Parallel Processing Letters, Vol. 15, No. 4 (2005) 469–479

World Scientific
www.worldscientific.com

Solving the Independent-set Problem in a DNA-Based Supercomputer Model

WENG-LONG CHANG
Department of Information Management Southern Taiwan University of Technology Tainan, Taiwan 701.

MINYI GUO
Department of Computer Software, The University of Aizu Aizu-Wakamatsu City, Fukushima 965-8580, Japan

JESSE WU
Department of Mathematics, National Kaohsiung Normal University Kaohsiung 802, Taiwan

Received April 15, 2003
Revised August 23, 2004
Communicated by Guest Editor

ABSTRACT

In this paper, it is demonstrated how the DNA (DeoxyriboNucleic Acid) operations presented by Adleman and Lipton can be used to develop the parallel genetic algorithm that solves the *independent-set problem*. The advantage of the genetic algorithm is the huge parallelism inherent in DNA based computing. Furthermore, this work represents obvious evidence for the ability of DNA based parallel computing to solve NP-complete problems.

Keywords: DNA based computing, parallelism, parallel genetic algorithm, NP-complete problems

1. Introduction

It is nowadays possible to yield a soup of roughly 10^{18} DNA strands that fits in a test tube through advances in molecular biology [1]. 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 [2] demonstrated that each DNA strand could be applied to figure out solutions for an instance of the NP-complete Hamiltonian path problem (HPP). Lipton [3] proposed to apply DNA computing to resolve the NP-complete satisfiability (SAT) problem. Mathematically, an *independent set* of a graph $G = (V, E)$ is a subset $V^1 \in V$ of vertices such that each edge in E is incident on at most one

vertex in V^1 [9]. The independent-set problem is to find a maximum-size independent set in G. This problem has been shown to be an NP-complete problem [10]. In this paper, we demonstrate how the DNA (DeoxyriboNucleic Acid) operations presented by Adleman and Lipton can be employed to generate a parallel genetic algorithm to resolving the independent-set problem.

The paper is organized as follows. In Section 2, the DNA operations proposed by Adleman and Lipton are in detail introduced. Section 3 describes the genetic method for finding a maximum-size independent set to any undirected graph. In Section 4, the experimental result of simulated DNA computing is also given. Conclusions are drawn in Section 5.

2. DNA Model of Computations

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

2.1. The Adleman-Lipton Model

A DNA (DeoxyriboNucleic Acid) is the molecule that plays the main role in DNA based computing [16]. In the biochemical world of large and small *molecules, polymers*, and *monomers*, DNA is a polymer, which is strung together from monomers called *deoxyriboNucleotides*. The monomers used for the construction of DNA are deoxyribonucleotides, which each deoxyribonucleotide contains three components: a *sugar*, a *phosphate group*, and a *nitrogenous base*. This sugar has five carbon atoms - for the sake of reference there is a fixed numbering of them. Because the base also has carbons, to avoid confusion the carbons of the sugar are numbered from 1' to 5' (rather than from 1 to 5). The phosphate group is attached to the 5' carbon, and the base is attached to the 1' carbon. Within the sugar structure there is a hydroxyl group attached to the 3' carbon.

Distinct nucleotides are detected only with their bases, which come in two sorts: *purines* and *pyrimidines*. Purines include *adenine* and *guanine*, abbreviated A and G [1,16]. Pyrimidines contain *cytosine* and *thymine*, abbreviated C and T. Because nucleotides are only distinguished from their bases, they are simply represented as A, G, C, or T nucleotides, depending upon the sort of base that they have. The structure of a nucleotide is illustrated (in a very simplified way) in 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').

The DNA operations proposed by Adleman and Lipton [2,3,11,12] are described below. These operations will be used for figuring out solutions of the independent set problem.

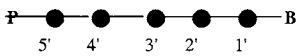


Figure 1: A schematic representation of a nucleotide.

The Adleman-Lipton's Model:

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

1. it 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 consist of the short strand S and $-(P, S)$ is all of the molecules of DNA in P which do not contain the short strand S.

2. *Separate.* Given a tube P and length L for a double strand of DNA, generate one tube $*(P, L)$, where $*(P, L)$ is all of the molecules of DNA in P which length is equal to L.

3. *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.

4. *Anneal.* The operation is to represent all of the operations that combine a test tube of single stranded DNA with other prepared strands and let them anneal together to form double strands.

5. *Detect.* Given a tube P, say 'yes' if P includes at least one DNA molecule, and say 'no' if it contains none.

6. *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.

7. *Discard.* Given a tube P, the operation will discard the tube P.

8. *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

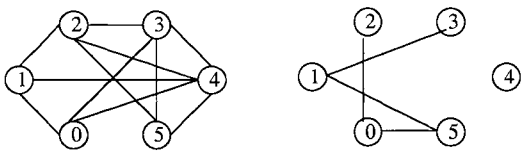
Techniques in the Adleman-Lipton model could be used to solve the NP-complete Hamiltonian path problem and satisfiability (SAT) problem in linearly increasing time and exponentially increasing volumes of DNA [2,3]. Ouyang et al. [4] showed that restriction enzymes could be used to solve the NP-complete 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 [4]. Arita et al. [5] described new molecular experimental techniques for searching a Hamiltonian path. Morimoto et al. [6] offered a solid-phase method to finding a Hamiltonian path. Narayanan et al. [7] proved that the Adleman-Lipton model was extended towards solving the travelling salesman problem. Shin et al. [8] presented an encoding scheme that applies fixed-length codes for representing integer and real values. Their method could also be employed towards ascertaining the travelling salesman problem. Amos [13] proposed parallel filtering model for resolving the Hamiltonian path problem, the sub-graph isomorphism problem, the 3-vertex-colorability problem, the clique problem and the independent-set problem. Roweis et al. [14] proposed sticker-based model to enhance Adleman's experiments. Their model could be used for determining solutions to an instance of the set cover problem. Perez-Jimenez et al. [15] employed sticker-based model to resolve knapsack problems. In our previous work, Chang and Guo [17,18,19,20] proved how the DNA operations presented by Adleman and Lipton could be employed for developing DNA algorithms to resolving the dominating-set problem, the vertex cover problem, the clique problem, the set cover problem, the problem of exact cover by 3-sets, the 3-dimensional matching problem and the set-packing problem.

3. DNA-Based Parallel Algorithm for Solving the Independent-set Problem

The independent-set problem asks: Given a network consisting of n vertices and m edges, how many vertices are in a maximum-size independent set? For example, the graph includes 6 vertices and 11 edges in Figure 2a. The graph defines such a problem. All of the independent set for the graph in Figure 2a includes $\{V_0\}, \{V_1\}, \{V_2\}, \{V_3\}, \{V_4\}, \{V_5\}, \{V_0, V_2\}, \{V_0, V_5\}, \{V_1, V_3\}$ and $\{V_1, V_5\}$. It is very obvious that the independent set of the maximum size to the graph contains $\{V_0, V_2\}, \{V_0, V_5\}, \{V_1, V_3\}$ and $\{V_1, V_5\}$. Hence, the size of the independent-set problem in this graph is two. Finding a maximum-size independent set is a NP-complete problem, so it can be formulated as a "search" problem [10]. Adleman [2,12] designed a biological system that produces all of the possible paths, and then he makes use of basic biological operations to select the Hamiltonian path from all of the possible paths. Lipton applies Adleman's model to generate all of the solutions for the satisfiability problem, and then he employs DNA operations proposed by Adleman to extract the answer from all of the solutions for the satisfiability problem. Their models enlighten how to resolve computational search problems (i.e., the NP-complete problems).

The first step of determining the independent-set problem is to yield a test tube, which consists of all of the possible independent sets. It is assumed that an undirected graph has n vertices, and an n-digit binary number represents each possible independent set. A bit set to 1 represents a vertex in the independent set,



(a) (b)

Figure 2: The graph of our problem, in which the independent-set problem is $\{V_0, V_2\}, \{V_0, V_5\}, \{V_1, V_3\}$ and $\{V_1, V_5\}$.

and a bit set to 0 represents a vertex out of the independent set. For example, the maximum-size independent set, $\{V_0, V_2\}$, for the graph in Figure 2a is represented by binary number 000101. By this way, all of the possible independent sets in an undirected n-vertex graph are transformed into an ensemble of all n-digit binary numbers.

To implement this way, we suggested that an unsigned integer X is represented by a binary number $x_{n-1}, x_{n-2}, \cdots, x_0$, where the value of x_i is 1 or 0 for $0 \leq i \leq n-1$. The integer X contains 2^n kinds of possible values. Each possible value represents an independent set. Therefore, it is very obvious that an unsigned integer X forms 2^n independent sets. A bit x_i in an unsigned integer X represents the i-th vertex in an undirected graph. If the i-th vertex is in an independent set, then the value of x_i is set to 1. If the i-th vertex is out of an independent set, then the value of x_i is set to 0.

To generate a complete test tube, we design the data structure that is the form of double-stranded DNA (dsDNA). It is assumed that x_i^1 denotes the value of x_i to be 1 and x_i^0 denotes to be 0. We encode all of the possible values for an unsigned integer X, forming all of the possible independent sets, into a test tube of double-stranded DNA as follows. First, we assign to each x_i^0 a random 20-mer sequence of DNA denoted O_i^0 .The random 20-mer sequence of DNA used in [2,12] should also suffice here. Next, we assign to each x_i^1 a random 30-mer sequence of DNA denoted O_i^1. The 30-mer sequence of DNA applied in [3,11] should also suffice here. The first half of O_i^0 or O_i^1 is denoted by c_i and the last half of O_i^0 or O_i^1 is also defined by d_i. Therefore, O_i^0 or O_i^1 can be represented by $c_i d_i$. Two adjacency bits, x_i and x_j, are regarded as one edge from the i-th position to the j-th position. For each edge from the i-th position to the j-th position, an oligonucleotide was created, where denotes the sequence that is the Watson-Crick complement of d_i and defines the sequence that is the Watson-Crick complement of c_j. Finally, a test tube is filled with the following kinds of DNA strands:

1. For each bit, put many copies of a $5' \rightarrow 3'$ DNA sequence of the form $c_i d_i$ into the test tube.

2. For each edge from the i-th position to the j-th position, place many copies of a $3' \rightarrow 5'$ DNA sequence of the form $\bar{d}_i\bar{c}_i$.

3. Put many copies of a $3 \rightarrow 5'$ sequence complementary to the first half of the initial bit (x_0). Similarly, put many copies of a $3' \rightarrow 5'$ sequence complementary to the last half of the last bit $(x_n - 1)$.

The oligonucleotides in the test tube were mixed together in a single ligation reaction in [2,3,11,12]. The $\bar{d}_i\bar{c}_i$ served as splints to bring oligonucleotides associated with compatible edges together for ligation. Therefore, the ligation reaction resulted in the formation of DAN molecules encoding all of the possible independent sets. After the test tube above is generated, we design the following genetic method to resolve a maximum-size independent set for any undirected graph.

Algorithm:

(1) Input(P), where the tube P is to encode all of the possible independent sets for given an n-vertex graph G with edges $e_1, e_2, \ldots, e_z$.

(2) **Forall** $k = 1$ to z, where z is the number of edges in the graph G.

(a) Let $e_k =< V_i, V_j >$, where e_k is one edge in the graph G and V_i and V_j are vertices in the graph G.

(b) Bits x_i and x_j, respectively, represent V_i and V_j.

(c) $Q^1 = +(P, x_i^1)$ and $Q = -(P, x_i^1)$.

(d) $Q^2 = +(Q^1, x_j^1)$ and $Q^3 = -(Q^1, x_j^1)$.

(e) $P = \cup(Q, Q^3)$.

(3) R^1 = Anneal (P), where the operation is to represent all of the operations that combine a test tube of single stranded DNA with other prepared strands and let them anneal together to form double strands.

(4) $P = *(R^1, l)$, where l is equal to the longest length for double strands of DNA in the tube R^1.

(5) If (detect (P) = 'yes') then.

(a) Read(P), where the operation describes 'sequence' of a molecular in the tube P.

Theorem 1 *In light of those steps in the Algorithm, the independent-set problem can be resolved with "polynominal" time complexity.*

Proof. Clearly every element of the input tube P corresponds to an independent set to a graph G. Any two vertices connected in the graph G cannot be members of the same independent set. That is to say that the corresponding bits cannot be set to 1. Therefore, Step (2) of the algorithm creates from the current tube a new tube from which any two vertices connected in the graph G have been removed. After $2 * z$ extractions, z merges, 1 anneal, 1 separation, 1 detection and

1 read, the algorithm will output 'sequence' of a molecule if at least one maximum-size independent set is contained in the tube P. Thus, the algorithm answers the independent-set problem for the graph G in 'linear' time (in the number of edges) which an electronic computer might require exponential time. This is the value of parallelism; each step acts on as many 2^n inputs simultaneously. □.

The genetic algorithm can be used to finding a maximum-size independent set for the graph in Figure 2a. Any two vertices in an independent set for the graph in Figure 2a are disconnected each other. This is to imply that the adjacency vertices of each edge in the graph in Figure 2a can not be the members of the same independent set. In light of Step (1) of the algorithm, the tube P is filled with 64 double strands of DNA, representing 64 possible independent sets for the graph in Figure 2a. All of the edges in the graph in Figure 2a are $< V_0, V_2 >, < V_0, V_3 >, < V_0, V_4 >, < V_1, V_2 >, < V_1, V_4 >, < V_2, V_3 >, < V_2, V_4 >, < V_2, V_5 >, < V_3, V_4 >, < V_3, V_5 >$ and $< V_4, V_5 >$. The number of the edges in the graph in Figure 2a is eleven, so the number of execution to Step (2) of the algorithm is 11 times. According to the first execution of Steps 2a and 2b of the algorithm, the first edge e_1 is $< V_0, V_2 >$ and the number containing the first edge is $***1*1$ ($*$ can be either 1 or 0). Therefore, in light of Steps 2c, 2d and 2e, the number $***1*1$ are removed from the tube P. Similarly, according to the remaining execution of Steps 2a to 2e, the numbers, representing other ten edges, also are removed from the tube P. Therefore, the remaining strands in the tube P are to represent legal independent sets. So, finding a maximum-size independent set is to search the longest length of strands in the tube P. Steps (3) and (4) of the algorithm are applied to find the answer from the tube P. Steps 5 and 5a of the algorithm read the answer from the tube P. Thus, a maximum-size independent set for the graph G in Figure 2a is $\{V_0, V_2\}, \{V_0, V_5\}, \{V_1, V_3\}$ or $\{V_1, V_5\}$.

4. Experimental Results of Simulated DNA Algorithm

We developed a tool for simulating DNA computing and other molecular computation methods. The tool is applied towards designing the biological experiments, such as code design for resolving the independent-set problem. The operations presented by Adleman and Lipton in Section 2 was implemented on the tool. Simultaneously, the parallel genetic algorithm proposed in Section 3 was also implemented on the tool.

In simulations, the rule of code design for resolving the independent-set problem is equal to that of vertex color. It is assumed that four bases, $\{A, C, G, T\}$, represents four different colors. The same base (color) did not construct two adjacency nucleotides in single-stranded DNA. To avoid common nucleotides in different strands, if a kind of color sequence was used to generate a new strand, then it was removed for other strands and hence could not be employed to yield other new strands. According to the rule, single strand of DNA for every vertex (i.e., the

corresponding bit) in the graph in Figure 2a is shown in Table 1. Similarly, single strand of DNA to each edge, regarded as the splint, is also shown in Table 2. A $3' \rightarrow 5'$ sequence complementary to the first half of the initial bit (x_0^0) and a $3' \rightarrow 5'$ sequence complementary to the first half of the initial bit (x_0^1) are, respectively, $ACTCGTATAC$ and $AGTGAGACATGATCG$. Similarly, a $3' \rightarrow 5'$ sequence complementary to the last half of the last bit (x_{n-1}^0) and a $3' \rightarrow 5'$ sequence complementary to the last half of the last bit (x_{n-1}^1) are, respectively, $CTGCTCTACG$ and $GAGTCGTCATGAGCA$.

Vertex	$5' \rightarrow 3'$ DNA Sequence
x_0^0	$TGAGCATATGCATGTCGATC$
x_0^1	$TCACTCTGTACTAGCGCTATCGTCATAGTA$
x_1^0	$GATCAGCACTGCTAGTCACA$
x_1^1	$ATCGACGCATCGCTCACGTGTGCGTGCTGC$
x_2^0	$TAGTGTAGCGAGATATGTCT$
x_2^1	$GCGATGTCGCTCGTGCAGCGAGACGAGACA$
x_3^0	$AGCACTGAGTATCACTACGT$
x_3^1	$GTATCGATCAGTGATGTGACATAGAGTAGA$
x_4^0	$CACTATCTGATAGCGAGCTA$
x_4^1	$ACTGTACGAGTGTCTATAGACTCAGTGCAC$
x_5^0	$CGTAGCACTCGACGAGATGC$
x_5^1	$CTGCGAGTACACACACTCAGCAGTACTCGT$

Table 1: Sequences chosen to represent the vertices in the graph in Figure 2a.

In simulations, 32 copies of DNA sequences, representing every bit in Table 1 or every edge in Table 2, are generated. Similarly, 32 copies of a $3' \rightarrow 5'$ sequence complementary to the first half of the initial bit (x_0^0) and 32 copies of a $3' \rightarrow 5'$ sequence complementary to the first half of the initial bit (x_0^1) are also yielded. Next, 32 copies of a $3' \rightarrow 5'$ sequence complementary to the last half of the last bit (x_{n-1}^0) and 32 copies of a $3' \rightarrow 5'$ sequence complementary to the last half of the last bit (x_{n-1}^1) are also produced. The hybridization process and the ligation process were emulated as realistic as biological processes. Therefore, the tube for the graph in Figure 2a was generated in simulations. The tube contained 64 different double strands of DNA. That is to say that the tube consists of 64 different independent sets.

In simulations, a file was used to store 64 different sequences of DNA. In the genetic algorithm, Step 1 of simulation is to read the sequences from the file and the sequences were stored in arrays. In light of the number of the edges in the graph G in Figure 2a, the number of the execution for Step 2 is equal to the number of the edges. In Steps 2(a) and 2(b) of simulation, two different sequences, representing adjacency vertices of the same edge, were found from Table 1 and were applied for removing illegal independent sets. In Steps 2(c) and 2(d) of simulation, the first

sequence was regarded as a sub-string and was employed to separate the solutions into two parts. The solutions in the first part did not contain the independent sets, which include two adjacency vertices of the same edge. The solutions in the second part consisted of the independent sets, which contain two adjacency vertices of the same edge. Because the solutions in the second part were illegal, they were discarded. In Step 2(e) of simulation, all of the legal solutions produced by Steps 2(c) and 2(d) were stored in arrays. Repeat the execution of Step 2 until all of the edges in the graph G were processed. Since the legal solutions generated by Steps 2(a) to 2(e) were single strands of DNA, the aim of Step 3 is to form double strands of DNA to the legal solutions. Therefore, this process was simulated by explicitly producing the corresponding complementary sequences for the single strands of DNA. In Step 4, the longest sequence from the legal solutions is chosen as a maximum-size independent set. The longest sequence is a maximum-size independent set because in the proposed encoding scheme the code length of V_i was set to 30 if the value of V_i is equal to 1. Hence this process was simulated by explicitly finding the longest sequence from the legal solutions. The goal of Step 5 is to check whether there are the longest sequences in the legal solutions. Thus, this process was simulated by explicitly examining whether the set of the legal solutions is empty. This process generated a resulted value, *'yes'*, if the set of the legal solutions is non-empty. A resulted value, *'no'*, was generated by this process if the set of the legal solutions is empty. If this process generated a resulted value, *'yes'*, then in Step 5(a) of simulation, a legal solution from the set of the legal solutions was selected. If a 30-mer sequence in Table 1 was the sub-string of the selected solution, then the corresponding vertex was in a maximum-size independent set. Otherwise, the corresponding vertex was out of a maximum-size independent set.

5. Conclusions

The main result of this paper is that DNA based computers can be employed for resolving the independent-set problem to any undirected graph. The design of DNA codes and each operation in the algorithm proposed are finished through computer simulations. The size of all of the possible independent sets is 2^n, where n is the number of vertices in any undirected graph. Due to the limit of memory space and hard-disk space, the value of n should be less than or equal to 22 in the simulation.

The genetic algorithms are grouped in two ways [11]. First, the genetic algorithms are classified in light of how the volume changes during the molecular computation. *Decreasing Volume Algorithms* decrease the number of strands in a test tube when the algorithms execute, in *Constant Volume Algorithms*, the number of strands constant is maintained throughout the computation, and *Mixed Algorithms* are those fitting in neither of the previous classes. Second, an algorithm is said to be *Uniform* if the following condition is satisfied in every test tube during the computation: any two different strands have the same number of copies in the test tube. According to [11], dealing with a constant volume and uniform algorithm is

Splint	$3' \rightarrow 5'$ DNA Sequence
(x_5^0, x_4^0)	$TCATCTCGTAACAGCGAGTC$
(x_5^0, x_4^1)	$TCATCTCGTAACAGCGAGTC$
(x_5^1, x_4^0)	$AGACTACTGCAGATGACAGCGAGTC$
(x_5^1, x_4^1)	$AGACTACTGCAGATGCAGTGCATCTGTGAG$
(x_4^0, x_3^0)	$GCTATCTAGCCTACAGTCTG$
(x_4^0, x_3^1)	$GCTATCTAGCTGCGATCGACTGTCG$
(x_4^1, x_3^0)	$CGCTCAGACTGTACACTACAGTCTG$
(x_4^1, x_3^1)	$CGCTCAGACTGTACATGCGATCGACTGTCG$
(x_3^0, x_2^0)	$CGACAGCATGGCTGTGCTAT$
(x_3^0, x_2^1)	$CGACAGCATGTATCGTGATAGATGTC$
(x_3^1, x_2^0)	$TGTCACGCTCTGCTCGCTGTGCTAT$
(x_3^1, x_2^1)	$TGTCACGCTCTGCTCTATCGTGATAGATGT$
(x_2^0, x_1^0)	$CTCGCGTGAGTCGACTACAG$
(x_2^0, x_1^1)	$CTCGCGTGAGCGATCATACGATAGA$
(x_2^1, x_1^0)	$ACTATCTCATCTCACTCGACTACAG$
(x_2^1, x_1^1)	$ACTATCTCATCTCACCGATCATACGATAGA$
(x_1^0, x_0^0)	$TAGCTGACACGTCTACGCGT$
(x_1^0, x_0^1)	$TAGCTGACACGACAGAGTGCAGCTA$
(x_1^1, x_0^0)	$CATGTGTATGTAGTAGTCTACGCGT$
(x_1^1, x_0^1)	$CATGTGTATGTAGTAGACAGAGTGCAGCTA$

Table 2: Sequences chosen are regarded as the splint for adjacency bits in Table 1

easier than manipulating other algorithms. Because the algorithm proposed is uniform and constant volume, it is shown from the simulation that the implementation to a constant volume and uniform algorithm is easier than that of other algorithm

[1] R. R. Sinden. DNA Structure and Function. Academic Press, 1994.

[2] L. Adleman. Molecular computation of solutions to combinatorial problems. Science, 266:1021-1024, Nov. 11, 1994.

[3] R. J. Lipton. DNA solution of hard computational problems. Science, 268:542:545, 1995.

[4] Q. Ouyang, P.D. Kaplan, S. Liu, and A. Libchaber. DNA solution of the maximal clique problem. Science, 278:446-449, 1997.

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

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

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

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

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Introduction to algorithms. [10] M. R. Garey, and D. S. Johnson. Computer and intractability. Freeman, San Fransico, CA, 1979.

[10] M. R. Garey, and D. S. Johnson. Computer and intractability. Freeman, San Fransico, CA, 1979.

[11] D. Boneh, C. Dunworth, R. J. Lipton and J. Sgall. On the computational Power of DNA. In Discrete Applied Mathematics, Special Issue on Computational Molecular Biology, Vol. 71 (1996), pp. 79-94.

[12] L. M. Adleman. On constructing a molecular computer. DNA Based Computers, Eds. R. Lipton and E. Baum, DIMACS: series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society. 1-21 (1996)

[13] M. Amos. "DNA Computation", Ph.D. Thesis, department of computer science, the University of Warwick, 1997.

[14] S. Roweis, E. Winfree, R. Burgoyne, N. V. Chelyapov, M. F. Goodman, Paul W.K. Rothemund and L. M. Lipton. "A Sticker Based Model for DNA Computation". 2nd annual workshop on DNA Computing, Princeton University. Eds. L. Landweber and E. Baum, DIMACS: series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society. 1-29 (1999).

[15] M.J. Perez-Jimenez and F. Sancho-Caparrini. "Solving Knapsack Problems in a Sticker Based Model". 7nd annual workshop on DNA Computing, DIMACS: series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 2001.

[16] G. Paun, G. Rozenberg and A. Salomaa. DNA Computing: New Computing Paradigms. Springer-Verlag, New York, 1998. ISBN: 3-540-64196-3.

[17] W.-L. Chang and M. Guo. "Solving the Dominating-set Problem in Adleman-Lipton's Model". In the Proceedings of Third International Conference on Parallel and Distributed Computing, Applications and Technologies, Japan, 2002.

[18] W.-L. Chang and M. Guo. " Solving the Clique Problem and the Vertex Cover Problem in Adleman-Lipton's Model". In the Proceedings of IASTED International Conference, Networks, Parallel and Distributed Processing, and Applications, Japan, 2002.

[19] W.-L. Chang and Minyi Guo. "Solving the Set Cover Problem and the Problem of Exact Cover By 3-Sets in the Adleman-Lipton Model", The Third International Conference on Unconventional Models of Computation, Japan, 2002.

[20] W.-L. Chang and M. Guo. " Resolving the 3-Dimensional Matching Problem and the Set Packing Problem in Adleman-Lipton's Model". In the Proceedings of IASTED International Conference, Networks, Parallel and Distributed Processing, and Applications, Japan, 2002.