ho, M. Guo, Molecular solutions for the subset-sum problem on DNA-based supercomputing, Biosystems 73 (2) (2004) 117–130.

[27] M. Ho, W.-L. Chang, M. Guo, Is Cook's theorem correct for DNA-based computing—towards solving the NP-complete problems on a DNA-based supercomputer model, Journal of Parallel and Distributed Scientific and Engineering Computing, in press.
[28] W.-L. Chang, M. Guo, M. Ho, Solving the set-splitting problem in sticker-based model and the Adleman–Lipton model. Future Generation Computer System, in press.
[29] R.P. Feynman, in: D.H. Gilbert (Ed.), Minaturization, Reinhold Publishing Corporation, New York, 1961, pp. 282–296.