Using sticker to solve the 3-dimensional matching problem in molecular supercomputers

Minyi Guo*
Department of Computer Software,
University of Aizu,
Aizu-Wakamatsu City,
Fukushima 965 8580, Japan
E-mail: minyi@u-aizu.ac.jp
*Corresponding author

Weng-Long Chang
Department of Information Management,
Southern Taiwan University of Technology,
Tainan County, Taiwan, 710, ROC
E-mail: changwl@csie.ncku.edu.tw

Jiannong Cao
Department of Computing,
Hong Kong Polytechnic University,
Hong Kong
E-mail: csjcao@comp.polyu.edu.hk

Abstract: Adleman demonstrated that DNA (*Deoxyribonucleic acid*) strands could be applied for dealing with solutions to an instance of the NP-complete Hamiltonian path problem (HPP) (Adleman, 1994). The Adleman techniques could also be used to solve the NP-complete satisfiability (SAT) problem (the first NP-complete problem) (Lipton, 1995). Furthermore, *sticker* is used for enhancing the Adleman-Lipton model (Roweis et al., 1999). In this paper, we first use sticker to construct solution space of DNA library sequences for the 3-*dimensional matching problem*. Then, in the Adleman-Lipton model, we propose an algorithm to remove illegal solution and find legal solution for the 3-*dimensional matching problem* from solution space of sticker. Finally, a simulation result for our algorithm is generated.

Keywords: molecular supercomputing; DNA-based parallel algorithms; the NP-complete problem; parallel computing.

Reference to this paper should be made as follows: Guo, M., Chang, W-L. and Cao, J. (2004) ‘Using sticker to solve the 3-dimensional matching problem in molecular supercomputers’, *Int. J. High Performance Computing and Networking*, Vol. 1, Nos. 1/2/3, pp.128–139.

Biographical notes: Minyi Guo received PhD degree in information science from University of Tsukuba, Japan, in 1998. He is currently a Professor at the Department of Computer Software, The University of Aizu, Japan. From 2001 to 2003, he was a Visiting Professor at a couple of universities. He is the Editor-in-Chief of the *Journal of Embedded Systems* and is also in editorial board of many other journals. Guo’s research interests include parallel and distributed processing, parallelising compilers, data parallel languages, data mining, molecular computing and software engineering. He is a member of the ACM, IEEE, IEEE Computer Society, and IEICE.

Weng-Long Chang received the PhD degree in Computer Science and Information Engineering from National Cheng Kung University, Taiwan, Republic of China, in 1999. He is currently an Assistant Professor of Southern Taiwan University of Technology. His research interests include molecular computing, and languages and compilers for parallel computing.

Jiannong Cao received the BSc degree in Computer Science from Nanjing University, China, in 1982, and the MSc and PhD degrees from Washington State University, USA, in 1986 and 1990, all in Computer Science. Before joined the Hong Kong Polytechnic University in 1997, where he is currently an Associate Professor, he has been on faculty of Computer Science in James Cook University and The University of Adelaide in Australia, and the City University of Hong Kong. Dr. Cao's research interests include parallel and distributed computing, networking, mobile computing, fault tolerance, and distributed software architecture and programming. He has authored or coauthored over 120 journal and conference papers in the above areas. Dr. Cao has directed many funded research/teaching projects and participated in several research labs and centers. He is the Director of the Internet and Mobile Computing Lab in the Computing Department. Dr. Cao is a Vice Chairman and member of the Computer Architecture Professional Committee, China Computer Federation. He is a member of the IEEE Computer Society, the IEEE Communication Society, IEEE, and ACM. He has served as a member of editorial boards of several international journals, a reviewer for international journals/conference proceedings, and also as an organising/programme committee member for many international conferences.

1 INTRODUCTION

Nowadays, it is possible to produce roughly 10^{18} DNA strands that fit in a test tube through advances in molecular biology (Sinden, 1994). Those 10^{18} DNA strands can be applied for representing 10^{18} bit information. Basic biological operations can be used to simultaneously operate 10^{18} bit information. This is to say that there are 10^{18} data processors to be parallel executed. Hence, it is very obvious that biological computing can provide very huge parallelism for dealing with the problem in real world.

Adleman wrote the first paper in which it was demonstrated that DNA (*deoxyribonucleic acid*) 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 solving the NP-complete satisfiability (SAT) problem (the first NP-complete problem) (Lipton, 1995). Adleman and his co-authors 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 3-*dimensional matching problem*. Simultaneously, we also apply DNA operations in the Adleman-Lipton model to develop one DNA algorithm. The main result of the proposed DNA algorithm shows that the 3-*dimensional matching problem* is resolved with biological operations in the Adleman-Lipton model from solution space of sticker. Furthermore, this work represents obvious evidence for the ability of DNA-based computing to solve the NP-complete problem.

The rest of this paper is organised as follows. In Section 2, the Adleman-Lipton model is introduced in detail and the comparison of the model with other models is also given. In Section 3, the first DNA algorithm is proposed for solving the 3-*dimensional matching* 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 subsection 2.1, the summary of DNA structure and the Adleman-Lipton model is described in detail. In subsection 2.2, the comparison of the Adleman-Lipton model with other models is also introduced in detail.

2.1 The Adleman-Lipton model

A DNA (*deoxyribonucleic acid*) is a polymer, which is strung together from monomers called *deoxyribonucleotides* (Sinden, 1994; Paun et al., 1998). Distinct nucleotides are detected only with their bases, which come from *adenine*, *guanine*, *cytosine*, and *thymine*. Those bases are abbreviated as *A*, *G*, *C*, and *T*, respectively. A DNA strand is essentially a sequence (polymer) of four types of nucleotides detected by one of four bases they contain (Sinden, 1994; Boneh et al., 1996; Paun et al., 1998). 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 as 20 bp. (More discussion of the relevant biological background refers to Sinden, 1994; Boneh et al., 1996; Paun et al., 1998.)

In the Adleman-Lipton model (Adleman, 1994; Lipton, 1995), *splints* were used to construct to correspond to the edges of a particular graph, the paths of 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 hybridisation has higher probabilities of errors. Hence, Adleman and his co-authors (Roweis et al., 1999) proposed the sticker-based model, which was an abstract model of molecular computing based on DNA with a random access memory and a new form of encoding the information, to enhancing the Adleman-Lipton model.

The DNA operations proposed by Adleman and Lipton, cited from Adleman (1994), Lipton (1995), Boneh et al. (1996), and Adleman (1996), are described below. These operations will be used for calculating solutions of the 3-dimensional matching 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:

- *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 short strand S and $-(P, S)$ is all of the molecules of DNA in P which do not contain the short strand S.
- *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.
- *Detect*. Given a tube P, say 'yes' if P includes at least one DNA molecule, and say 'no' if it contains none.
- *Append*. Given a tube P and a short strand of DNA, Z, the operation will append the short strand, Z, onto the end of every strand in the tube P.
- *Discard*. Given a tube P, the operation will discard the tube P.
- *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

Techniques in the Adleman-Lipton model could be used to solve the NP-complete Hamiltonian path problem and satisfiability (SAT) problem in linearly increasing time and exponentially increasing volumes of DNA (Adleman, 1994; Lipton, 1995). Quyang et al. (1997) showed that restriction enzymes could be used to solve the NP-complete clique problem (MCP). 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).

Arita et al. (1997) described new molecular experimental techniques for searching a Hamiltonian path. Morimoto et al. (1999) offered a solid-phase method to finding a Hamiltonian path. Narayanan and Zorbala (1998) proved that the Adleman-Lipton model was extended towards solving the travelling salesman problem. Shin et al. (1999) presented an encoding scheme that applies fixed-length codes for representing integer and real values. Their method could also be employed towards solving the travelling salesman problem. Amos (1997) proposed parallel filtering model for resolving the Hamiltonian path problem, the sub-graph isomorphism problem, the 3-vertex-colourability problem, the clique problem, and the independent-set problem.

Roweis et al. (1999) proposed sticker-based model to enhance 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 resolve knapsack problems. Fu (1997) proposed new algorithms to resolve 3-SAT, 3-coloring, and the independent set. Winfree's self-assembling reactions for tiling fault tolerance in error-preventing codes and self-control of non-determinism and molecule formation and reaction efficiency were proposed in Winfree et al. (1998). Garzon and Deaton (1999). presented a review of the most important advances in molecular computing. In our previous work, Chang and Guo (2002a, 2002b) proved how the DNA operations from solution space of *splint* in the Adleman-Lipton model could be employed for developing DNA algorithms. Those DNA algorithms could be applied for resolving 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. In our previous work, Chang et al., (2003) also employed the sticker-based model and the Adleman-Lipton model to deal with the dominating-set problem for decreasing error rate of hybridisation.

3 USING STICKER FOR SOLVING THE 3-DIMENSIONAL MATCHING PROBLEM IN THE ADLEMAN-LIPTON MODEL

In subsection 3.1, the summary of the 3-dimensional matching problem is described. Applying sticker to constructing solution space of DNA sequences for the 3-dimensional matching problem is introduced in subsection 3.2. In subsection 3.3, one DNA algorithm is proposed to resolve the 3-dimensional matching problem. In subsection 3.4, the complexity of the proposed algorithm is described.

3.1 Definition of the 3-dimensional matching problem

Assume that W, X, and Y are disjoint sets having the same number q of elements. Suppose that W, X, and Y are $\{w_1, \ldots w_q\}$, $\{x_1, \ldots, x_q\}$ and $\{y_1, \ldots, y_q\}$, respectively. Assume that a finite set $C \subseteq W \times X \times Y$ and C is

$\{(w_k, x_l, y_m)|\ w_k \in W, x_l \in X$, and $y_m \in Y$ for $q \geq k$, l, and $m \geq 1\}$. Assume that $|C|$ denotes the number of elements in C and $|C| \geq t$, where t is a positive integer. A 3-*dimensional matching* for C is a subset $C^1 \subseteq C$ with $|C^1| \leq t$ such that no two elements of C^1 agree in any coordinate (Cormen et al., 2003; Garey and Johnson, 1979). The 3-dimensional matching problem is to find a *minimum-size* 3-dimensional matching for C. The problem has been proved to be an NP-complete problem (Garey and Johnson, 1979).

The 3-dimensional matching problem asks: Given four finite sets above, how many elements are in a *minimum-size* 3-dimensional matching? In Figure 1, three finite sets W, X, and Y are $\{1, 2\}$, $\{3, 4\}$ and $\{5, 6\}$, respectively, and a finite set $C \subseteq W \times X \times Y$ and $C = \{(1, 3, 5), (2, 4, 6), (1, 4, 6)\}$. The four sets, W, X, Y, and C denote such a problem. The *minimum-size* 3-dimensional matching for C is $\{(1, 3, 5), (2, 4, 6)\}$. Hence, the size of the 3-dimensional matching problem in Figure 1 is two. It is indicated from Garey and Johnson (1979) that finding a minimum-size 3-dimensional matching is an NP-complete problem, so it can be formulated as a 'search' problem.

$$W = \{1, 2\}, X = \{3, 4\}, Y = \{5, 6\} \text{ and } C \subseteq W \times X \times Y \text{ and}$$
$$C = \{(1, 3, 5), (2, 4, 6), (1, 4, 6)\}.$$

Figure 1 *The finite sets of our problem*

3.2 Using sticker for constructing solution space of DNA sequence for the 3-dimensional matching 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 remove illegal solution and find legal solution from solution space. Therefore, the first step of resolving the 3-dimensional matching problem is to produce a test tube, which contains all of the possible 3-dimensional matching. Assume that a set C_d with $\{w_k, x_l, y_m\}$ is a subset in C and it only contains three *order* elements. The first, second, and third elements in C_d come from W, X, and Y, respectively. Therefore, the finite set C can be regarded as a collection of subsets of three order elements, and C can be represented as $\{C_1, \ldots, C_B\}$, where C_d is $\{(w_k, x_l, y_m)|\ w_k \in W, x_l \in X$, and $y_m \in Y$ for $q \geq k$, l, and $m \geq 1\}$ for $B \geq d \geq 1$.

Suppose that a B-digit binary number represents all possible 2^B choices for subsets of three order elements. Also assume that $3 \times B$ *one*-digit binary numbers represent $3\times B$ elements in the subsets in C. Suppose that the $(3 \times d)$th, $(3 \times d + 1)$th and $(3 \times d + 2)$th one-digit binary numbers, respectively, correspond to the first element, the second element and the third element in the subset C_d for $B \geq d \geq 1$. That is to say that the three order elements in the subset C_d are represented with three continuous bits. Suppose that C^1 is 3-dimensional matching for C. If the dth bit in B bits is set to 0, then it represents the corresponding subset out of C^1 and the three order elements in the subset are excluded from C^1. That is to say that the $(3 \times d)$th, $(3 \times d + 1)$th, and $(3 \times d + 2)$th one-digit binary numbers are not appended onto the tail of those binary numbers, containing the value 0 of the dth bit. If the dth bit in B bits is set to 1, then it represents the corresponding subset in C^1 and the three order elements in the subset are included in C^1. Therefore, it is very obvious that the $(3 \times d)$th, $(3 \times d + 1)$th and $(3 \times d + 2)$th one-digit binary numbers, subsequently, are appended onto the tail of those binary numbers, containing the value 1 of the dth bit.

To implement this way, it is assumed that a B-bit binary number Z is represented as a binary number $z_1, \ldots, z_B$, where the value of z_j is 1 or 0 for $1 \leq j \leq B$. A bit z_j is the jth bit in a B-bit binary number Z and it represents the jth subset in C. Assume that $3 \times q$ one-digit binary numbers are, subsequently, $w_1, \ldots, w_q$, $x_1, \ldots, x_q$ and $y_1, \ldots, y_q$, where $q \geq B$. Assume that w_k, x_l and y_m represent the *kth* element in W, the lth element in X, and the mth element in Y, respectively. A B-bit binary number Z includes all possible 2^B choices of subsets. Each choice of subsets corresponds to a possible 3-dimensional matching. If the value of z_j is set to 1 and the corresponding subset consists of three order elements w_k, x_l, and y_m, then the value 1 for the three bits is, subsequently, appended onto the tail of those binary number, including the value 1 of the *jth* bit.

Consider the four finite sets in Figure 1. Three subsets, C_3, C_2, and C_1 represent $\{1, 3, 5\}$, $\{2, 4, 6\}$ and $\{1, 4, 6\}$, respectively, for C. Table 1 denotes solution space of 3-dimensional matching for the four finite sets in Figure 1. In Table 1, the binary number, 000, indicates that the corresponding 3-dimensional matching is empty. In Table 1, the binary numbers, 001, 010, and 011, represent that those corresponding 3-dimensional matching are $\{C_3\}$, $\{C_2\}$, and $\{C_2, C_3\}$, respectively. The binary numbers, 100, 101, and 110, in Table 1 represent that those corresponding 3-dimensional matching, subsequently, are $\{C_1\}$, $\{C_1, C_3\}$, and $\{C_1, C_2\}$. In Table 1, the binary number, 111, represents that the corresponding 3-dimensional matching is $\{C_1, C_2, C_3\}$. Though there are eight 3-digit binary numbers for representing eight possible 3-dimensional matching in Table 1, not every 3-digit binary number corresponds to a *legal* solution. Hence, in the following subsection, basic biological operations are used to develop an algorithm for removing illegal solutions and determining legal answers.

Table 1 *The solution space for the four finite sets in Figure 1*

3-digit binary number	*The corresponding 3-dimensional matching*
000	$\varnothing$
001	$\{C_3\}$
010	$\{C_2\}$
011	$\{C_2, C_3\}$
100	$\{C_1\}$
101	$\{C_1, C_3\}$
110	$\{C_1, C_2\}$
111	$\{C_1, C_2, C_3\}$

To represent all possible 3-dimensional matching for the 3-dimensional matching problem, *sticker* (Roweis et al., 1999; Braich et al., 2000) is used to construct solution space for that problem resolved. For every bit z_j representing the *jth* subset in *C*, two *distinct* 15 base value sequences were designed. One represents the value '1' of z_j and another represents the value '1' of z_j. For the sake of convenience of presentation, assume that z_j^1 denotes the value of z_j to be 1 and z_j^0 defines the value of z_j to be zero. Similarly, for every bit w_k, x_l, and y_m representing the *kth*, *lth*, and *mth* elements in *W*, *X* and *Y*, two *distinct* 15 base value sequences were also designed. One represents the value, 1, for w_k, x_l, and y_m and another represents the value, 0, to w_k, x_l, and y_m. For the sake of convenience of presentation, assume that w_k^1, x_l^1, and y_m^1 denote the value of w_k, x_l, and y_m to be 1 and w_k^0, x_l^0, and y_m^0 define the value of w_k, x_l, and y_m to be zero.

Each of the 2^B possible 3-dimensional matching was represented by a library sequence of $15 \times (B + 3 \times q)$ 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.

Errors in the separation of the library strands are errors in the computation (Roweis et al., 1999; Braich et al., 2000). Sequences must be designed to ensure that library strands have little secondary structure that might inhibit intended probe-library hybridisation. The design must also exclude sequences that might encourage unintended probe-library hybridisation. To help achieve these goals, sequences were computer-generated to satisfy the following constraint (Braich et al., 2000).

1 library sequences contain only As, Ts, and Cs
2 all library and probe sequences have no occurrence of more consecutive identical nucleotides
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 4 mismatches with all 15 base alignment of itself or any other library sequence
5 no probe sequence has a run of more than 7 matches with any 8 base alignment of any library sequence (except for with its matching value sequence)
6 no library sequence has a run of more than 7 matches with any 8 base alignments 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 (Mir, 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., 2000) was modified for generating those DNA sequences to satisfying the constraints above. For example, for representing the three subsets in *C* in Figure 1, the DNA sequences generated were: $z_1^0 = AATTCACAAACAATT$, $z_2^0 = ACTCCTTCCCTACTC$, $z_3^0 = TCTCTCTCTAATCAT$, $z_1^1 = TCTCCCTATTTATTT$, $z_2^1 = TCACCAAACCTAAAA$, and $z_3^1 = CCATCATCTACCTTA$. Similarly, for representing every element in *W*, *X*, and *Y* in Figure 1, the DNA sequences generated were: $w_1^0 = ACTCACATACACCAC$, $w_2^0 = CTTCTCCACTATACT$, $x_1^0 = AAACTATCATACTTC$, $x_2^0 = TTCAATAAACATTTT$, $y_1^0 = TTTTTCTCTCCCAAA$, $y_2^0 = TTTACCCTACTATCA$, $w_1^1 = CAACCTATTATCTTA$, $w_2^1 = CCTAAATCTCCAATA$, $x_1^1 = CTCTCAACAATCAAA$, $x_2^1 = ACCTCCTTAACACTT$, $y_1^1 = CCCTATCACTAATAC$ and $y_2^1 = TATAACCCATCCATA$. For every possible 3-dimensional matching to *C* in Figure 1, the corresponding library strand was synthesized by employing a mix-and-split combinatorial synthesis technique (Cukras et al., 1998). Similarly, for any *n*-subset set, all of the library strands for representing every possible 3-dimensional matching could be also synthesised with the same technique.

3.3 The DNA algorithm of solving the 3-dimensional matching problem

The following DNA algorithm is proposed to solve the 3-dimensional matching problem.

Algorithm 1: Solving the 3-dimensional matching problem.

```
(1) Input ($T_0$), where the tube $T_0$ is to encode all possible $2^B$ choices of subsets of three order elements for three finite sets $W = \{w_1, \ldots, w_q\}$, $X = \{x_1, \ldots, x_q\}$, and $Y = \{y_1, \ldots, y_q\}$ and a collection $C = \{C_1, \ldots, C_B\}$, where $C_d = \{(w_k, x_l, y_m) | w_k \in W, x_l \in X$, and $y_m \in Y$ for $q \geq k, l$, and $m \geq 1\}$ for $B \geq d \geq 1$.
(2) Forall $i = 1$ to $B$, where $B$ is the number of subsets in $C$.
    (2a) $T_{ON} = +(T_0, z_i^1)$ and $T_{OFF} = -(T_0, z_i^1)$.
        Assume that $(w_k, x_l, y_m)$ is the element in the subset, $C_i$.
    (2b) $T_{BAD} = +(T_{ON}, w_k^1)$ and $T_{ON} = -(T_{ON}, w_k^1)$.
    (2c) Discard the tube $T_{BAD}$.
    (2d) Append($T_{ON}$, $w_k^1$).
    (2e) $T_{BAD} = +(T_{ON}, x_l^1)$ and $T_{ON} = -(T_{ON}, x_l^1)$.
    (2f) Discard the tube $T_{BAD}$.
    (2g) Append($T_{ON}$, $x_l^1$).
    (2h) $T_{BAD} = +(T_{ON}, y_m^1)$ and $T_{ON} = -(T_{ON}, y_m^1)$.
    (2i) Discard the tube $T_{BAD}$.
```

```
        (2j) Append($T_{ON}$, $y_m^1$).
        (2k) $T_0 = \cup(T_{ON}, T_{OFF})$.
    End Forall
    (3) Forall $j$ = 1 to $q$
        (3a) $T_0 = +(T_0, w_j^1)$ and $T_{BAD} = -(T_0, w_j^1)$.
        (3b) Discard the tube $T_{BAD}$.
        (3c) $T_0 = +(T_0, x_j^1)$ and $T_{BAD} = -(T_0, x_j^1)$.
        (3d) Discard the tube $T_{BAD}$.
        (3e) $T_0 = +(T_0, y_j^1)$ and $T_{BAD} = -(T_0, y_j^1)$.
        (3f) Discard the tube $T_{BAD}$.
    End Forall
    (4) Forall $i$ = 0 to $B - 1$
        For $j = i$ down to 0
            (a) $T_{j+1}^{ON} = +(T_j, z_{i+1}^1)$ and $T_j = -(T_j, z_{i+1}^1)$.
            (b) $T_{j+1} = \cup(T_{j+1}, T_{j+1}^{ON})$.
        EndFor
    End Forall
    (5) For $d$ = 1 to $B$
        (a) If (detect ($T_d$) = 'yes') then
                (b) Read ($T_d$) and terminate the algorithm.
        EndIf
    End For
```

Theorem 3-1: *In light of those steps in Algorithm 1, the 3-dimensional matching problem for finite sets W, X, and Y and B-subset set C can be resolved.*

Proof: A test tube of DNA strands, which represent all possible 2^B input bit sequences $z_1, \ldots, z_B$, is yielded in Step (1). It is very obvious that the test tube contains all possible 2^B choices of 3-dimensional matching.

From definition of 3-dimensional matching (Garey and Johnson, 1979), Step (2) will be executed B times for representing three order elements in each subset. The first execution of Step (2a) uses 'extraction' operation to form two test tubes: T_{ON} and T_{OFF}. The first tube T_{ON} consists of all of the strands that have $z_i = 1$. That is to say that the first subset occurs in the tube T_{ON}. The second tube T_{OFF} consists of all of the strands that have $z_i = 0$. This is to say that the first subset does not appear in the tube T_{OFF}. Obviously, from the definition of 3-dimensional matching, the number of order elements in each subset in C is all three. The first, second, and three elements are from W, X, and Y, respectively. Therefore, Steps (2b) to (2j) are used to represent all of three order elements in each subset. The first execution of Step (2b) applies 'extraction' operation to generate two tubes: T_{BAD} and T_{ON}. The first tube T_{BAD} includes all of the strands that have $w_k = 1$. That is to say that the first element in the first subset appears repeatedly in other subsets in the tube T_{BAD}. From the definition of 3-dimensional matching (Garey and Johnson, 1979), the first tube T_{BAD} includes illegal choices of 3-dimensional matching. Hence, the first tube T_{BAD} is discarded in Step (2c). The second tube T_{ON} includes all of the strands that have $w_k = 0$. That is to say that the first element in the first subset does not occur in the tube T_{ON}. Step (2d) uses 'append' operation to append the short strand, w_k^1, representing the first element in the first subset, onto the end of every strand in the tube T_{ON}. Hence, the tube T_{ON} now contains the first element in the first subset. Similarly, the first execution of Step (2e) applies 'extraction' operation to generate two tubes: T_{BAD} and T_{ON}. The first tube T_{BAD} includes all of the strands that have $x_l = 1$. That is to say that the second element in the first subset appears repeatedly in other subsets in the tube T_{BAD}. Due to definition of 3-dimensional matching, the first tube T_{BAD} includes illegal choices of 3-dimensional matching. Hence, the first tube T_{BAD} is discarded in Step (2f). The second tube T_{ON} includes all of the strands that have $x_l = 0$. That is to say that the second element in the first subset does not occur in the tube T_{ON}. From the definition of 3-dimensional matching, Step (2g) uses 'append' operation to append the short strand, x_l^1, representing the second element in the first subset, onto the end of every strand in the tube T_{ON}. Hence, the tube T_{ON} now contains the second element in the first subset. Then, the first execution of Step (2h) applies 'extraction' operation to generate two tubes: T_{BAD} and T_{ON}. The first tube T_{BAD} includes all of the strands that have $y_m = 1$. That is to say that the third element in the first subset occurs repeatedly in other subsets in the tube T_{BAD}. It is indicated from the definition of 3-dimensional matching that the first tube T_{BAD} includes illegal choices of 3-dimensional matching. Hence, the first tube T_{BAD} is discarded in Step (2i). The second tube T_{ON} includes all of the strands that have $y_m = 0$. That is to say that the third element in the first subset does not occur in the tube T_{ON}. Because of definition of 3-dimensional matching, Step (2j) uses 'append' operation to append the short strand, y_m^1, representing the third element in the first subset, onto the end of every strand in the tube T_{ON}. Thus, the tube T_{ON} now contains the third element in the first subset. This is to say that all of the elements in the first subset are included in T_{ON}. Therefore, the first execution of Step (2k) applies 'merge' operation to pour two tubes T_{ON} and T_{OFF} into the tube T_0. The tube T_0 includes every element in the first subset. Similarly, after other $(B - 1)$ times to Step (2) are executed, every order element in each subset is represented in the corresponding DNA sequences in the tube T_0.

Step (3) is used for checking whether DNA strands in the tube T_0 exactly represent every element in finite sets W, X, and Y. Since the number of elements in finite sets W, X, and Y is all q, Step (3) will be executed q times for finding correct choices of 3-dimensional matching. The first execution of Step (3a) applies 'extraction' operation to check which subsets include the first element in W and which subsets do not contain the first element in W. Therefore, two tubes are generated. The second tube T_{BAD} includes illegal choices of subsets and therefore the tube T_{BAD} is discarded in Step (3b). Similarly, the first execution of Step (3c) applies 'extraction' operation to check which subsets contain the first element in X and which subsets do not consist of the first element in X. Two tubes are generated. The second tube T_{BAD} includes illegal choices of subsets and therefore the tube T_{BAD} is discarded in Step (3d). Then, the first execution of Step (3e) applies 'extraction' operation to check which subsets consists of the first element in Y and which subsets do not include the first element in Y. Two tubes are generated. Due to the definition of 3-dimensional matching, the second tube T_{BAD} includes illegal choices of subsets and therefore the tube T_{BAD} is discarded in Step (3f). Similarly, after other $(q-1)$ times for Step (3) are executed, every element in W, X, and Y is represented in the corresponding DNA sequences in the tube T_0. That is to say that the remaining DNA strands in the tube T_0 represent legal choices of 3-dimensional matching.

When each time of the outer loop in Step (4) is executed, the number of execution for the inner loop is $(i+1)$ times. At the first execution of the outer loop, the inner loop is only executed one time. Therefore, Steps (4a) and (4b) are executed one time. Step (4a) 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 $z_1 = 1$. The second tube T_0 consists of all of the strands that have $z_1 = 0$. That is to say that the first tube encodes every 3-dimensional matching with the first subset in C and the second tube represents every 3-dimensional matching without the first subset in C. Then, Step (4b) applies 'merge' operation to pour the tube T_1^{ON} into the tube T_1. After repeat to execute Steps (4a) and (4b), it finally produces B *new* tubes. The tube T_d for $B \geq d \geq 1$ encodes those DNA strands that contain d subsets.

Because the 3-dimensional matching problem is to find a minimum-size 3-dimensional matching, the tube T_1 first is detected with 'detection' operation in Step (5a). If it returns 'yes', then the tube T_1 contains minimum-size 3-dimensional matching. Therefore, Step (5b) uses 'read' operation for describing 'sequence' of a molecular in the tube T_1 and terminating the algorithm. Otherwise, repeat to execute Step (5a) until a minimum-size 3-dimensional matching is found in the tube detected.

The sets in Figure 1 can be applied for showing the power of Algorithm 1. In Figure 1, three finite sets W, X, and Y are {1, 2}, {3, 4}, and {5, 6}, respectively. A finite set $C \subseteq W \times X \times Y$ and $C = \{(1, 3, 5), (2, 4, 6), (1, 4, 6)\}$. Every triple element in C can be regarded as a 3-order-element subset. Therefore, the finite set C can be regarded as a collection of subsets of three order elements. Assume that the collection C is $\{C_1, C_2, C_3\}$, where C_1, C_2 and C_3 are {1, 3, 5}, {2, 4, 6}, and {1, 4, 6}, respectively. From Step (1) in Algorithm 1, the tube T_0 is filled with eight library stands with those techniques mentioned in subsection 3.2, representing eight possible 3-dimensional matching for W, X, Y, and C.

Because the number of the subsets in the collection C is three, the number of execution to Step (2) of Algorithm 1 is three times. At the first execution of Step (2a) in Algorithm 1, two tubes are yielded. The first tube, T_{ON}, includes the numbers 1** (* can be either 1 or 0). The second tube, T_{OFF}, contains the numbers 0**. That is to say that the first tube, T_{ON}, includes $\{C_1\}$, $\{C_1, C_3\}$, $\{C_1, C_2\}$, and $\{C_1, C_2, C_3\}$ and the second tube, T_{OFF}, contains Φ, $\{C_3\}$, $\{C_2\}$, and $\{C_2, C_3\}$. Due to the first execution to Step (2b) of Algorithm 1, two tubes are generated. The first tube, T_{BAD}, contains all of the strands that have $w_1 = 1$. That is to say that the first element, 1, in C_1 appears repeatedly in the tube T_{BAD}. It is indicated from the definition of 3-dimensional matching that the tube T_{BAD} includes illegal choices. Therefore, the tube T_{BAD} is discarded in Step (2c). The second tube T_{ON} contains all of the strands that have $w_1 = 0$. This is to say that the first element, 1, in C_1 does not occur in the tube T_{ON}. Step (2d) of Algorithm 2 uses 'append' operation to append DNA sequences of representing 1 onto the tube T_{ON}. The tube T_{ON} now contains the first element in C_1. This is to say that the tube T_{ON} consists of the strands representing those bit strings: 1001$(w_1 = 1)$, 1011$(w_1 = 1)$, 1101$(w_1 = 1)$ and 1111$(w_1 = 1)$.

Similarly, the first execution of Step (2e) applies 'extraction' operation to generate two tubes: T_{BAD} and T_{ON}. The first tube T_{BAD} includes all of the strands that have $x_1 = 1$. That is to say that the second element, 3, in C_1 appears repeatedly in other subsets in the tube T_{BAD}. The first tube T_{BAD} includes illegal choices. Hence, the first tube T_{BAD} is discarded in Step (2f). The second tube T_{ON} includes all of the strands that have $x_1 = 0$. That is to say that the second element, 3, in C_1 does not occur in the tube T_{ON}. Step (2g) uses 'append' operation to append DNA sequences of representing 3 onto the end of every strand in the tube T_{ON}. The tube T_{ON} now contains also the second element in C_1. That is to say that the tube T_{ON} contains the strands representing those bit strings: 10011$(w_1 = 1)(x_1 = 1)$, 10111$(w_1 = 1)(x_1 = 1)$, 11011$(w_1 = 1)(x_1 = 1)$ and 11111 $(w_1 = 1)(x_1 = 1)$.

Then, the first execution of Step (2h) applies 'extraction' operation to generate two tubes: T_{BAD} and T_{ON}. The first tube T_{BAD} includes all of the strands that have $y_1 = 1$. That is to say that the third element, 5, in C_1 appears repeatedly in other subsets in the tube T_{BAD}. That is to say, the first tube T_{BAD} includes illegal choices. Thus, the first tube T_{BAD} is discarded in Step (2i). The second tube T_{ON} includes all of the strands that have $y_1 = 0$. That is to say that the third element, 5, in C_1 does not occur in the tube T_{ON}. Step (2j) uses 'append' operation to append DNA sequences of representing 5 onto the end of every strand in the tube T_{ON}. The tube T_{ON} now contains the third element in C_1. This is to say that the tube T_{ON} includes the strands representing those bit strings: 100111$(w_1 = 1)(x_1 = 1)(y_1 = 1)$, 101111 $(w_1 = 1)(x_1 = 1)(y_1 = 1)$, 110111$(w_1=1)(x_1=1)(y_1=1)$, and 111111$(w_1=1)(x_1=1)(y_1=1)$. Then, the first execution of Step (2k) applies 'merge' operation to pour two tubes T_{ON} and T_{OFF} into the tube T_0. The tube T_0 now includes the strands representing those bit strings: 000, 001, 010, 011, 100111$(w_1 = 1)(x_1 = 1)(y_1 = 1)$, 101111$(w_1 = 1)(x_1 = 1)$ $(y_1 = 1)$, 110111$(w_1 = 1)(x_1 = 1)(y_1 = 1)$,and 111111$(w_1 = 1)$ $(x_1 = 1)(y_1 = 1)$. Similarly, after other two times to Step (2) are executed, the tube T_0 contains the strands representing those bit strings: 000, 001111$(w_1 = 1)(x_2 = 1)$ $(y_2 = 1)$, 010111$(w_2 = 1)(x_2 = 1)(y_2 = 1)$, 100111$(w_1 = 1)$ $(x_1 = 1)(y_1 = 1)$, and 110111111$(w_1 = 1)(x_1 = 1)(y_1 = 1)$ $(w_2 = 1)(x_2 = 1)(y_2 = 1)$.

From the definition of 3-dimensional matching, finding a 3-dimensional matching for W, X, Y, and C is to check whether every element in W, X, and Y appears exactly in chosen subsets. Because the number of the elements in W, X, and Y is all 2, Step (3) of Algorithm 1 will be executed two times. The first execution of Step (3a) in Algorithm 1 results in that two tubes are generated. The first tube T_0 contains those bit strings: $001111(w_1 = 1)(x_2 = 1)(y_2 = 1)$, $100111(w_1 = 1)(x_1 = 1)(y_1 = 1)$, and $110111111(w_1 = 1)(x_1 = 1)(y_1 = 1)(w_2 = 1)(x_2 = 1)(y_2 = 1)$. The second tube T_{BAD} includes these bit strings: 000 and $010111(w_2 = 1)(x_2 = 1)(y_2 = 1)$. This is to imply that 3-dimensional matching in the second tube T_{BAD} is all illegal choices. Therefore, Step (3b) is used to discard the second tube T_{BAD}. Similarly, The first execution of Step (3c) in Algorithm 1 indicates that two tubes are produced. The first tube T_0 contains those bit strings: $100111(w_1 = 1)(x_1 = 1)(y_1 = 1)$ and $110111111(w_1 = 1)(x_1 = 1)(y_1 = 1)(w_2 = 1)(x_2 = 1)(y_2 = 1)$. The second tube T_{BAD} consists of the bit string: $001111(w_1 = 1)(x_2 = 1)(y_2 = 1)$. That is to say that 3-dimensional matching in the second tube T_{BAD} is an illegal choice. Therefore, Step (3d) is used to discard the second tube T_{BAD}. Similarly, the first execution of Step (3e) in Algorithm 1 represents that two tubes are yielded. The first tube T_0 includes $100111(w_1 = 1)(x_1 = 1)(y_1 = 1)$ and $110111111(w_1 = 1)(x_1 = 1)(y_1 = 1)(w_2 = 1)(x_2 = 1)(y_2 = 1)$. The second tube T_{BAD} does not consists of any bit string. Step (3f) is used to discard the second tube T_{BAD}. The similar processing can be applied to deal with the second element in W, X, Y. After all of operations in Step (3) are processed, the remaining strands in the tube T_0 represent legal 3-dimensional matching, $110111111(w_1 = 1)(x_1 = 1)(y_1 = 1)(w_2 = 1)(x_2 = 1)(y_2 = 1)$.

The first execution of Step (4a) in Algorithm 1 applies 'extraction' operation to produce two tubes, T_1^{ON} and T_0. The tube T_1^{ON} contains the bit string, $110111111(w_1 = 1)(x_1 = 1)(y_1 = 1)(w_2 = 1)(x_2 = 1)(y_2 = 1)$. The tube T_0 does not include any bit string. Step (4b) applies 'merge' operation to pour the tube T_1^{ON} into the tube T_1. The tube T_1 includes the bit string, $110111111(w_1 = 1)(x_1 = 1)(y_1 = 1)(w_2 = 1)(x_2 = 1)(y_2 = 1)$, representing a 3-dimensional matching, $\{C_1, C_2\}$. After repeat to execute Steps (4a) and (4b), it finally produces three *new* tubes. The new tube T_d for $3 \geq d \geq 1$ encodes the 3-dimensional matching that contains d subsets. The tube T_2 contains the bit string, $110111111(w_1 = 1)(x_1 = 1)(y_1 = 1)(w_2 = 1)(x_2 = 1)(y_2 = 1)$. The tubes T_1 and T_3 do not include any bit string.

The 3-dimensional matching problem is to find a minimum-size 3-dimensional matching. Therefore, Step (5) is used to find a minimum-size 3-dimensional matching. Because the number of the element in C is three, Step (5a) at most will be executed three times. When the first time of Step (5a) is executed, the tube T_1 is first detected with 'detection' operation. 'Detection' operation for the tube T_1 returns 'no'. That is to say that the tube T_1 does not contains a minimum-size 3-dimensional matching. Next, the second time of Step (5a) is executed, the tube T_2 is detected with 'detection' operation. 'Detection' operation for the tube T_2 returns 'yes'. Therefore, Step (5b) applies 'read' operation to describe 'sequence' of a molecular in the tube T_2 and terminates the algorithm. A minimum-size 3-dimensional matching is found to be $\{(1, 3, 5), \{2, 4, 6\}\}$.

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 tubes used in Algorithm 1, and the longest library strand in solution space in Algorithm 1.

Theorem 3-2: *Suppose that W, X, and Y are disjoint sets which have the same number q of elements. Assume that W, X, and Y are $\{w_1, \ldots, w_q\}$, $\{x_1, \ldots, x_q\}$, and $\{y_1, \ldots, y_q\}$, respectively. Suppose that $C \subseteq W \times X \times Y$ and C is $\{C_1, \ldots, C_B\}$ where C_d is $\{w_k, x_l, y_m \mid w_k \in W, x_l \in X$, and $y_m \in Y$ for $q \geq k, l$, and $m \geq 1\}$ for $1 \leq d \leq B$. The 3-dimensional matching problem for W, X, Y and C can be resolved with $O(B^2)$ biological operations in the Adleman-Lipton model.*

Proof: Algorithm 1 can be applied for solving the 3-dimensional matching problem for any B-subset set C. Algorithm 1 includes four main steps. Step 2 is mainly used to construct DNA sequences for every element in each subset in C and to remove illegal library strands from all of the 2^B possible library strands. From Algorithm 1, it is very obvious that Step (2a) to Step (2k) totally take $4 \times B$ 'extraction' operations, $3 \times B$ 'discard' operations, $3 \times B$ 'append' operations and B 'merge' operations. Step (3) is mainly applied to check which library strands exactly include every element in W, X, and Y. It is indicated from Algorithm 1 that Step (3a) to Step (3f) take $3 \times B$ 'extraction' operations and $3 \times B$ 'discard' operations. Step (4) is mainly applied to figure out the number of element in every legal 3-dimensional matching. Algorithm 1 indicates that Step (4a) takes $(B \times (B - 1)/2)$ 'extraction' operations and Step (4b) takes $(B \times (B - 1)/2)$ 'merge' operations. Step 5 is used to find a minimum-size 3-dimensional matching from legal 3-dimensional matching. According to Algorithm 1, Step (5a) at most takes B 'detection' operations and Step (5b) takes one 'read' operation. Hence, from the statements mentioned above, it is at once inferred that the time complexity of Algorithm 1 is $O(B^2)$ biological operations in the Adleman-Lipton model.

Theorem 3-3: *Suppose that W, X, and Y are disjoint sets which have the same number q of elements. Assume that W, X, and Y are $\{w_1, \ldots, w_q\}$, $\{x_1, \ldots, x_q\}$, and $\{y_1, \ldots, y_q\}$, respectively. Suppose that $C \subseteq W \times X \times Y$ and C is $\{C_1, \ldots, C_B\}$ where C_d is $\{w_k, x_l, y_m \mid w_k \in W, x_l \in X$, and $y_m \in Y$ for $q \geq k, l$, and $m \geq 1\}$ for $1 \leq d \leq B$. The 3-dimensional matching problem for W, X, Y and C can be resolved with $O(2^B)$ library strands in the Adleman-Lipton model.*

Proof: Refer to Theorem 3-2.

Theorem 3-4: *Suppose that W, X, and Y are disjoint sets that have the same number q of elements. Assume that W, X, and Y are $\{w_1, \ldots, w_q\}$, $\{x_1, \ldots, x_q\}$,* *and $\{y_1, \ldots, y_q\}$, respectively. Suppose that $C \subseteq W \times X \times Y$ and C is $\{C_1, \ldots, C_B\}$ where C_d is $\{w_k, x_l, y_m \mid w_k \in W, x_l \in X$, and $y_m \in Y$ for $q \geq k, l$, and $m \geq 1\}$ for $1 \leq d \leq B$. The 3-dimensional matching problem for W, X, Y and C can be resolved with O(B) tubes in the Adleman-Lipton model.*

Proof: Refer to Theorem 3-2.

Theorem 3-5: *Suppose that W, X, and Y are disjoint sets which have the same number q of elements. Assume that W, X, and Y are $\{w_1, \ldots, w_q\}$, $\{x_1, \ldots, x_q\}$, and $\{y_1, \ldots, y_q\}$, respectively. Suppose that $C \subseteq W \times X \times Y$ and C is $\{C_1, \ldots, C_B\}$ where C_d is $\{w_k, x_l, y_m \mid w_k \in W, x_l \in X$, and $y_m \in Y$ for $q \geq k, l$, and $m \geq 1\}$ for $1 \leq d \leq B$. The 3-dimensional matching problem for W, X, Y and C can be resolved with the longest library strand, $O(15 \times B + 15 \times 3 \times q)$, in the Adleman-Lipton model.*

Proof: Refer to Theorem 3-2.

4 EXPERIMENTAL RESULTS OF SIMULATED DNA COMPUTING

We finished the modification of the Adleman program (Braich et al., 2000) in a PC with one Pentium(R) 4 and 128 MB main memory. Our operating system is Window 98 and our compiler is C++ Builder 6.0. This program modified was applied to generate DNA sequences for solving the 3-dimensional matching problem. Because the source code of the two functions *srand48*() and *drand48*() was not found in the *original* Adleman program, we used 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 added subroutines to the Adleman program to simulating Algorithm 1 in subsection 3.3.

The Adleman program is used to 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 the ‘1’ and the ‘0’) and checking to see if the library strands satisfy the seven constraints in subsection 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 $(B + 3 \times q)$-bits that satisfy the constraints are found then the program has succeeded and it outputs these sequences.

Consider those sets *W*, *X*, *Y* and *C* in Figure 1. The set, *C*, includes three subsets: C_1, C_2, and C_3. The sets *W*, *X*, and *Y* are {1, 2}, {3, 4}, and {5, 6}, respectively. DNA sequences generated by the Adleman program modified were shown in Table 2. The program took one mutation and two mutations to make new DNA sequences for the front six elements and the last three elements. With the nearest neighbour parameters, the 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 delta G for the probe binding a ‘1’ in the first bit is thus estimated to be 25 kcal/mol and the delta G for the probe binding a '0' is estimated to be 24.3 kcal/mol.

Table 2 *Sequences chosen to represent the three subsets in C and every element in W, X and Y in Figure 1*

Subset	*5’→3’ DNA Sequence*
C_1^0	AATTCACAAACAATT
C_2^0	ACTCCTTCCCTACTC
C_3^0	TCTCTCTCTAATCAT
C_1^1	TCTCCCTATTTATTT
C_2^1	TCACCAAACCTAAAA
C_3^1	CCATCATCTACCTTA
w_1^0	ACTCACATACACCAC
w_2^0	CTTCTCCACTATACT
x_1^0	AAACTATCATACTTC
x_2^0	TTCAATAAACATTTT
y_1^0	TTTTTCTCTCCCAAA
y_2^0	TTTACCCTACTATCA
w_1^1	CAACCTATTATCTTA
w_2^1	CCTAAATCTCCAATA
x_1^1	CTCTCAACAATCAAA
x_2^1	ACCTCCTTAACACTT
y_1^1	CCCTATCACTAATAC
y_2^1	TATAACCCATCCATA

The program simulated a mix-and-split combinatorial synthesis technique (Cukras et al., 1998) to synthesise the library strand to every possible 3-dimensional matching. Those library strands were shown in Table 4 and, respectively, represent every possible 3-dimensional matching: ∅, $\{C_3\}$, $\{C_2\}$, $\{C_2, C_3\}$, $\{C_1\}$, $\{C_1, C_3\}$, $\{C_1, C_2\}$, and $\{C_1, C_2, C_3\}$. The program was 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 subsection 3.2) in the probe sequences.

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)
C_1^1	114.4	299.4	25
C_1^0	107.8	278.6	24.3
C_2^1	111.5	284.8	26.2
C_2^0	109.1	279	25.9
C_3^1	105.2	270.5	24.4
C_3^0	97.3	252.3	22.1
w_1^1	107	282.4	22.5
w_1^0	94.7	239.8	22.8
w_2^1	111.1	288.3	25
w_2^0	101.9	266	22.4
x_1^1	101.3	258	24.1
x_1^0	102.1	269.7	21.4
x_2^1	109.6	282.8	25
x_2^0	110.6	289.3	23.9
y_1^1	106.3	278.3	23.1
y_1^0	114.8	292.2	27.5
y_2^1	109.6	282.8	25.1
y_2^0	106.8	278.4	23.5

Table 4 *DNA sequences chosen represent every possible 3-dimensional matching in the tube T_0*

∅	5'-AATTCACAAACAATTACTCCTTCCCTACTCT CTCTCTCTAATCAT-3'
	3'-TTAAGTGTTTGTTAATGAGGAAGGGATGAG GAGAGAGATTAGTA-5'
$\{C_3\}$	5'-AATTCACAAACAATTACTCCTTCCCTACTCC CATCATCTACCTTA-3'
	3'-TTAAGTGTTTGTTAATGAGGAAGGGATGAG GGTAGTAGATGGAAT-5'
$\{C_2\}$	5'-AATTCACAAACAATTTCACCAAACCTAAAAT CTCTCTCTAATCAT-3'
	3'-TTAAGTGTTTGTTAAAGTGGTTTGGATTTTA GAGAGAGATTAGTA-5'
$\{C_2, C_3\}$	5'-AATTCACAAACAATTTCACCAAACCTAAAAC CATCATCTACCTTA-3'
	3'-TTAAGTGTTTGTTAAAGTGGTTTGGATTTTG GTAGTAGATGGAAT-5'
$\{C_1\}$	5'-TCTCCCTATTTATTTACTCCTTCCCTACTCT CTCTCTCTAATCAT-3'
	3'-AGAGGGATAAATAAATGAGGAAGGGATGA GAGAGAGAGATTAGTA-5'
$\{C_1, C_3\}$	5'-TCTCCCTATTTATTTACTCCTTCCCTACTCC CATCATCTACCTTA-3'
	3'-AGAGGGATAAATAAATGAGGAAGGGATGA GGGTAGTAGATGGAAT-5'
$\{C_1, C_2\}$	5'-TCTCCCTATTTATTTTCACCAAACCTAAAAT CTCTCTCTAATCAT-3'
	3'-AGAGGGATAAATAAAAGTGGTTTGGATTTT AGAGAGAGATTAGTA-5'
$\{C_1, C_2, C_3\}$	5'-TCTCCCTATTTATTTTCACCAAACCTAAAAC CATCATCTACCTTA-3'
	3'-AGAGGGATAAATAAAAGTGGTTTGGATTTT GGTAGTAGATGGAAT-5'

Table 5 *The energy over all probe/library strand interactions*

	Enthalpy energy (H)	*Entropy energy* (S)	*Free energy* (G)
Average	106.728	276.255	24.1222
Standard deviation	5.34658	14.4172	1.53821

The Adleman program was employed for computing the distribution of the types of potential mishybridisations. The distribution of the types of potential mishybridisations 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, 280, 574, 933, 1520, 2008, 2074, 1552, 920, 507, 207, 70, 13, 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 subsection 3.2 has been satisfied. It is very clear that the number of matches peaks at 5(2074). That is to say that there are 2074 occurrences where a probe matches a strand at 5 places.

The results for simulation of Step 2 to Step 5 are shown in Tables 6–9, respectively. From the tube T_2, the answer was found to be $\{C_1, C_2\}$.

Table 6 *DNA sequences generated by Step 2 represent possible 3-dimensional matching in the tube T_0*

∅	5'-AATTCACAAACAATTACTCCTTCCCTACT CTCTCTCTCTAATCAT-3'
$\{C_3\}$	5'-AATTCACAAACAATTACTCCTTCCCTACT CCCATCATCTACCTTA
	CAACCTATTATCTTAACCTCCTTAACACTTT ATAACCCATCCATA-3'
$\{C_2\}$	5'-AATTCACAAACAATTTCACCAAACCTAAA ATCTCTCTCTAATCAT
	CCTAAATCTCCAATAACCTCCTTAACACTTT ATAACCCATCCATA-3'
$\{C_1\}$	5'-TCTCCCTATTTATTTACTCCTTCCCTACT CTCTCTCTCTAATCAT
	CAACCTATTATCTTACTCTCAACAATCAAAC CCTATCACTAATAC-3'
$\{C_1, C_2\}$	5'-TCTCCCTATTTATTTTCACCAAACCTAAA ATCTCTCTCTAATCAT
	CAACCTATTATCTTACCTAAATCTCCAATAC TCTCAACAATCAAAA
	CCTCCTTAACACTTCCCTATCACTAATACTA TAACCCATCCATA-3'

Table 7 *DNA sequences generated by Step 3 represent legal 3-dimensional matching in the tube T_0*

$\{C_1, C_2\}$	5'-TCTCCCTATTTATTTTCACCAAACCTAAAA TCTCTCTCTAATCAT
	CAACCTATTATCTTACCTAAATCTCCAATAC TCTCAACAATCAAAA
	CCTCCTTAACACTTCCCTATCACTAATACTA TAACCCATCCATA-3'

Table 8 *DNA sequences generated by Step 4 represent legal 3-dimensional matching in the tube T_2*

$\{C_1, C_2\}$	5'-TCTCCCTATTTATTTTCACCAAACCTAAAATCTCTCTCTAATCAT CAACCTATTATCTTACCTAAATCTCCAATACTCTCAACAATCAAAA CCTCCTTAACACTTCCCTATCACTAATACTATAACCCATCCATA-3'

Table 9 *The answer was found from Step 5 in the tube T_2*

The answer	*5'→3' DNA sequence*
$\{C_1, C_2\}$	5'-TCTCCCTATTTATTTTCACCAAACCTAAAATCTCTCTCTAATCAT CAACCTATTATCTTACCTAAATCTCCAATACTCTCAACAATCAAAA CCTCCTTAACACTTCCCTATCACTAATACTATAACCCATCCATA-3'

5 CONCLUSIONS

The famous Cook's theorem (Cormen et al., 2003; 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 is correct in a general *electronic* computer. In this paper, we proposed the algorithm for dealing with the 3-dimensional matching problem. Another famous NP-complete problem is the set-partition problem. The set-partition problem can be reduced to the 3-dimensional matching problem. But our algorithm cannot be applied to solving the set-partition problem. Therefore, we are not sure whether Cook's theorem is correct in a *molecular* computer.

Currently, there still are lots of NP-complete problems unsolved because it is very difficult to make basic biological operations for supporting 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 problem unresolved with the Adleman-Lipton model and the sticker model.

REFERENCES

Adleman, L. (1994) 'Molecular computation of solutions to combinatorial problems', *Science*, November 11, Vol. 266, pp.1021–1024.

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

Amos, M. (1997) *DNA Computation*, PhD Thesis, Department of Computer Science, the University of Warwick, pp.29–38.

Arita, M., Suyama, A. and Hagiya, M. (1997) 'A heuristic approach for Hamiltonian path problem with molecules', *Proceedings of 2nd Genetic Programming (GP-97)*, pp.457–462.

Boneh, D., Dunworth, C., Lipton, R.J. and Sgall, J. (1996) 'On the computational power of DNA', *Discrete Applied Mathematics, Special Issue on Computational Molecular Biology*, Vol. 71, pp.79–94.

Braich, R.S., Johnson, C., Rothemund, P.W.K., Hwang, D., Chelyapov, N. and Adleman, L.M. (2000) 'Solution of a satisfiability problem on a gel-based DNA computer', *Proceedings of the 6th International Conference on DNA Computation in the Springer-Verlag Lecture Notes in Computer Science series*, pp.27–42.

Chang, W-L. and Guo, M. (2002a) 'Solving the dominating-set problem in Adleman-Lipton's model', *The Third International Conference on Parallel and Distributed Computing, Applications and Technologies*, Japan, pp.167–172.

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

Chang, W-L., Ho, M. and Guo, M. (2004) 'Fast parallel molecular solution to the dominating-set problem on massively parallel bio-computing', *Parallel Computing*, Vol. 30, Nos. 9–10, pp.1109–1125.

Cormen, T.H., Leiserson, C.E. and Rivest, R.L. (2003) *Introduction to Algorithms*, ISBN 0-07-013151-1, the MIT Press, pp.966–1021.

Cukras, A.R., Faulhammer, D., Lipton, R.J. and Landweber, L.F. (1998) 'Chess games: a model for RNA-based computation', *Proceedings of the 4th DIMACS Meeting on DNA Based Computers*, held at the University of Pennsylvania, June 16–19, pp.27–37.

Fu, B. (1997) *Volume Bounded Molecular Computation*, PhD Thesis, Department of Computer Science, Yale University, pp.45–63.

Garey, M.R. and Johnson, D.S. (1979) *Computer and Intractability*, Freeman, San Francisco, CA, pp.50–53.

Garzon, M.H. and Deaton, R.J. (1999) 'Biomolecular computing and programming', *IEEE Transactions on Evolutionary Computation*, Vol. 3, pp.236–250.

Lipton, R.J. (1995) 'DNA solution of hard computational problems', *Science*, Vol. 268, pp.542–545.

Mir, K. (1998) 'A restricted genetic alphabet for DNA computing', in Baum, E.B. and Landweber, L.F. (Eds.): *DNA Based Computers II: DIMACS Workshop, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, Providence*, June 10–12, RI, Vol. 44, pp.243–246.

Morimoto, N., Arita, M. and Suyama, A. (1999) 'Solid phase DNA solution to the Hamiltonian path problem', *DIMACS (Series in Discrete Mathematics and Theoretical Computer Science)*, Vol. 48, pp.93–206.

Narayanan, A. and 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.

Paun, G., Rozenberg, G. and Salomaa, A. (1998) *DNA Computing: New Computing Paradigms*, ISBN: 3-540-64196-3, Springer-Verlag, New York, pp.43–74.

Perez-Jimenez, M.J. and Sancho-Caparrini, F. (2001) 'Solving Knapsack problems in a sticker based model', *2nd Annual Workshop on DNA Computing, DIMACS: series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, pp.161–171.

Quyang, Q., Kaplan, P.D., Liu, S. and Libchaber, A. (1997) 'DNA solution of the maximal clique problem', *Science*, Vol. 278, pp.446–449.

Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N.V., Goodman, M.F., Rothemund, P.W.K. and Adleman, L.M. (1999) 'A sticker based model for DNA computation', in Landweber, L. and Baum, E. (Eds.): *2nd Annual Workshop on DNA Computing*, *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, Princeton University, pp.1–29.

Shin, S-Y., Zhang, B-T. and Jun, S-S. (1999) 'Solving traveling salesman problems using molecular programming', *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99)*, Vol. 2, pp.994–1000.

Sinden, R.R. (1994) *DNA Structure and Function*, Academic Press, New York, pp.73–97.

Winfree, E., Liu, F., Wenzler, L.A. and Seeman, N.C. (1998) 'Design and self-assembly of two-dimensional DNA crystals', *Nature*, Vol. 394, pp.539–544.