A DNA Algorithm for the Graph Coloring Problem

Wenbin Liu,*,†,‡ Fengyue Zhang,† and Jin Xu†

Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan City 430074, China, and Department of Information Science and Engineering, Shandong University of Science and Technology, Taian City 271019, China

Received June 3, 2002

A DNA algorithm based on surfaces for the graph coloring problem is presented. First the whole combinatorial color assignments to the vertices of a graph are synthesized and immobilized on a surface; then a vertex is legally colored while those adjacent to it with illegal colors are deleted; and the cycle is repeated until finally the correct color assignments to the graph are reached. Compared with the other DNA algorithms, our algorithm is easy to implement and error-resistant.

1. INTRODUCTION

The concept of DNA computing was first proposed by L. Adleman in 1994.[1] The potential power of massive parallelism and high information density of biomolecules offers a novel method to solve complex computational problems. Several groups have examined the feasibility of solving other combinatorial NP-complete problems with DNA computing.[2−5] Notably Smith's group has developed a technique to implement DNA computing on a solid surface upon which a set of DNA molecules encoding all candidate solutions to a given problem are immobilized and processed cyclically.[5−8] The advantage of this approach is that DNA computation can be performed in a faster and more error-resistant fashion than those based on test tubes.

The graph coloring problem is a well-known NP-complete problem, and all known algorithms for it have exponential worst-case complexity. In 1996 Amos et al. proposed a DNA algorithm to solve the three-coloring problem based on a test tube.[9] First, an initial test tube T is generated in which all candidate DNA strands in the form of $v_1c_1v_2c_2\cdots v_nc_n$, representing the whole possible colorings to a graph G (v_i and c_i respectively denote the ith vertex and its color while n is the total number of vertices in the graph) are present. In each loop $i(1 \leq i \leq n)$ of the algorithm, test tube T is copied into three test tubes T_1, T_2, and T_3 which contain the same DNA strands as does T. Then for each vertex v_i, only strands that both assign color j to it and vertices connected with its other two colors remain in the test tube $T_j(1 \leq j \leq 3)$. Finally the three test tubes were merged into T to proceed to another loop. As the algorithm is terminated, strands in test tube T will only encode legal colorings. In another approach, Bach et al. developed a generator that can produce DNA strands in the form of $v_1v_2\cdots v_n$ ($v_i \in (0,1)$), representing all subsets of vertices V in graph G whose size is no more than $n/3$ (if vertex v_i is 0 then it is not in the subset and otherwise it is in the subset). Then elements that are not independent subsets of vertices were discarded from the initial test tube T. For each element s remaining in T, if the subgraph $G - s(s \in T)$ is bipartite then graph G can be three colored.[10]

The algorithms described above were based on test tubes, and their computational power was greatly limited by the volume of DNA molecules which can be manipulated in the laboratory. As the complexity of a problem increases, longer DNA strands will have to be employed to encode the candidate solutions. This may lead to the undesired secondary structures such as hairpin and mishybridization during the chemical reactions. Loss of DNA strands is another factor that may ruin the whole computing process. In this paper, we introduce a new algorithm to solve the graph coloring problem based on surface.

2. BACKGROUND

2.1. The Graph Coloring Problem. The graph coloring problem can be described as follows: Given a graph G and an integer k, is there a way to color the vertices with k colors such that adjacent vertices are colored differently? Generally, there exist k^n (n is the total number of vertices in graph G) candidate solutions for this problem. The specific graph concerned in this paper is shown in Figure 1. It is a typical four critical graph, and, based on graph theory,[12] it cannot be legally colored with colors less than four.

2.2. Algorithm for the Graph Coloring Problem. Our algorithm can be described as follows:

Step 1: Assign k colors to each vertex of the graph G;

Step 2: Select the first legal color for vertex $v_i(1 \leq i \leq n - 1)$ and delete other color assignments;

Step 3: Delete those vertices that both have the same color with vertex v_i and connected with it in parallel;

Step 4: Repeat steps 2 and 3 until the last vertex $v_n$is reached; detect whether there exist legal colors for v_n. If the answer is yes, then output "YES", otherwise output "NO".

3. IMPLEMENTATION OF THE ALGORITHM

3.1. Encoding Scheme. In this paper we use four colors B (blue), G (green), R (red), and Y (yellow) to color the graph in Figure 1. Single strands in the form of 5′-C_6-T_{15}-V_{12}-CCCGGG-F_{12}-3′ are used to encode one possible coloring of each vertex in the graph. The sequence T_{15} is used to separate the hybridizing sequences from the surface, while

* Corresponding author e-mail: wbliu@mail.hust.edu.cn.
† Huazhong University of Science and Technology.
‡ Shandong University of Science and Technology.

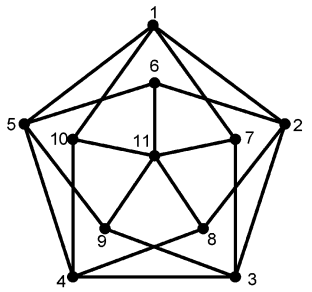


Figure 1. The graph of coloring problem.

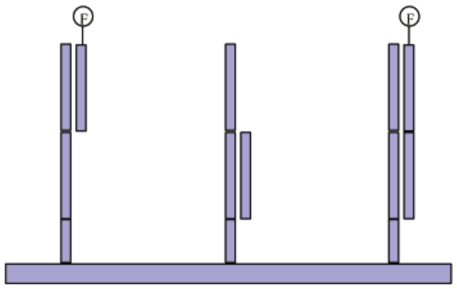


Figure 2. Example of the mark operation.

sequence V_{12}-CCC and GGG-F_{12} respectively encode vertex v_i and its colors. It must be noted that sequences encoding vertices and colors must guarantee enough difference so that they can be uniquely distinguished in the hybridization process. The middle sequence CCCGGG is recognizable by restriction endonuclease *SamI* which can split it at the middle site as shown in the following:

5’ -CCC ↕ GGG-3’

3’ -GGG ↕ CCC-5’

3.2. Surface Based Operations. In this section we introduce the logical operations used in this paper and their implementation on the surface-based DNA computer. The four basic operations used in this paper are as follows: mark, unmark, scan, and delete.

1. Mark. All strands on the surface satisfying some constraint are identified as marked. In DNA computing, the constraint usually means there exists a unique subsequence in these strands. Single strands are marked simply by making them double-stranded as all strands on the surface are single-stranded at the beginning. These subsequences being added to the surface will anneal with strands satisfying the constraint. Partial double strands will be formed according to the Watson−Crick complement rule as in Figure 2.

2. Unmark. This biological operation is opposite to the Mark operation− it denatures all double strands into single strands through heating the surface to about 94 °C. At the end of this operation accomplished, the complementary strands will be washed away by distilled water.

3. Scan. In each loop of the computing process, we need to detect if there are legal colors left for the current vertex v_i. In the mark operation, all of the complementary subsequences encoding colors are fluorescent-tagged at the 5′-end as follows:

$$3'\text{-CCC-}N_{12}\text{-FAM-}5'$$

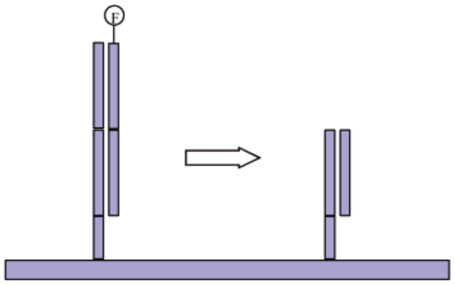
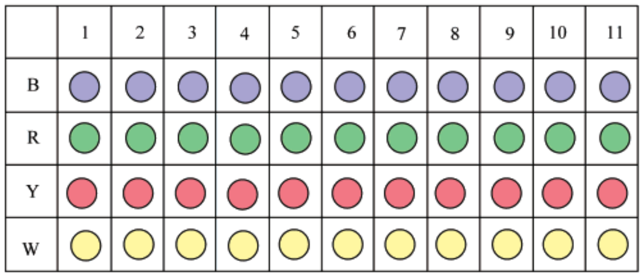


Figure 3. Illustration of the delete operation.



	1	2	3	4	5	6	7	8	9	10	11
B	🔵	🔵	🔵	🔵	🔵	🔵	🔵	🔵	🔵	🔵	🔵
R	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢	🟢
Y	🔴	🔴	🔴	🔴	🔴	🔴	🔴	🔴	🔴	🔴	🔴
W	🟡	🟡	🟡	🟡	🟡	🟡	🟡	🟡	🟡	🟡	🟡

Figure 4. Single strands immobilized on an addressed microarrays.

In which N_{12} represents the complement subsequences of F_{12}. One way to implement this operation is to detect fluorescent images formed on the surface by a Molecular Dynamics FluorImage 575.

4. Delete. All strands (single or double) satisfying some conditions are removed from the surface through the enzymatic manipulation of DNA molecules. The unmarked single strands can be removed from the end by exonuclease *E. coli*, while double strands containing the recognition site of endonuclease *SamI* can be split at the middle position as shown in Figure 3. After this operation is accomplished, the surface is washed by distilled water to remove the residues.

3.3. Implementation. To implement step 1 of the algorithm, all single strands encoding a possible coloring of vertices in Figure 1 are synthesized according to the formula described in 3.1 and are immobilized on a surface. These single strands are arranged in addressed microarrays as showed in Figure 4. Each column on the surface is assigned to one vertex, while the four rows represent its four different colorings.

To implement step 2, first all strands on the surface are marked with their complementary subsequences of the four colors. Subsequently the first legal color for vertex v_1 (or v_i) is scanned. In this paper, the first legal color assigned for the current vertex v_i is defined as the color according to the order of *B*, *G*, *R*, and *Y* during the computing process. For example the result for vertex v_1 is *B*. If there are no legal colors to be found for the current vertex v_i, then terminate the algorithm and output “NO” during the computing process.

Next the operation unmark is initiated to reset the surface. Then complementary subsequences of the first legal color and vertices $v_j(v_j \neq v_1)$ are added to the surface. After that only strands that do not assign the first legal color to vertex v_1 (or v_i) is unmarked and they are deleted by exonuclease *E. coli*. Finally only those strands coloring vertex v_1 with *B* remain on the surface.

To implement step 3, the operation unmark is initiated again to reset the surface. First we mark strands including

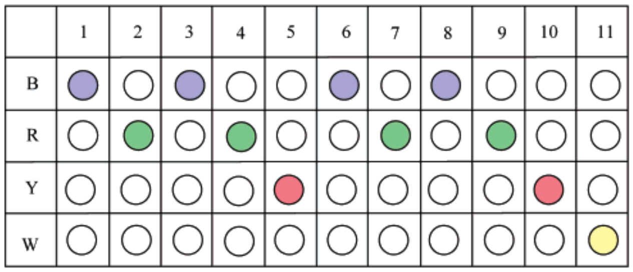


Figure 5. Result of the final coloring.

the first legal color and then mark those including vertices connected with vertex v_1(or v_i). At this point strands satisfying both conditions are marked as the third case shown in Figure 2. DNA ligase is added to the surface to seal the niche between the complementary subsequences of vertices and the first legal color (i.e. they are joined together). Since the middle sequences CCCGGG are then double-stranded, strands containing both the subsequences of the first legal color and those of the vertices connected with vertex v_1(or v_i) are cut at the middle position as shown in Figure 3.

By the end of this loop, vertex v_1 is then legally colored, and other vertices assigned with conflicting color with it are removed from the surface. The process described is repeated until the last vertex v_n is reached. The final result can be obtained by scanning the surface with the four possible colors. The fluorescent dots shown in Figure 5 represent the color assigned to each vertex. The result shows that the graph in Figure 1 can be legally colored with at least four different colors. If only three colors were used, then the last vertex 11 will have no legal color left. If only two colors were used, then the computing process will terminate at loop 5. It should be pointed out that in each loop one must keep assigning the first legal color to the current vertex v_i so that the minimal number of colors is used.

4. CONCLUSION

DNA computing is a new computational paradigm that may provide some advantages over conventional electronic computing techniques. Much work has been done to search better ways to describe DNA operations and to take advantage of the existing knowledge of mathematics and molecular biology. In this paper, we introduce a DNA algorithm for solving the graph coloring problem based on surface. The candidate solution space is represented by the combination of unique short strands immobilized on addressed microarrays on the surface. This is totally different from those algorithms proposed by others who use one single strand as a possible solution. Compared with those approaches reported, our method has the following advantages:

1. As is known, the stability of DNA strands is reverse-proportional to their lengths, and the maximal length that can be handled in the laboratory now is about 10000 bp. Hence the complexity of problems that can be addressed through earlier algorithms[9,10] will be greatly limited by this constraint, while our method can be applied to a larger problem instance reliably without increasing the length of DNA strands significantly. The length of DNA strands is only needed to increase the encoding of more vertices with enough diversity as the problem size increases.

2. The difficulty and importance of encoding in DNA computing and designing a robust set of DNA strands have been increasingly recognized. With the length of DNA strands increasing, the undesired hairpin structures and mishybridization, both of which may seriously ruin the computing process, will beome hard to deal with in the test tube scenario. Because only short DNA strands are needed in our algorithm, the computational process can be implemented in a reliable way, and the tightness for encoding is also greatly alleviated. Moreover, the total number of the unique DNA strands needed to be synthesized ($n + k$) increases linearly with the scale of a given problem. Only short DNA strands are needed in our algorithm, hence the tightness for encoding is greatly reduced.

3. The operations used in this paper are mainly mark, unmark, scan, and delete, and the biological means to implement them are relatively easy to manipulate in the laboratory. As we arrange all the single strands in the microarrays, the final result is easily detected without using other additional techniques such as PCR and sequencing which are laborious and error prone. The efficiency of our method can be seen from the time complexity of our algorithm O (n) (verses O (n^2) in others).[10,11]

ACKNOWLEDGMENT

This paper is supported by the National Scientific Foundation of China under grants 60103021 and 60144047. The authors gratefully acknowledge the anonymous referees' suggestions to this paper. Our appreciation extends as well to Dr. Longbin Liu (Amgen company) for his careful reading and comments that improved the presentation of this work.

REFERENCES AND NOTES

(1) Alderman, L. M. Molecular computations to combinatorial problems. *Science* **1994**, *266*, 1021−1024.
(2) Lipton, R. J. DNA Solution of Hard Computational Problems. *Science* **1995**, *268*, 542−545.
(3) Ouyang, Q. et al. Solution of the Maximal Clique Problem. *Science* **1997**, *278*, 446−449.
(4) Head, T. et al. Computing with DNA by Operating on Plasmids. *Biosystem* **2000**, *57*, 87−93.
(5) Smith, L. M. et al. A Surface-based Approach to DNA Computation. *J. Comput. Biol.* **1998**, *5*, 255−267.
(6) Anthony, G. et al. Demonstration of a Word Design Strategy for DNA computing on Surfaces. *Nucleic Acids Res.* **1997**, *25*, 4748−4757.
(7) Frutos, G. et al. Enzymatic Ligation Reactions of DNA "Words" on Surfaces for DNA Computing. *J. Am. Chem. Soc.* **1998**, *120*, 10277−10282.
(8) Liu, Q. et al. DNA computing on surfaces. *Nature* **2000**, *403*, 175−179.
(9) Amos, M. et al. Error-resistant implementation of DNA computations. Proceedings of the Second Annual Meeting on DNA Based Computers; 1996; Vol. 44, pp 151−161.
(10) Bach, E. et al. DNA Models and Algorithms for NP-complete Problems. Proceedings of the 11th Annual IEEE Conference on Computational Complexity; 1996; pp 154−168.
(11) Bondy, J. A.; Murty, U. S. R. *Graph Theory with Applications*; The Macmillan Press Ltd.: 1976.

CI025546E