Fast parallel bio-molecular solutions: the set-basis problem

Weng-Long Chang*

Department of Computer Science and Information Engineering,
National Kaohsiung University of Applied Sciences,
415 Chien Kung Road, 807 Kaohsiung, Taiwan, ROC
E-mail: changwl@cc.kuas.edu.tw
*Corresponding author

Michael Ho

Department of Information Management,
School of Information Technology,
Ming Chuan University,
5, Teh-Ming Rd., Gwei-Shan, 333 Taoyuan, Taiwan, ROC
E-mail: mhoincerritos@yahoo.com

Minyi Guo

School of Computer Science and Engineering,
University of Aizu, Aizu Wakamatsu City, 965 8580 Fukushima, Japan
E-mail: minyi@u-aizu.ac.jp

Chengfei Liu

Faculty of Information and Communication Technologies,
Swinburne University of Technology,
Melbourne, 3122 VIC, Australia
E-mail: cliu@swin.edu.au

Abstract: In the paper, it is demonstrated how to apply sticker in the sticker-based model for constructing solution space of DNA for the setbasis problem and how to apply DNA operations in the Adleman-Lipton model to solve that problem from solution space of sticker. Furthermore, this work shows the ability of DNA-based computing for resolving the NP-complete problems.

Keywords: biological parallel computing; DNA-based supercomputer; NP-complete problem; set-basis problem.

Reference to this paper should be made as follows: Chang, W-L., Ho, M., Guo, M. and Liu, C. (2006) 'Fast parallel bio-molecular solutions: the set-basis problem', *Int. J. Computational Science and Engineering*, Vol. 2, Nos. 1/2, pp.72–80.

Biographical notes: Weng-Long Chang received his PhD Degree in Computer Science and Information Engineering from the National Cheng Kung University, Taiwan in 1999. His research interests include molecular computing, and languages and compilers for parallel computing.

Michael Ho is an Associate Professor of Ming Chuan University with 25 years industrial/academic experiences in the computing field. He had worked as a Senior Software Engineer and Project Leader developing B2B/B2C/C2C web/multimedia applications and a Senior Database Administrator for SQL/Oracle/DB2 with many major US corporation LAN/WAN systems. He had ten years of college teaching/research experiences as an Associate/Assistant Professor at Southern Taiwan University of Technology/Central Missouri State University/the University of Texas. He earned a PhD in IS/CS with management /accounting minors from UT Austin. His research interests include computation theories, software engineering, multimedia, data mining, SOC, parallel/quantum/DNA computing.

Minyi Guo received his PhD Degree in Information Science from the University of Tsukuba, Japan in 1998. From 1998 to 2000, He had been a research scientist of NEC Soft, Ltd. Japan. He is currently an Associate Professor at the Department of Computer Software, The University of Aizu, Japan. He has served as General Chair, Program Committee and Organising Committee Chair for many international conferences. He is an Editor in Chief of the *Journal of Embedded Systems*. He is also on the editorial board of the *International Journal of High Performance Computing and Networking*, *Journal of Embedded Computing*, *Journal of Parallel and Distributed Scientific and Engineering Computing*, and *International Journal of Computer and Applications*. His research interests include parallel and distributed processing, parallelising compilers, data parallel languages, data mining, molecular computing and software engineering.

Chengfei Liu received his PhD Degree in Computer Science from the Nanjing University in 1988. He is currently an Associate Professor at the Faculty of Information and Communication Technologies, Swinburne University of Technology, Australia. He was a Senior Lecturer at the University of South Australia, a Lecturer at the University of Technology Sydney, and a Senior Research Scientist at DSTC in University of Queensland. He also held visiting positions at the IBM Silicon Valley Laboratory and the University of Aizu. His current research interests include advanced databases, XML data management and integration, advanced transaction models, workflows, and Web services.

1 Introduction

Nowadays, producing roughly 10^{18} DNA strands, that too in a test tube, through advances in molecular biology is possible (Sinden, 1994). Basic biological operations can be applied to simultaneously operate 10^{18} bits of information. This is to say that there are 10^{18} data processors to be parallelly executed. Hence, it is very clear that biological computing can provide a very similar parallelism for dealing with the problem in the real world.

Adleman (1994) wrote the first paper that DNA strands could be used to deal with solutions for an instance of the NP-complete Hamiltonian path problem (HPP). Lipton (1995) wrote the second paper that demonstrated that the Adleman techniques could be used to solve the NP-complete satisfiability (SAT) problem (the first NP-complete problem). Adleman and his coauthors (Roweis et al., 1999) proposed sticker for enhancing the Adleman-Lipton model.

In this paper, we use a sticker in the sticker based model for constructing a solution space of DNA for the setbasis problem. Simultaneously, we also apply DNA operations in the Adleman-Lipton model to develop a DNA algorithm. It is shown from the main result of the proposed DNA algorithm that the setbasis problem is resolved with biological operations in the Adleman-Lipton model from the solution space of the sticker. Furthermore, this work shows the ability of DNAbased computing for resolving the NP-complete problems.

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 given. Section 3 introduces a DNA algorithm for solving the setbasis problem from the solution space of the sticker in the Adleman-Lipton model. In Section 4, the experimental result of simulated DNA computing is discussed. Conclusions are drawn in Section 5.

2 DNA model of computation

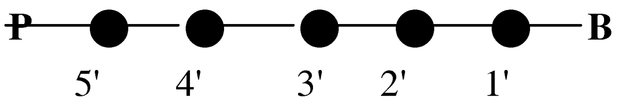
In Subsection 2.1, a summary of DNA structure is given and the Adleman-Lipton model is described in detail. In Subsection 2.2, a comparison of the Adleman-Lipton model with other models is given.

2.1 The Adleman-Lipton model

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

Distinct nucleotides are detected only with their bases, which come in two sorts: purines and pyrimidines (Sinden, 1994; Paun et al., 1998). Purines include adenine and guanine, abbreviated *A* and *G*. Pyrimidines contain cytosine and thymine, abbreviated *C* and *T*. Because nucleotides are only distinguished from their bases, they are simply represented as *A*, *G*, *C*, or *T* nucleotides, depending upon the sort of base that they have. The structure of a nucleotide, cited from (Paun et al., 1998), is illustrated (in a very simplified way) in Figure 1. In Figure 1, B is one of the four possible bases (*A*, *G*, *C*, or *T*), *P* is the phosphate group, and the rest (the ‘stick’) is the sugar base (with its carbons enumerated 1' through 5').

Figure 1 A schematic representation of a nucleotide



Nucleotides can link together in two different ways (Sinden, 1994; Boneh et al., 1996; Paun et al., 1998). In the first method, the 5'-phosphate group of one nucleotide is joined with a 3'-hydroxyl group of the other, forming a phosphodiester bond. The resulting molecule has the 5'-phosphate group of one nucleotide, denoted as the 5' end, and the 3'-OH group of the other nucleotide available, denoted as the 3' end, for bonding. This gives the molecule directionality, and we can talk about the direction of 5' end to 3' end or 3' end to 5' end. The second method is that the base of one nucleotide interacts with the base of the other to form a hydrogen bond. This bonding is the subject of the following restriction on the base pairing: *A* and *T* can pair together, and *C* and *G* can pair together – no other pairings are possible. This pairing principle is called the Watson-Crick complementarity (named after James D. Watson and Francis H.C. Crick who deduced the famous double helix structure of DNA in 1953, and won the Nobel Prize for the discovery).

A DNA strand is essentially a sequence (polymer) of four types of nucleotides detected by one of the four bases that 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 monomer (i.e., 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 monomer, then the length of the double stranded DNA is 20 base pairs, also written 20 bp. (For more discussion of the relevant biological background refer to Sinden (1994), Boneh et al. (1996) and Paun et al. (1998)).

In the Adleman-Lipton model (Adleman, 1994; Lipton, 1995), splints are used to construct and correspond to the edges of a particular graph, the paths of which represent 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 coauthors (Roweis et al., 1999) proposed the sticker based model, which is an abstract model of molecular computing based on DNA with a random access memory and a new form of encoding the information, to enhance the Adleman-Lipton model.

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

The Adleman-Lipton model:

A (test) tube is a set of molecules of DNA (i.e., a multiset 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 strand *S* as a substrand 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.
- *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 *The comparison of the Adleman-Lipton model with other models*

Techniques in the Adleman-Lipton model could be applied for solving the NPcomplete 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) proved that restriction enzymes could be used to solve the NPcomplete 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 increases exponentially. Arito et al. (1997) described new molecular experimental techniques for searching a Hamiltonian path. Morimoto et al. (1999) offered a solid phase method for finding a Hamiltonian path. Narayanan et al. (1998) demonstrated 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 integers and real values. Their method could also be employed towards solving the travelling salesman problem. Amos (1997) proposed the parallel filtering model for resolving the Hamiltonian path problem, the subgraph isomorphism problem, the 3-vertex-colourability problem, the clique problem and the independent set problem. Roweis et al. (1999) proposed the stickerbased 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 the stickerbased model (Roweis et al., 1999) to resolve knapsack problems. Fu (1997) proposed new algorithms to resolve 3-SAT, 3-Coloring and the independent set. In our previous work, Chang et al. (2002a, 2002b, 2002c, 2002d) proved how the DNA operations from the 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 setpacking problem. In our previous work, Guo et al. (2004) also employed the stickerbased model and the Adleman-Lipton model to deal with the dominating set problem for decreasing the error rate of hybridisation.

3 Using sticker for solving the set-basis problem in the Adleman-Lipton model

In Subsection 3.1, the set-basis problem is defined. Applying sticker for constructing solution space of DNA sequences for the set-basis problem is introduced in Subsection 3.2. In subsection 3.3, a DNA algorithm is proposed for resolving the set-basis problem. In subsection 3.4, the complexity of the proposed algorithm is discussed.

3.1 Definition of the set-basis problem

Assume that a finite set S is $\{s_1, \ldots, s_d\}$, where s_e is the eth element for $1 \le e \le d$ in S. $|S|$ is denoted as the number of elements in S and $|S|$ is equal to d. Suppose that a collection C is a set of subsets of the finite set S and is $\{C_1, \ldots, C_f\}$, where C_g is the gth element for $1 \le g \le f$ in C. $|C|$ is denoted as the number of subsets in C and $|C|$ is equal to f. Assume that a positive integer $k \le |C|$. Mathematically, the set-basis problem is to find a collection B of subsets of S with $|B| = k$ such that for each $C_g \in C$, there is a subcollection of B whose union is exactly C_g. That problem has been proved to be an NPcomplete problem (Garey and Johnson, 1979).

A finite set S and a collection C of subsets for S are shown in Figure 2. The finite set S is {1,2} and the collection C is {{1}, {2}}. The two sets define a set-basis problem. The set-basis for S and C in Figure 2 is {{1}, {2}}. Hence, the size of the set-basis problem for S and C in Figure 2 is two. It is indicated from (Garey and Johnson, 1979) that finding a set-basis is a NPcomplete problem, so it can be formulated as a 'search' problem.

Figure 2 A finite set S and a collection C of subsets for S

$$S = \{1,2\} \text{ and } C = \{\{1\},\{2\}\}$$

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

In the Adleman-Lipton model, the main idea is to first generate solution space of DNA sequences for those problems. Then, basic biological operations are used to select a legal solution and to remove the illegal solution from the solution space. Therefore, for a finite set S, with d elements and a collection C with f elements for subsets of the finite set, the first step for resolving the set-basis problem is to produce a test tube, which includes all of the possible subsets for the finite set. A d-digit binary number can be used to represent each possible subset for S. Suppose that S^1 is a subset of S and a d-digit binary number is used to represent S^1. If the ith bit in the d-digit binary number is set to 1, then it represents that the ith element in S is in S^1. If the ith bit in a d-digit binary number is set to 0, then it represents that the corresponding element is not in S^1. In this way, all of the possible subsets in S are transformed into an ensemble of all d-digit binary numbers.

Hence, with the above method, Table 1 denotes the solution space of the subsets for the finite S in Figure 2. The binary number 00 in Table 1 represents the empty subset. The binary numbers 01 and 10 in Table 1 represent the subsets {1} and {2}, respectively. The binary number 11 in Table 1 represents the subset {2,1}. Though there are four 2-digit binary numbers for representing four possible subsets in Table 1, not every 2-digit binary number corresponds to a legal solution. Hence, in next subsection, basic biological operations are used to develop an algorithm for removing illegal subsets and determining legal solutions.

Since a collection C with f elements is a set of subsets of S, every element in each subset in C comes from S. Therefore, every subset in C is represented by the same method above. Table 2 denotes representation of each subset in the collection C in Figure 2. The first subset {1} in Table 2 is represented as the 2 digit binary number 01. The second subset {2} in Table 2 is represented as the 2 digit binary number 10.

Table 1 The solution space of the subsets for the finite S in Figure 2

2-digit binary number	*The corresponding subset*
00	<*t*>
01	{1}
10	{2}
11	{2,1}

Table 2 Denotation of each subset in the collection C in Figure 2

The subset	*The corresponding 2-digit binary representation*
{1}	01
{2}	10

To implement this method, assume that an unsigned integer X is represented by a binary number x_d, x_{d-i}, …, x_1, where the value of x_i is 1 or 0 for $1 \le i \le d$. The integer X can take 2^d possible values. Each such value represents a subset for a finite set S of size d. Therefore, it is very clear that an unsigned integer X forms 2^d possible subsets. A bit x_i in X represents the ith element in S. Given a subset of S, if the ith element is in the subset, then the value of x_i is set to 1; if the ith element is not in the subset, then the value of x_i is set to 0.

To represent all possible subsets for a finite set S with d elements for the set-basis problem, a sticker (Roweis et al., 1999; Braich et al., 2000) is used to construct the solution space for that problem. For every bit x_i where $1 \le i \le d$, two distinct 15 base value sequences are designed. One represents the value 1 for x_i and another represents the value 0 for x_i. For the convenience of presentation, assume that x_i^1 denotes the value of x_i to be 1 and x_i^0 defines the value of x_i to be zero. Each of the 2^d possible subsets is represented by a library sequence of $15 \times d$ 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. Similarly, d 15 base value sequence is also applied to represent every element in a subset of a collection C, where C is a set of subsets for a finite set S.

It is pointed out from (Roweis et al., 1999; Braich et al., 2000) 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 hybridisation. The design must also exclude sequences that might encourage unintended probe-library hybridisation. To help achieve these goals, sequences are computer generated to satisfy the following constraint (Braich et al., 2000).

- library sequences contain only A's, T's, and C's
- all library and probe sequences have no occurrence of 5 or more consecutive identical nucleotides; i.e., no runs of more than 4 A's, 4 T's, 4 C's or 4 G's occur in any library or probe sequences
- every probe sequence has at least four mismatches with all 15 base alignment of any library sequence (except for with its matching value sequence)
- every 15 base subsequence of a library sequence has at least four mismatches with all 15 base alignment of itself or any other library sequence
- no probe sequence has a run of more than seven matches with any eight base alignment of any library sequence (except for with its matching value sequence)
- no library sequence has a run of more than seven matches with any eight base alignment of itself or any other library sequence
- 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 (KalimMir, 1996). 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 two elements in the finite set S in Figure 2, the DNA sequences generated are:

$$x_1^0 = CCACATATCCATCCC, \quad x_2^0 = CCTACCTCTCACCTT,$$
$$x_1^1 = CCCATCTTTCTTAAC \text{ and } x_2^1 = CATTACCTCTTTACT.$$

Because the first subset in the collection C in Figure 2 only includes the first element in S, two 15 base DNA sequences, $CCTACCTCTCACCTT(x_2^0)$ and $CCCATCTTTCTTAAC(x_1^1)$ are used for representing it. Similarly, the second subset in the collection C in Figure 2 only contains the second element in S, so two 15 base DNA sequences, $CATTACCTCTTTACT(x_2^1)$ and $CCACATATCCATCCC(x_1^0)$ are used for representing it. For every possible subset of the finite set S in Figure 2, the corresponding library strand is synthesised by employing a mix and split combinatorial synthesis technique (Cukras et al., 1998). Similarly, for any d-element set, all of the library strands for representing every possible subset can be also synthesised with the same technique.

3.3 *The DNA algorithm for solving the set-basis problem*

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

Algorithm 1: Solving the set-basis problem.

(1) Input (T_0), where the tube T_0 includes solution space of DNA sequences to encode all of the possible subsets for any d-element set S, with those techniques mentioned in Subsection 3.2.

```
(2) For = 1 to |C|, where |C| is the number of subsets in a collection C.
    (a) For $k = 1$ to $|C_j|$, where $|C_j|$ is the number of elements in $C_j$ in C. Assume that the kth element in $C_j$ is the ith element in S and XI is used to represent it.
    (b) $T_0 = +(T_0, x_i^{\prime})$ and $T_{OFF} = –(T_0, x_i^{\prime})$.
    (c) $T_{ON} = U(T_{OFF}, T_{ON})$
EndFor
    (d) For $k = 1$ to $|S| – |C_j|$, where $|S|$ is the number of elements in S. Assume that the mth element in S for $1 < m < d$ is out of $C_j$ and $x_m$ is used to represent it.
    (e) $T_0 = +(T_0, x^0{}_m)$ and $T_{OFF} = –(T_0, x^0{}_m)$.
    (f) $T_{ON} = U(T_{OFF}, T_{ON})$
EndFor
    (g) $T_j = U(T_j, T_0)$
    (h) $T_0 = U(T_0, T_{ON})$
EndFor
(3) For $j = 1$ to |C|, where |C| is the number of subsets ina collection C.
    (a) Read ($T_j$).
EndFor
```

Theorem 3.1: *From those steps in Algorithm 1, the set-basis problem can be solved.*

Proof: In Step 1, a test tube of DNA strands, that encode all 2^d possible input bit sequences $x_d, …, x_1$, is generated. It is very clear that the test tube includes all 2^d possible subsets for any d-element set, S.

In Step 2, it contains one outer loop and two inner loops. The outer loop will execute $|C|$ times, where $|C|$ is the number of subsets in a collection C. The first inner loop will execute $(|C_j| \times |C|)$ times, where $|C_j|$ is the number of elements in C_j in C. The second inner loop will execute $((|S| – C_j|) \times |C|)$ times, where $|S|$ is the number of elements in S. According to definition of set-basis (Cormen et al., 1990; Garey and Johnson, 1979), the first execution of Step 2b applies 'extraction' to form two test tubes: T_0 and T_{OFF}. The first tube T_0 contains all of the strands that have $x_i = 1$. The second tube T_{OFF} consists of all of the strands that have $x_i = 0$. The tube T_{ON} represents those subsets, which contains the element s_i. The tube T_{OFF} represents those subsets, which do not include the element s_i. Then, the first execution of Step 2(c) uses the 'merge' operation to pour two tubes, T_{OFF} and T_{ON} into the tube, T_{ON}. That is to say that the tube T_{ON} obtains the strands in the tube T_{OFF}. After Steps 2(b)–2(c) are repeated to execute $(|C_j|)$ times, the tube T_0 includes the strands, which currently satisfy the subset C_j in C to definition of set-basis.

Next, the first execution of Step 2(e) employs the 'extraction' operation to generate two test tubes: T_0 and T_{OFF}. The first tube T_0 contains all of the strands that have $x_m = 0$. The second tube T_{OFF} consists of all the strands that have $x_m = 1$. Because there is no element, s_m, in C_j, the tube T_0 represents those strands which do not contain the element s_m and the tube T_{OFF} represents those strands which include the element s_m. The first execution of Step 2(f) uses the 'merge' operation to pour two tubes, T_{OFF} and T_{ON} into the tube, T_{ON}. This is to say that the tube T_{ON} obtains the strands in the tube T_{OFF}. After Steps 2(e)–2(f) are repeated to execute $(|S| – |Cj|)$ times, the tube T_0 includes the strands, which satisfy the subset C_j in C for definition of set-basis.

Then, the first execution of Step 2(g) uses the 'merge' operation to pour two tubes, T_j and T_0 into the tube, T_j. That is to say that the tube T_j consists of the strands satisfying the subset C_j in C for definition of set-basis. The first execution of Step 2(h) also applies the 'merge' operation to pour two tubes, T_{ON} and T_0 into the tube, T_0. This is to say that the tube T_0 includes the strands, which do not satisfy the subset C_j in C for definition of set-basis. For other subsets in C, the similar processing is also finished. Therefore, after all of the second steps are processed, d new tubes are generated. The new tube T_j for $d \geq j \geq 1$ contains those strands that satisfy the subset C_j in C for definition of set-basis.

Since the set-basis problem is to find a collection B of subsets of S such that for each $C_j \in C$, there is a subcollection of B whose union is exactly C_j, the tube T_j for $d \geq j \geq 1$ contains those DNA sequences that satisfy the subset C_j in C for definition of set-basis. Therefore, the first execution of Step 3(a) employs 'Read' operation to describe 'sequence' of a molecule in the tube T_1. After Step 3(a) is repeated to execute $|C|$ times, the answer for the set-basis problem is found and described. □

The finite set S and the collection C in Figure 2 are used to show the power of Algorithm 1. It is pointed out from Step 1 in Algorithm 1 that the tube T_0 is filled with four library stands with those techniques mentioned in subsection 3.2, representing four possible subsets for the set S in Figure 2. The number of the elements in S in Figure 2 is two, so the number of executions to the outer loop in Step 2 of Algorithm 1 is two times. The number of the element in the two subsets in the collection C is one. Therefore, the number of executions for the first inner loop in Step 2 of Algorithm 1 is two times and the number of execution for the second inner loop in Step 2 of Algorithm 1 is also two times.

According to the first execution of Step 2(b) of Algorithm 1, two tubes are generated. The first tube, T_0, includes those subsets: {1} and {1, 2} and the second tube, T_{OFF}, also contains those subsets: ϕ and {2}. Next, the first execution to Step 2(c) in Algorithm 1 pours the two tubes T_{OFF} and T_{ON} into the tube T_{ON}. Therefore, the tube T_{ON} now includes those subsets: ϕ and {2}. Two tubes are produced from the first execution to Step 2(e) in Algorithm 1. The first tube, T_0, includes the subset: {1} and the second tube, T_{OFF}, also contains the subset: {1, 2}. Then, the first execution to Step 2(f) in Algorithm 1 pours the two tubes T_{OFF} and T_{ON} into the tube T_{ON}. Hence, the tube T_{ON} now includes those subsets: ϕ, {2} and {1, 2}. The first execution to Step 2(g) in Algorithm 1 pours the two tubes T_j and T_0 into the tube T_j. So, the tube T_j now includes the subset: {1}. Next, the first execution to Step 2(h) in Algorithm 1 pours the two tubes T_0 and T_{ON} into the tube T_0. Hence, the tube T_0 now includes those subsets: ϕ, {2} and {1, 2}. The same processing can be applied to deal with the second subset {2} in the collection *C*. After all of the second steps are processed, it finally produces two new tubes. The two tubes T_1 and T_2, respectively, include {1} and {2}.

Because the number of the subsets in the collection *C* is two, the number of execution to Step 3 is two times. The first execution of Step 3(a) employs the 'Read' operation to describe the 'sequence' of a molecule in the tube T_1. After Step 3(a) is repeated to execute two times, the answer for the set-basis problem to the finite set *S* and the collection *C* in Figure 2 is found to be {{1}, {2}}.

3.4 The complexity of the proposed DNA algorithm

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

Theorem 3.2: *The set-basis problem for any d-element set S and any f-subset collection C can be solved with $O(d \times f)$ biological operations in the Adleman-Lipton model.*

Proof: Algorithm 1 can be applied for solving the set-basis problem for any *d*-element set *S* and any *f*-subset collection *C*. Algorithm 1 includes two main steps. Step 2 is mainly used to determine legal set-basis and to remove illegal set-basis from all of the 2*d* possible library strands. From Algorithm 1, it is very obvious that Step 2(b) and Step 2(e) totally take ($d \times$ f) 'extraction' operations and Step 2(c) and Step 2(f) totally take ($d \times f$) 'merge' operations. Since *C* contains *f* elements, therefore, it is very clear from Algorithm 1 that Step 2(g) and Step 2(h) totally take ($2 \times f$) 'merge' operations. Step 3 is used to find a set-basis. It is pointed out from Algorithm 1 that Step 3(a) takes *f* 'read' operations. Hence, from the statements mentioned above, it is at once inferred that the time complexity of Algorithm 1 is $O(d \times$ f) biological operations in the Adleman-Lipton model. □

Theorem 3.3: *The set-basis problem for any d-element set S and any f-subset collection C can be solved with a sticker to construct $O(2^d)$ library strands in the Adleman-Lipton model, where d is the number of elements in S.*

Proof: Refer to Theorem 3.2. □

Theorem 3.4: *The set-basis problem for any d-element set S and any f-subset collection can be solved with O(f) tubes in the Adleman-Lipton model, where f is the number of elements in C.*

Proof: Refer to Theorem 3.2. □

Theorem 3.5: *The set-basis problem for any d-element set S and any f-subset collection can be solved with the longest library strand, $O(15 \times d)$, in the Adleman-Lipton model, where d is the number of elements in S.*

Proof: Refer to Theorem 3.2. □

4 Experimental results of simulated DNA computing

We modified the Adleman program (Braich et al., 2000) and implemented in a PC with one Pentium II 200 MHz CPU and 64 MB main memory. Our operating system is Windows 98 and our compiler is C++ Builder 6.0. The modified program was executed to generate DNA sequences for solving the set-basis problem for any *d*-element set *S* and any *f*-subset collection *C*. Because the source code of the two functions drand48() and drand48() were 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 for 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. For any *d*-element set *S* and any *f*-subset collection *C*, the size of library strands is 2^d. Due to the limit of memory space and harddisk space, the value of *d* was less than or equal to 20.

The Adleman program is used for constructing each 15-base DNA sequence for each bit of the library. For each bit, the program is applied for generating two 15-base random sequences (for 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 fails and so it exits, printing the sequence found so far. If *d*-bits that satisfy the constraints are found then the program succeeds and it outputs these sequences.

Consider the finite set *S* and the collection *C* in Figure 2. The finite set includes {1, 2} and the collection *C* contains {{1}, {2}}. DNA sequences generated by the Adleman program modified were shown in Table 3. The program, correspondingly, takes one mutation to make new DNA sequences for the first element and the second element in *S*. With the nearest neighbour parameters, the program is 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 is shown in Table 4. Only *G* really matters to the energy of each bit. For example, the delta *G* for the probe binding a *T* in the first bit is thus estimated to be 25.8 kcal/mol and the delta *G* for the probe binding a '0' is estimated to be 28.0 kcal/mol.

Table 3 Sequences chosen to represent the two elements in *S* in Figure 2

Vertex	*5′ → 3′ DNA sequence*
x_2^0	CCTACCTCTCACCTT
x_1^0	*CCACATATCCATCCC*
x_2^1	*CATTACCTCTTTACT*
x_1^1	*CCCATCTTTCTTAAC*

Table 4 The energy for the binding of each probe to its corresponding region on a library strand

Vertex	*Enthalpy energy* (H)	*Entropy energy* (S)	*Free energy* (G)
x_2^0	109.3	278.4	26.2
x_1^0	110.9	278	28
x_2^1	106.7	279.7	23
x_1^1	112.1	288.8	25.8

The program simulates a mix and split combinatorial synthesis technique (Cukras et al., 1998) to synthesise the library strand to every possible subset. Those library strands are shown in Table 5 and, correspondingly, represent four possible subsets: φ, {1}, {2}, and {1, 2}. The program is also applied to figure out the average and standard deviation for the enthalpy, entropy and free energy over all probe/library strand interactions. The energy is shown in Table 6. 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 5 DNA sequences chosen represent all possible subsets

5' – CCT ACCTCTC ACCTTCC AC AT ATCC ATCCC – 3'
3' – GGATGGAGAGTGGAAGGTGTATAGGTAGGG –5'
5' – CCT ACCTCTC ACCTTCCC ATCTTTCTT AAC – 3'
3' – GGATGGAGAGTGGAAGGGTAGAAAGAATTG – 5'
5' – CATTACCTCTTTACTCC AC AT ATCC ATCCC – 3'
3' – GT AATGGAGAAATGAGGTGT AT AGGT AGGG – 5'
5' – CATTACCTCTTTACTCCC ATCTTTCTT AAC – 3'
3' – GTAATGGAGAAATGAGGGTAGAAAGAATTG – 5'

Table 6 The energy over all probe/library strand interactions

	Enthalpy energy (*H*)	*Entropy energy* (*S*)	*Free energy* (*G*)
Average	109.75	281.225	25.75
Standard deviation	2.02161	4.41807	1.79095

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 probestrand 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, 52, 102, 85, 95, 105, 109, 77, 36, 13, 12, 2, 0, 0, 0, 0 and 0. It is pointed out from the last five zeros that there are 0 occurrences where a probe matches a strand at 11, 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 five (109). That is to say that there are 109 occurrences where a probe matches a strand at five places.

It is indicated from the execution of Step 2 of simulation that the result generated by Step 2 was shown in Table 7. The goal of Step 3 is to find a set-basis from the result generated by Step 2. Hence, Step 3(a) of simulation, the set-basis was shown in Table 8. That is to say that the answer of the set-basis problem for the finite set *S* and the collection *C* in Figure 2 is {{1}, {2}}.

Table 7 DNA sequences generated by Step 2 represent legal subsets

5' – CCT ACCTCTC ACCTTCCC ATCTTTCTT AAC – 3'
5' – CATTACCTCTTTACTCC AC AT ATCC ATCCC – 3'

Table 8 DNA sequence represents the answer of the set-basis problem for the finite set *S* and the collection *C* in Figure 2

5' – CCTACCTCTCACCTTCCCATCTTTCTTAAC – 3'
5' – CATTACCTCTTTACTCCACATATCCATCCC – 3'

5 Conclusions

Applying splints constructs the solution space of the DNA sequence for solving the NPcomplete problem in the Adleman-Lipton and this is the reason that hybridisation has higher probabilities for errors. Adleman and his coauthors (Roweis et al., 1999) proposed a sticker to decrease probabilities of errors in hybridisation in the Adleman-Lipton model. In the proposed algorithm, the size of the solution space of the sticker is exponential. Hence, this is the limit to which we can resolve the size of the NPcomplete problem. The main result of the proposed algorithm shows that the set-basis problem is resolved with biological operations in the Adleman-Lipton model from solution space of sticker. Furthermore, this work demonstrates the ability of DNA based computing to solve NPcomplete problems.

References

Adleman, L.M. (1996) 'On constructing a molecular computer. DNA based computers', *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, pp.1–21, in the First Annual Meeting on DNA-based Computers.

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

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

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

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

Cormen, T.H., Leiserson, C.E. and Rivest, R.L. (1990) *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts.

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, 16–19 June, pp.27–37.

Fu, B. (1997) '*Volume Bounded Molecular Computation*, *PhD Thesis*, Department of Computer Science, Yale University, http://fano.ics.uci.edu/cites/Document/ Volume-Bounded-Molecular-Computation.html.

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

KalimMir (1996) 'A restricted genetic alphabet for DNA computing', in Baum, E.B. and Landweber, L.F. (Eds.): *DNA Based Computers II: DIMACS Workshop*, 10–12 June, *Vol.44 of DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, Providence, RI, 1998, pp.243–246.

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

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*, Springer-Verlag, New York, ISBN: 3-540-64196-3.

Perez-Jimenez, M.J. and Sancho-Caparrini, F. (2001) 'Solving Knapsack problems in a sticker based model', *7th Annual Workshop on DNA Computing, DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society.

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*, Princeton University, *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, 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, ISBN 0126457506.