DNA Solution of a Graph Coloring Problem

Yachun Liu,*[†] Jin Xu,[‡] Linqiang Pan,[‡] and Shiying Wang[‡]

Department of Mathematics and Physical Science, Nanhua University Hengyang, Hunan, 421001 P. R. China, and Department of Control Science and Engineering, Huazhong University of Science & Technology Wuhan, Hubei, 430074 P. R. China

Received February 28, 2001

The graph-theoretic parameter that has probably received the most attention over the years is the chromatic number. As is well-known, the coloring problem is an NP−Complete problem. In this paper, it has been solved by means of molecular biology techniques. The algorithm is highly parallel and has satisfactory fidelity. This work shows further evidence for the ability of DNA computing to solve NP−Complete problems.

1. INTRODUCTION

DNA (deoxyribonucleic acid) is a double stranded sequence of four nucleotides; the four nucleotides that compose a strand of DNA are as follows: adenine (A), guanine (G), cytosine (C), and thymine (T); they are often called bases. The chemical structure of DNA (the famous double helix) was discovered by James Watson and Francis Crick in 1953. It consists of a particular bond of two linear sequences of bases. This bond follows a property of complementarity: adenine bonds with thymine and vice versa, cytosine bonds with guanine and vice versa. This is known as Watson−Crick complementarity and it is also denoted by over bars: $\bar{A} = T$, $\bar{T} = A$, $\bar{C} = G$, $\bar{G} = C$. Each DNA strand has two different ends that determine its polarity: the 3′ end, and the 5′ end. The double helix is an antiparallel (two strands of opposite polarity) bonding of two complementary strands.

Molecular biology is used to suggest a new way of solving a NP-complete problem. The idea (due to Leonard Adleman) is to use strands of DNA to encode the (instance of the) problem and to manipulate them using techniques commonly available in any molecular biology laboratory, to simulate operations that select the solution of the problem, if it exists. After Adleman's paper appeared in *Science* in November 1994—many authors have been interested in DNA computing. DNA computing must not be confused with biocomputing. Usually, biocomputing means everything that computer scientists can do to help biologists to study genes. For example, algorithms and data structures have been developed to investigate the properties of the sequences of nucleotides in DNA or RNA and those of amino acids in the primary structure of a protein. In DNA computing, instead, molecular biology is suggested to solve problems for computer scientists. In the past years, many techniques have been developed in order to study and manipulate DNA in a lab, for various biological applications. The advances in molecular biology technology allow recombinant DNA operations to be routine in all the molecular biology laboratories, while these techniques were once considered very sophisticated. We now describe some of the operations that are available on DNA strands. The aim of the following description (that involves various kinds of operations) is to give the reader an idea of the possible implementation of the algorithms that appear in DNA computing, which we will describe in the following sections:

- *Annealing and melting:* the hydrogen bonding between two complementary sequences is weaker than the one that links nucleotides of the same sequence. It is possible to pair (anneal) two antiparallel and complementary single strands, and it is possible to separate (melt) them, obtaining two single strands from a double one. These operations are realized by creating adequate conditions of temperature, pH, etc.
- *Synthesis:* A desired strand of DNA can be synthesized in lab. This is possible for strands up to a certain length. Longer 'random' strands are available. They consist of DNA sequences that have been cloned from many different organisms. They can be easily produced in large quantities.
- *Extraction:* Given a test tube T and a strand s, it is possible to extract all the strands in T that contain s as a subsequence and to separate them from those that do not contain it.
- *Union:* Given two or more test tubes, say T_1, T_2, ⋯, T_n, it is possible to put in a new test tube the union of all the strands contained in T_1, T_2, ⋯, T_n.
- *Amplification:* The content of a test tube can be amplified, giving a duplication of all the strands in it.
- *Polymerize:* A single strand that has a portion of double stranded subsequence can be polymerized. The result is an entire double stranded molecule (the opportune complementary sequences are synthesized and annealed to the original strand).
- *Ligation:* There is an enzyme called Ligase that causes the concatenation of two sequences in a unique strand.
- *PCR:* It is an acronym for Polymerize Chain Reaction; it is not only an operation but also a machine that allows realizing some operations on DNA strands. One of them is the amplification, another one allows to select, in a test tube, all of the strands that begin (or end) with a given subsequence.
- *Detect:* Given a test tube T, we denote with detection the lab operation that checks whether T contains at least a DNA strand.

* Corresponding author e-mail: liuyachun65@263.net.
[†] Nanhua University Hengyang.
[‡] Huazhong University of Science & Technology Wuhan.

• *Length:* With a technique that uses a particular gel, it is possible to separate strands of different length.

• *Restriction enzymes:* There are many kind of enzymes that are the molecules capable of operating on other molecules. Some of them are the restriction enzymes that cut DNA double strands where specific subsequence appears.

Computer scientists rank computational problem in three classes:[1] easy, hard, and uncomputable. In 1994, Adleman[2] showed that DNA can be used to solve a computationally hard problem—the directed Hamiltonian Path Problem (usually called HPP problem), and demonstrated the potential power of parallel, high-density computation by molecules in solution. This parallelism allows DNA computers to solve larger hard problems such as NP−Complete problems in linearly increasing time, in contrast to the exponentially increasing time required by a Turing machine. The tradeoff is that DNA computers require exponentially increasing volumes of DNA. As known, the HPP problem is NP−Complete. Thus, as long as nobody demonstrates that $P \neq NP$, there is no polynomial deterministic algorithm that solves it. Anyway, there exist nondeterministic polynomial algorithms. The following is one of them: Given a graph G = (V, E) and two designated vertices v_{in}, $v_{out} \in V$

1. Generate random path through G;
2. Keep only those paths that begin in v_{in} and end in v_{out};
3. Keep only the paths that enter exactly in n vertices (where $n = |V|$);
4. Keep only paths that enter each vertex at least once.
5. Detect if any paths survived the selection. If so, the answer is ‘Yes’, otherwise the answer is ‘no’.

This algorithm is the one that Adleman realized in a molecular biology lab. The trick of his implementation lies in the way he suggests to encode the instance of Hamiltonian Path Problem: for every vertex i in the graph G, a random strand of DNA of length 20 base pairs (bp) is synthesized and denoted with O_i. Then, for every edge ($i \rightarrow j$) in G, an opportune strand is synthesized: the 3′ half of O_i and 5′ half of O_j are concatenated. In this way, when we put in a test tube, for example, copies of the oligonucleotides $O_{i\rightarrow j}$, $O_{j\rightarrow k}$ and $\overline{O_j}$, the strands will anneal, creating double strands that encode paths through the graph (one of the two strands is the concatenation of the substrands that encode adjacent edges, and the other one is the concatenation of the complementary oligos associated with the vertices which the path enters).

Although the HPP problem is an NP−Complete one, Adleman’s result does not demonstrate the feasibility of solving all instances of NP−Complete problems. In 1995, Lipton[3] extended Adleman’s idea to solve the problem of the satisfiability of a Boolean formula (usually called SAT problem). The SAT problem consists of finding Boolean values to n variables $x_1, x_2, \cdots, x_n$ that make F true. Here, F is assumed to be of the form

$$F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

where each C_i is a clause of the form $v_1 \vee v_2 \cdots \vee v_k$, and each v_j is one of the n variables or its negation. Lipton suggests a molecular way of solving this problem (see ref 3).

In 1997, Qi Ouyang[4] et al. solved the maximal clique problem by means of molecular biology techniques. A pool of DNA molecules corresponding to the total ensemble of all possible cliques was built followed by a series of selection processes. Here, we present a molecular biology-based experimental solution to the coloring problem.

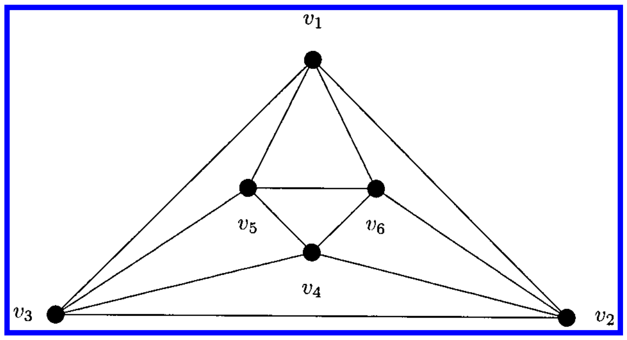


Figure 1. Octahedron − the graph of color problem

Mathematically, an assignment of colors (elements of some set) to the vertices of a graph G (one color to each vertex, so that adjacent vertices are assigned different colors) is called a coloring of G; a coloring in which n colors are used is an n-coloring. A graph G is n-colorable if there exists an m coloring of G for some $m \leq n$. It is obvious that if G has order p, then G can be p-colored, so G is p-colorable. The minimum n for which a graph G is n-colorable is called the vertex chromatic number, or simply the chromatic number of G, and is denoted by $c(G)$. If G is a graph for which $c(G) = n$, then G is n-chromatic. In a given coloring of a graph G, a set consisting of all those vertices assigned the same color is referred to as a color class. The color problem is: Given a graph containing N vertices and M edges, how can the graph G be colored by the minimum chromatic number of colors? And what is $c(G)$? The graph G of six vertices and twelve edges in Figure 1 defines such a problem. It is easy to prove that $c(G) = 3$, and $\{v_1, v_4\}$,$\{v_2, v_5\}$, $\{v_3, v_6\}$ are three different color classes. In fact, this graph is one of the five regular polyhedral, i.e., the octahedron. Note that taking example of octahedron is only for convenience and provides no special advantage in our computation.

2. THE CONSTRUCTION OF DNA DATA POOL

There are well-known algorithms for graph coloring problem. However, all known algorithms for this problem have exponential worst-case complexity, and hence there are instances of modest size for which these algorithms require an impractical amount of computer time to render a decision. Because the graph coloring problem has been proved to be NP-complete, it seems likely that there is no efficient (that is, polynomial time) algorithm to solve it. We designed the following nondeterministic algorithm to solve it:

(1) For a graph G with N vertices, each possible coloring scheme is represented by a rearrangement of N elements. A bit set to i represents that the corresponding vertex is assigned the ith color. For example, the rearrangement (2, 1, 5, 3, 6, 5) denotes that vertex v_1, v_2, v_3, v_4, v_5, v_6 is colored by the second, the first, the fifth, the third, the sixth, and the fifth color, respectively. (1, 2, 3, 1, 2, 3), (2, 3, 4, 2, 3, 4), and so on represent 3-chromatic schemes. Whereas (1, 1, 3, 1, 2, 3) represents an illegal coloring of octahedron. In this way, we transform the complete set of all possible coloring schemes in an N-vertex graph into an ensemble of all rearrangements of N elements. We call this the complete data pool. Obviously, the cardinality of this ensemble is N^N.

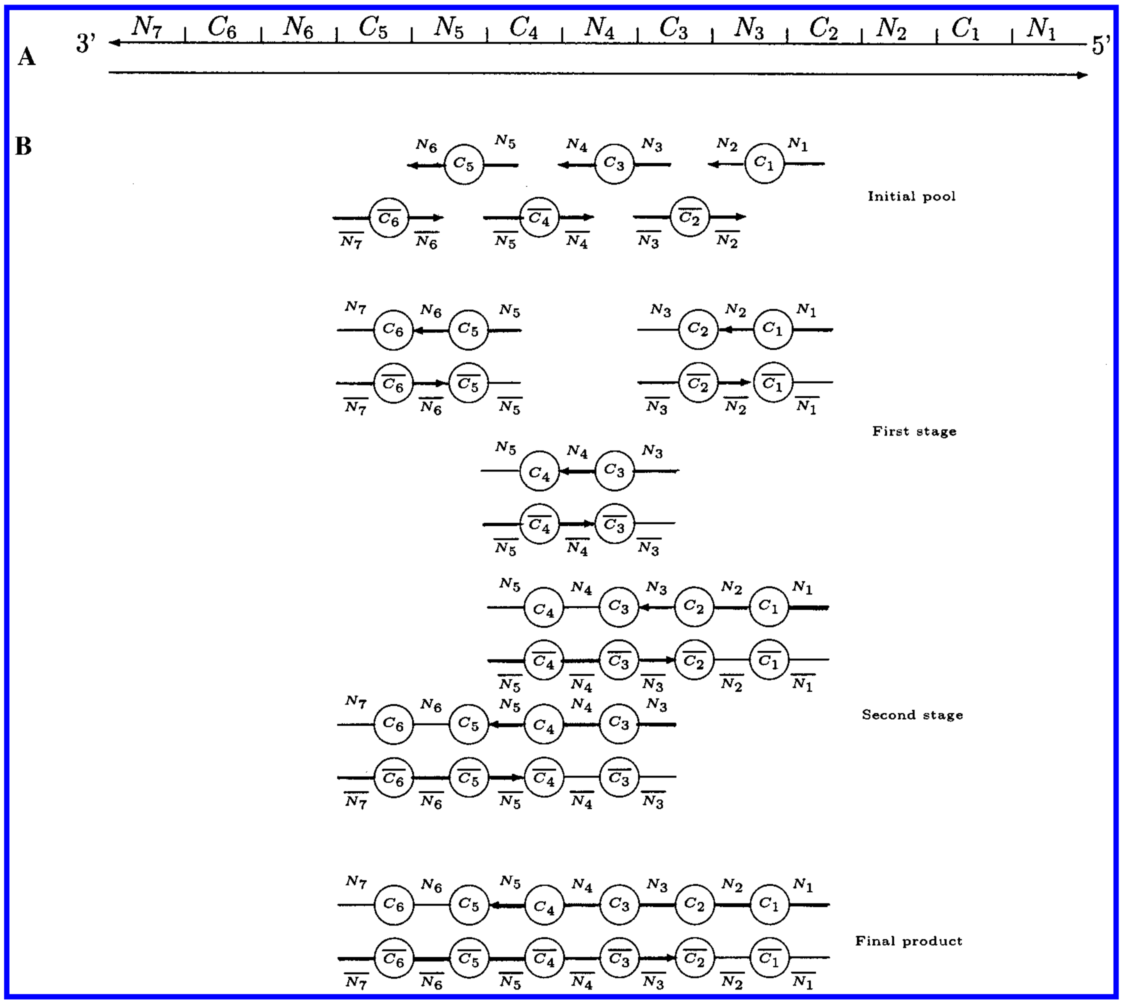

A
3'
5'
N_7 C_6 N_6 C_5 N_5 C_4 N_4 C_3 N_3 C_2 N_2 C_1 N_1
B
Initial pool
First stage
Second stage
Final product

Figure 2. (A) The data structure of a six-digit rearrangement in DNA. (B) Assembling rearrangements by overlap extension. A full string of DNA in (A) is recursively assembled from short pieces by 3′ end hybridization and extension. The name and color string of DNA are denoted by over bars and circles, respectively. The heavy arrows represent the ssDNA entering each stage of the reaction; the light lines represent extension.

(2) If two vertices connected by an edge in graph *G*, then they cannot be members of a color class. This means that the corresponding bits cannot both be set to the same integral number $i(1 \leq i \leq N)$. So we eliminate from the complete data pool all rearrangements in which repeated numbers present in the positions of adjacent vertices. For our problem, the rearrangements (*x*, *x*, *, *, *, *), (*x*, *, *x*, *, *, *), (*x*, *, *, *, *x*, *), (*x*, *, *, *, *, *x*); (*, *x*, *x*, *, *, *), (*, *x*,*, *x*, *, *), (*, *x*, *, *, *, *x*); (*, *, *x*, *x*, *, *), (*, *, *x*, *, *x*, *); (*, *, *, *x*, *x*, *), (*, *, *, *x*, *, *x*); (*, *, *, *, *x*, *x*) are removed, where $x \in \{1, 2, 3, 4, 5, 6\}$, wild-card * represents any integral number from 1 to 6.

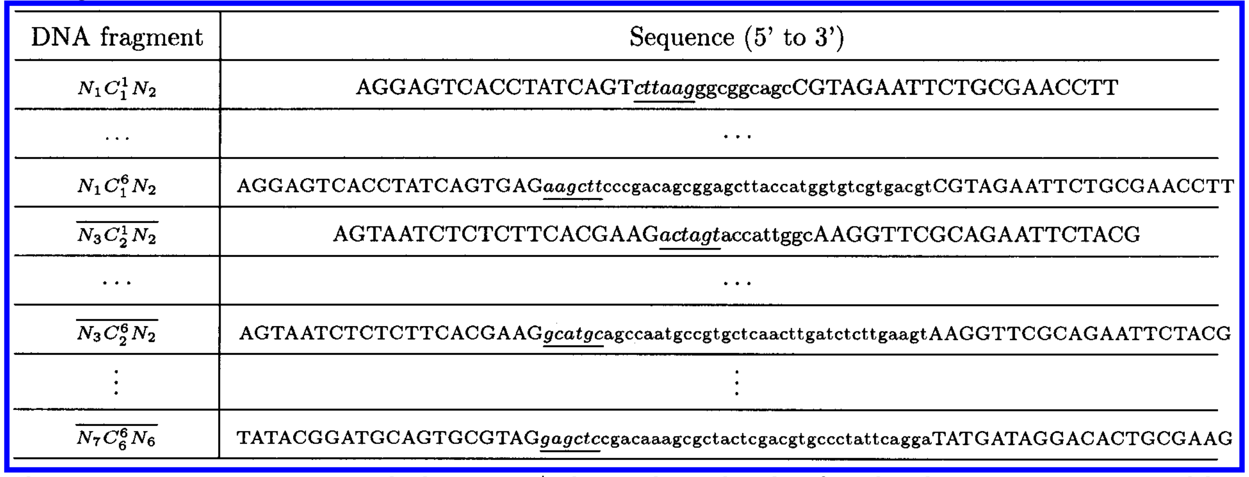
(3) We sort the remaining data pool to find the data containing the minimum number of colors. The rearrangements with the minimum number of colors tell us the chromatic number. For our problem, the data (1, 2, 3, 1, 2, 3), (2, 1, 3, 2, 1, 3), or (3, 1, 2, 3, 1, 2) will be selected out.

The first task in DNA computing is to construct an ensemble of DNA molecules to represent the complete data pool of the coloring problem. We design the data structure in the form of double stranded DNA (dsDNA). Each bit in a rearrangement is represented by two DNA sections N_vC_v (using the vertex *v*'s name N_v and its color C_v). For a DNA molecule representing a six-digit rearrangement, there are six color sections (C_1 to C_6) sandwiched sequentially between seven name sections (N_1 to N_7) (see Figure 2A). The last artificial name section N_7 is needed for polymers-chain-reaction (PCR) amplification. To work well, the length of N_i was set to 20 base pairs (bp). The length of C_i was set to $10 + 5j$ bp if $C_i = j(j = 1, 2, \cdots, 6)$.

Therefore, the longest DNA has 380-bp corresponding to the rearrangement 666666, and the shortest DNA has 230-bp corresponding to the rearrangement 111111. This design is simple and effective, because we only need to know the length of the answer. The oligonucleotide sequences of each N_i and C_i is first randomly generated. However, to avoid mispairing during data assembly, care was taken to avoid accidental homologies longer than 4 bp. For future convenience, we then embedded restriction sequences within each $C_i = j$, where $i, j \in \{1, 2, 3, 4, 5, 6\}$.

We use parallel overlap assembly (POA) to construct our DNA data pool. The construction starts with the 36 oligonucleotides partly listed in Table 1. Each oligonucleotide consists of two name motifs and one color motif, $N_iC_iN_{i+1}$ for odd *i* and $\overline{N_{i+1}C_iN_i}$ for even *i*, where the over bar represents the complementary sequence and the value of C_i can be 1, 2, $\cdots$, 6. The 36 fragment oligonucleotides are mixed together for thermal cycling. During each thermal cycle, the name strings in one oligonucliotide are annealed

Table 1. Oligonucleotides To Construct the DNA Data Pool[a]



DNA fragment	Sequence (5' to 3')
$N_1C_1^1N_2$	AGGAGTCACCTATCAGT$\underline{cttaag}$ggcggcagcCGTAGAATTCTGCGAACCTT
...	...
$N_1C_1^6N_2$	AGGAGTCACCTATCAGTGAG$\underline{aagctt}$cccgacagcggagcttaccatggtgtcgtgacgtCGTAGAATTCTGCGAACCTT
$\overline{N_3C_2^1N_2}$	AGTAATCTCTCTTCACGAAG$\underline{actagt}$accattggcAAGGTTCGCAGAATTCTACG
...	...
$\overline{N_3C_2^6N_2}$	AGTAATCTCTCTTCACGAAG$\underline{gcatgc}$agccaatgccgtgctcaacttgatctcttgaagtAAGGTTCGCAGAATTCTACG
$\vdots$	$\vdots$
$\overline{N_7C_6^6N_6}$	TATACGGATGCAGTGCGTAG$\underline{gagctc}$cgacaaagcgctactcgacgtgccctattcaggaTATGATAGGACACTGCGAAG

[a] Each string contains name sequences N_i and color strings C_i^j, where j indicates the value of C_i. The color sequences are written with lowercase letters; underlining indicates restriction enzyme sites. Thirty-six different kinds of restriction enzymes were applied to cut C_1^1, C_1^2, $\cdots$, C_1^6; C_2^1, C_2^2, $\cdots$, C_2^6; $\cdots$, C_6^1, C_6^2, $\cdots$, C_6^6, respectively.

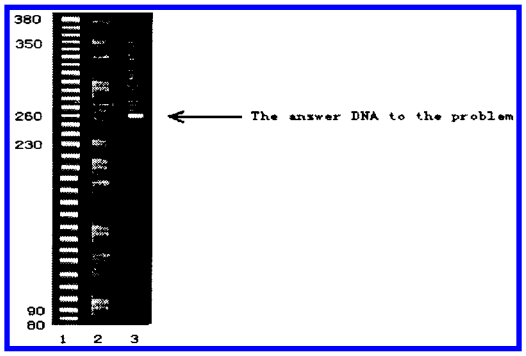


Figure 3. Electrophoresis of products after each operation. Lane 1, marker; lane 2, product of POA and PCR; lane 3, product of final PCR (seven cycles) after taking process of S1 nuclease digestion and series restriction enzyme digestion. The band indicated by an arrow in lane 3 is the answer to the problem.

to the complementary strings of the next oligonucleotide. The 3′ ends extended in the presence of polymerize to form a longer dsDNA. After a few thermal cycles, the initial DNA data pool with all combinations of $C_1C_2C_3C_4C_5C_6$ is built (Figure 2B).

The POA procedure is followed by PCR with a small fraction of POA product serving as template and with N_1 and $\bar{N}_7$ acting as primers. Thus, only those molecules with v_1 and v_7 at their ends are exponentially amplified. The results of POA and PCR, denoted by T_0, are shown in Figure 3, lane 2. Then, we digest the data pool T_0 with restriction enzymes. These enzymes break DNA at specific restriction sites, which we embed within the sequences $C_i^j(i, j \in \{1, 2, 3, 4, 5, 6\})$. The broken strings are not amplified exponentially by PCR with primers N_1 and $\bar{N}_7$. To cut a coloring that adjacent vertices belong to a same class—for example, the rearrangement (1, 1, *, *, *, *) (Figure 1)—we first divide the data pool into two test tubes, T_1 and T_2. In T_1 we cut strings containing $C_1 = 1$ with Afl II; in T_2 we cut strings containing $C_2 = 1$ with Spe I. Next, we combine T_1 and T_2 into test tube T_0', which does not contain (1, 1, *, *, *, *). Five sequential restriction operations (with different enzymes) eliminate all strings corresponding to data (2, 2, *, *, *, *), (3, 3, *, *, *, *), (4, 4, *, *, *, *), (5, 5, *, *, *, *), (6, 6, *, *, *, *). The remaining data DNA does not contain (x, x, *, *, *, *),where $x = 1, 2, \cdots, 6$. In another word, the adjacent vertices v_1 and v_2 in Figure 1 belong to different color classes. Repeating above procedures for every other edges, the final data DAN is then amplified by PCR. The results of restriction digestions and the PCR amplification are shown in Figure 3, lane 3. Reading the chromatic number is straightforward. In our data structure, the shortest length of DNA represents the chromatic number, and thus the lowest band (arrow in Figure 3, lane 3) is the answer. The length of this band, 260-bp, tells us that $c(G)$ must be 3. If asked what is the coloring of each vertex in the graph G, you may feel rough handling, because the molecules in Figure 3, lane 3, could be any of 90 different DNA strings. To demonstrate the answer by DNA molecular biology techniques simply, not by pure mathematical methods, we first extract 260-bp strings (the answers DNA) and then read the answer by molecular cloning.[4] The DNA of the answer is inserted into M_{13} bacteriophage (Phagescript, Stratagene) through site-specific mutagenesis. The mutagenized M_{13} Phage DNA (containing our answer DNA) is transfected into *E. coli* bacteria (XI-1 Blue, Stratagene), cloned, and its DNA extracted and sequenced, the answer of a computation follows that the color numbers of v_1 and v_4 are the same, so are v_2 and v_5, v_3, and v_6.

3. SOME COMMENTS ON THE COMPUTATION

The major errors in this calculation come from two sources. The first source of error is the production of single-stranded DNA (ssDNA) during PCR. This ssDNA cannot be cut by restriction enzymes. We avoid this error by digesting the ssDNA with S_1 nuclease before restriction digestions. This operation largely prevents false positive results. The second source of error is incomplete cutting by restriction enzymes, which also leads to incorrect answers. The combination of restriction digestion and PCR forms an exponential amplifier with a larger exponent for uncut strands than for cut strands. Repeating the digestion-PCR process should therefore increase the signal-to-noise ratio arising from incomplete digestion. We use one or two cycles of digestion-PCR and have found no qualitative difference. The selected restriction enzymes work well enough for our purpose.

Errors also possibly come from extraction and anneal operations. Let the reader imagine what would happen if an 'object' DNA were lost during one of the extractions in the experiment. It is for this reason we suggest to amplify (more than once) the content of test tube at each step of the experiment, in this way, the loss of a strand that encodes all possible considered colorings would not be catastrophic, because one of its copies should survive. The opposite error is less dangerous, because the solution could be verified at the end of the computation. Undesired annealing errors mean that a strand s could anneal with one that is similar to $\bar{s}$, but it is not the right one. In this case, we call them *partial matches*. Another kind of undesired matching could happen between two 'shifted' strands: for example, a strand xy could partially anneal with a strand $\bar{y}z$. Finally, a strand could anneal with itself, losing its linear structure. The probability of all these undesired annealing could be decreased with an opportune choice of the strands used to encode the data of the problem.

As correctly stated by Adleman, the information storage capacity of DNA is huge. In principle, 1 μmol of DNA can encode 2 gigabytes of information. However, rapid and accurate data access is needed to take advantage of massive parallelism. The major advantage of DNA computing lies in its high parallelism. If the ligation of two DNA molecules is considered as a single operation, at micromole scale, the number of operations per second during the ligation step would exceed that of current supercomputers by more than a 1000-fold. However, the number of vertices that this algorithm can handle is limited. The maximum number of vertices we can process with picmole operations is limited to 24 because of the exponential increase in the size of the pool with the size of the problem. With nanomole chemistry, we could have 30 vertices. Further scale-up quickly becomes impractical. New algorithms, resembling in vitro evolution in which the initial data pool need not contain every possible final answer, are needed.

ACKNOWLEDGMENT

This work is partially supported by a grant from the National Science Foundation of China. The authors sincerely appreciate the encouraging comments of the Editor of the journal on this paper. They also wish to thank an anonymous referee of this paper who provided many useful and constructive suggestions for the improvement of the paper. Finally, thanks to all the authors that appear in the references.

REFERENCES AND NOTES

(1) Casti, J. The complexity of a combinational problem. *New Sci.* **1997**, *2082*, 30.
(2) Adleman, L. M. Molecular computation of solutions to combinatorial problems. *Science* **1994**, *266*, 11 November.
(3) Lipton, R. J. DNA solution of hard computational problem. **1995**, *Science 268*, 28 April.
(4) Ouyang, Q.; Kaplan, P. D.; Liu, S.; Libchaber, A. DNA solution of the Maximal clique problem. *Science* **1997**, *278*, 17 October.
(5) Guarnieri, F.; Fliss, M.; Bancroft, C. Making DNA add. *Science* **1996**, *273*, 12 July.

CI010016O