Efficient DNA Computing Decoding Method

Zhiquan Frank Qiu and Mi Lu
Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843-3128, U.S.A.
{zhiquan, mlu}@ee.tamu.edu

Abstract

After Adleman [1] solved the Hamilton Path Problem using a combinatorial molecular method, many other hard computational problems have been investigated with the proposed DNA computer [8] [9] [10] [11] [13] [14]. Most of these algorithms can only provide the “yes” or “no” answers. At the time exact answers need to be generated, the strands in the wanted pool may be isolated and decoded by using electron microscopy one strand at a time [15]. However, this method to decode the strands is very inefficiency. During the algorithms implementation, usually a lot of duplicate strands are generated to reduce the error rate [2]. That makes the decoding process very difficult because the same answer may be decoded many times when only one of the duplicated strands needs to be decoded.

In this paper, a new DNA computing model is introduced based on which new algorithms are developed to solve the 3-Coloring problem. These new algorithms are presented as vehicles demonstrating the advantages of the new model, and they can be expanded to solve other NP-complete problems. They have the advantage of decoding all the strands in the final pool very quickly and efficiently. The advantages provided by the new model make DNA computing very efficient and attractive in solving computational intense problems.

Keywords

DNA Computing, Molecular Computing, Efficient Decoding, Coloring Problem, Parallel Processing, Speedup

I. Introduction

Since Adleman [1] solved the 7-vertex instance of the Hamiltonian Path Problem (HPP), there has been some ideas on how DNA can be used for computations [8] [9] [10] [11] [13] [14]. As one liter of water can hold 10^{22} bases of DNA, these methods all take advantage of the massive number of processors available where each strand is counted as one processor. This has raised the hope to solve the problems intractable for electronic computers. The major goal of these subsequent research in this area is to understand how DNA computing can be used to solve NP-complete problems. All these algorithms solve the problem by going through an exhaust search and then generate a final set of strands. If the final set is empty, then there is no solution for the problem. Otherwise, there is at least one answer for the problem. Usually the answer is “yes” or “no” based on the final set, such as the answer for the 3-Coloring problem [2] [3]. The answer is “no” when the final set is empty and “yes” otherwise. If an exact answer is needed, one of the strands in the final set may be decoded by using electron microscopy [15]. For example, when a solution for 3-Coloring problem is needed, “yes” or “no” answer is not good enough. A strand in the final set needs to be decoded when the set is not empty.

If all the answers for the problem need to be found, then all the strands in the final set need to be decoded. During the process toward the final set generation, many strands may be duplicated many times by the PCR (Polymerase Chain Reaction) in order to reduce the error rate [2]. At the end of the algorithm implementation, one solution may be represented by many strands that are exactly the same. These duplicated strands may be decoded one at a time if the electron microscope is used for decoding. This process is very inefficient because the strands that represent the same answer are decoded many times when they need to be decoded only once. In this paper, a new DNA computing model is introduced which can solve this problem. Not only can the algorithm designed based on this model find all the solutions, it can also accomplish the task very cost efficiently.

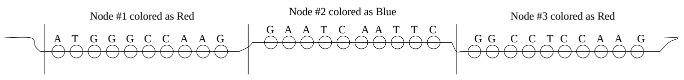


Fig. 1: AN EXAMPLE OF THREE NODES IN A GRAPH THAT ARE COLORED BY 3 COLORS: R(RED), B(BLUE) AND R(RED)

A. Our New Model

Our new model only adopts the matured DNA biological operations [1] [13]. The following basic principle operations: *synthesis*, *ligation*, *separation*, *combination* and *detection* are selected for building the new model.

synthesis $I(P, \pi)$ To generate a pool of coded strands, P, following criteria π. Strands are coded differently for different applications using the four base nucleotides: A, G, T and C. A set is defined as a group of strands, and the container holding a set of strands is called a pool. If the criteria is the colors of a node in a graph, then a pool of strands coding all the possible colors for the node is expected after *synthesis*. In the graph coloring problem, the strand is encoded for the colors of a number of nodes. Here, a few consecutive nucleotides on the strand coded for the color of one node form a region. For example, in Figure 1, one strand consists of three regions such that $s = \{RBR\}$ where $(CCAAG)$, $(AATTC)$ and $(CCAAG)$ represent the colors for three nodes as $R(Red), B(Blue)$ and $R(Red)$, respectively.

ligation $L(P_3, P_1, P_2)$ To bind strands in pool P_1 with strands in pool P_2. Each code s_{1_i} in P_1 is *ligated* to every other code s_{2_j} in P_2. If the strands in P_1 represent the codes $\{s_{1_i} | i = 1, 2, \cdots, c, where\ s_{1_i} \in P_1\}$ and those in P_2 represent the codes $\{s_{2_j} | j = 1, 2, \cdots, d, where\ s_{2_j} \in P_2\}$, after the *ligation*, the *ligated* strands are stored in P_3 and they represent the codes $\{s_k | k = 1, 2, \cdots, c \times d\}$, where $s_k = s_{1_i} s_{2_j}$ for $k = i + (j-1) \times c$.

separation $S(P, P_t, P_f, \theta)$ To partition strands in pool P, and store these strands in two new pools: P_t and P_f based on criteria θ. After each *separation* operation, the strands that meet the criteria will be stored in one pool, P_t, while all the strands that do not meet the criteria will be stored in the other pool, P_f. In order to perform the *separation* operation, many identical short strands defined as probes are attached to magnetic beads. These probes are then put in the pool containing the strands to be *separated.* Each probe can be paired up with a complementary strand to form a double helix. Such pair-up only occurs under the WC(Watson-Crick) complement rule: A only pairs with T and G only pairs with C. For example, in Figure 1, if the strands containing the region for node 1 colored as 'R' need to be *separated*, the DNA short strands $TACCCGGTTC$ should be used as probes because $TACCCGGTTC$ complements to $ATGGGCCAAG$. Also, the double helix can be separated by heating in order to make the paired strands apart from each other without breaking the chemical bonds that hold the nucleotides together inside the single strand. The strands in the pool containing a region that complements to the probes will be hybridized to, and captured by the probes, while all those without the region will remain in the pool [15].

A gel-based *separation* technique for DNA computing [4] has been developed which uses gel-layer probes instead of the bead to capture the strands. The capture layer only retains the strand with a region complementing to the probe when it is cooled down, and will let all strands pass when the layer is heated up. The advantage of using gel-based probes over bead-based probes is that the gel-based method is more accurate when capturing DNA molecules. In Figure 2 which illustrates the gel-based *separation*, a set of strands run from the left side buffer to the right. At each capture layer, the temperature is cold in order to capture the desired strands, and all unwanted strands are passed through into one pool. Then the temperature is raised to let all desired strands in the layer pass into another pool. The strands from the left buffer are *separated* and stored in two different pools.

combination $B(P, P_1, P_2)$ To pour two pools, P_1 and P_2, together to form a new one, P.

detection $D(P)$ To check if there is any strand left in the pool, P.

The rest of the paper are organized as follows: the next section will give an example problem: 3-Coloring problem and the new DNA computing algorithm, based on the new model we proposed, to solve the 3-Coloring problem. The introduction of the new algorithm with efficient decoding process that can generate exact solutions for the 3-Coloring problem is shown in Section III. The last section will conclude this paper.

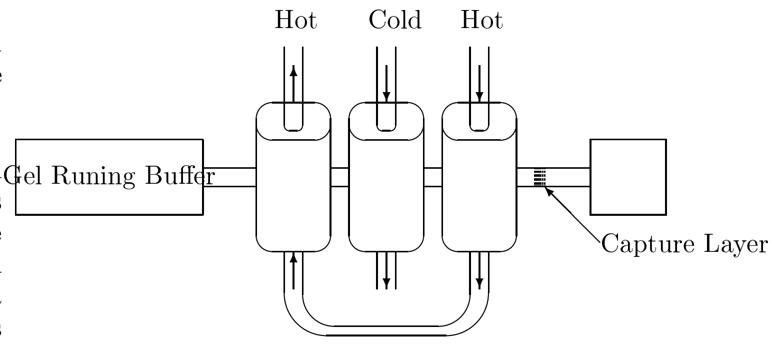


Fig. 2: SEPERATION OPERATION BASED ON GEL LAYERS

II. The Fundamental New Algorithm

Our new algorithm for the 3-Coloring problem is developed based on our new DNA computing model. The basic algorithm which will generate the answer for the 3-Coloring problem of a given graph is introduced in this section. In the next section, the algorithm will be advanced to show how the answers can be decoded efficiently.

A. 3-Coloring Problem

The 3-Coloring problem, a special case of the k-Coloring problem where k=3, is a well known representative of the class of NP-complete problems. A new algorithm for solving the 3-Coloring problem will be introduced, and will simplify the explanation of our new DNA computing model. The algorithms developed hereby can be expanded to solve the k-Coloring problem and be generalized to solve other NP-complete problems.

k-Coloring Problem: A k-Coloring problem is to color an undirected graph $G = (V, E)$ in such a way that no two adjacent vertices are sharing the same color [6]. Two nodes connected by an edge are referred to as adjacent vertices. The solution is a function $c : V \rightarrow 1, 2, \cdots, k$ such that $c(u) \neq c(v)$ for every edge $(u, v) \in E$. In other word, the numbers 1, 2, $\cdots$, k represent the k colors, and the adjacent vertices must have different colors. The k-Coloring problem is to determine whether k colors are adequate to color a given graph [7].

A simple example graph with ten nodes and ten edges, G(10,10), is given in Figure 3. It is clearly shown that the graph can be colored if $k \geq 3$.

In order to solve 3-Coloring problem, we need to generate a pool of encoded DNA strands representing all the possible color patterns of the n-node graph where each color pattern

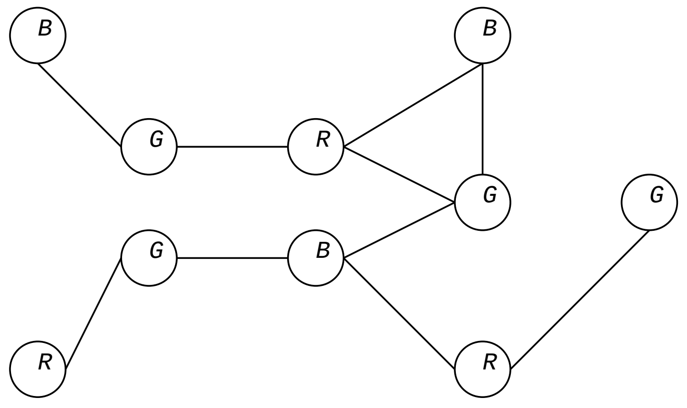


Fig. 3: AN EXAMPLE GRAPH G{10,10} THAT CAN BE COLORED AS R(RED), G(GREEN) AND B(BLUE)

is an assignment of colors to nodes [2] [3]. For example, for nodes $n_1n_2n_3n_4$, "$BBRG$" is one pattern which assigns Blue to n_1, Blue to n_2, Red to n_3 and Green to n_4, while "$RGBB$" is another pattern which colors $n_1n_2n_3n_4$ as Red, Green, Blue and Blue, respectively. After the strands are generated and stored in a pool, the strands representing the color patterns with no color conflict need to be *separated.* Two nodes along an edge are defined as having color conflict when they are sharing the same color. For the color patterns with some color conflicts existing along some edges of the graph, the correspond strand should be filtered out from the pool.

Our basic algorithm for 3-Coloring problem is introduced next. Following that, the efficient decoding algorithm and the advantages of the new algorithms will be described.

B. The New Algorithm

Given a graph $G = (V, E)$, with $V = \{v_i | i = 1, 2, \cdots, n\}$ being a set of nodes and $E = \{e_j | j = 1, 2, \cdots, m\}$ being a set of edges. Our approach to solve the 3-Coloring problem for such graph is divide-and-merge. Partition graph G into two subgraphs: $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \phi$ and $\mid V_1 \mid \approx \mid V_2 \mid$ by eliminating all edges (u, v) such that $u \in V_1$ and $v \in V_2$. Refer this set of edges as the cut-set of G, C [6] [5]. The partition process can be performed recursively, that is, subgraph G_i can be partitioned into G_{2i+1} and G_{2i+2}, until each subgraph contains only one vertex and n subgraphs exist in total (See Figure 4).

After partitioning the graph G into n subgraph, the algorithm starts to merge every two subgraphs recursively and in parallel. Before the merge, every subgraph is colored with 3 colors. During the merge, the color patterns of the two subgraphs are combined together. The merge operation continues until graph G is re-established. Note, to merge two subgraphs, the edges in the cut-sets eliminated earlier to partition the subgraphs will be added back and each addition of such edge will introduce a color conflict if the nodes it links are of the same color. Hence, the color patterns that worked for the subgraphs may not work for

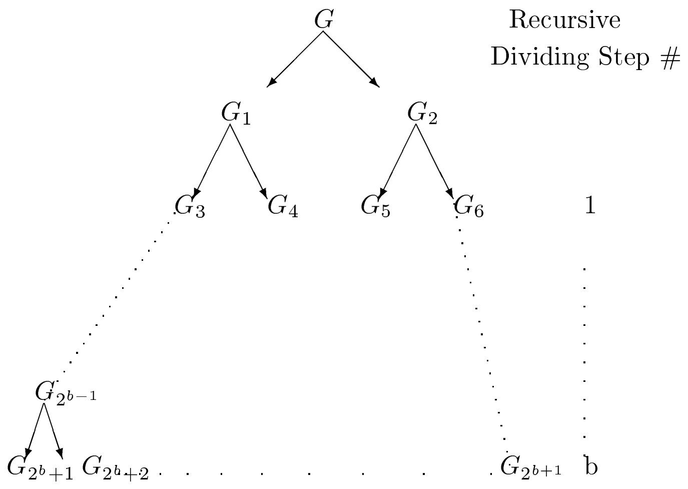


Fig. 4: DIVIDE THE GRAPH, G_1, WITH $n = 2^b$ NODES UNTIL EACH SUBGRAPH ONLY CONTAINS ONE NODE

```
Algorithm 1.
for $i$=1 to $n$ do
    In Parallel( $I(P_i$, color of node $i)$)
end
$f = n$
while $f \neq 1$ do
    In Parallel(Make multiple copies of strands in all pools)
    for All odd $i$ do
        In Parallel( $L(P_i, P_i, P_{i+1})$ )
        In Parallel( relabel all pools 1 to $\frac{f}{2}$)
    end
    $f = \frac{f}{2}$
end
$S(P, P_t, P_{f_1}, \theta)$, $\theta$ is color conflicts along $e_1$
$K$=1;
for $i$ =2 to $m$ do
    $S(P_t, P_{1_t}, P_{1_f}, \theta)$, $\theta_i$ is color conflicts along $e_i$
    for $j$ = 1 to $K$ do
        In Parallel { $S(P_{f_j}, P_{j_t}, P_{j_f}, \theta_i)$ }, $\theta_i$ is color conflicts along $e_i$
    end
    for $j$=1 to $K$ do
        In Parallel( $B(P_{f_j}, P_{j-1_f}, P_{j_t})$)
    end
    B$(P_t, P_{1_t}, \phi)$
    B$(P_{f_{K+1}}, P_{K_f}, \phi)$
    $K = K + 1$
end
Check if $P_t$ is empty to return "yes" or "no" accordingly.
```

Fig. 5: THE NEW DNA COMPUTING ALGORITHM TO SOLVE THE 3-COLORING PROBLEM FOR SPARSE GRAPHS

the merged graph after they are combined, and some combined color patterns should be eliminated. The elimination continues until the color patterns legitimate for the graph are found.

Our new algorithm for solving the 3-Coloring problem on a sparce graph is presented in Figure 5. The first *for* loop is used to generate n pools of strands to represent all possible color patterns for n subgraphs while initially each subgraph only contains one node.

The function of the *while* loop is to, first, merge the pairs of two subgraphs. The bio-operation needed to merge the two subgraphs is *ligation* which *ligates* strands in two pools to form longer strands. Let the color patterns for subgraph G_1 be s_i and those for G_2 be s_j. For a given s_i, all the s_j's should be *ligated* with it, and such operations are performed over all the s_i's. That is, the strand for one color pattern of a subgraph is replicated and each duplicated copy is *ligated* with one of those strands representing the color patterns of the other subgraph. After the merge, all the color patterns of the merged graph will be represented by the *ligated* long strands.

Inside the *while* loop, multiple copies of all the strands in all the pools need to be prepared for the next round of *ligation*. This duplication can be accomplished by using the PCR (Polymerase Chain Reaction) process [2] [12].

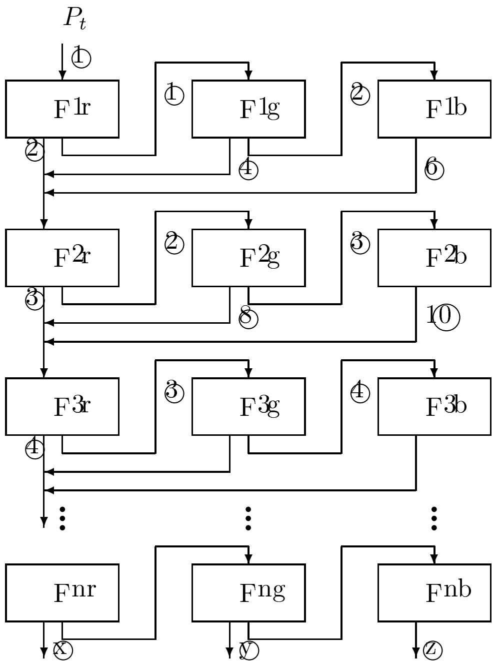
After the merge, some *ligated* strands may encode the color patterns that have color conflicts introduced by those edges in all cut-sets eliminated in the partition step. Our task is to investigate every edge in the cut-sets and detect all the color conflicts caused hereby. This is accomplished by the *separation* operation, i.e., in all the *ligated* strands, to filter out strands that contain any color conflict from the pool. For each edge under investigation, two nodes, i and j, are connected. We first *separate* the pool into three pools that contain the strands having node i colored as R, G and B. In these three pools, the strands having node j colored as R, G and B are filtered out respectively, by using the *separation* operation.

If there is any strand left in the final pool, P_t, then the 3-Coloring problem has an answer, "yes". Otherwise, the graph can not be colored by only three colors and the answer is "no".

III. Locate the Exact Solutions

After the final set that contains all the solutions for the 3-Coloring problem for the graph is generated, it is time to decode the strands in order to reach the color patterns that can color the graph. Each of the strand in the pool has one answer encoded and some strands in the pool may encode the same answer.

The new method introduced here can decode all the color patterns represented by the DNA strands in pool P_t without using the electron microscope to "read" the strands one by one. It is much more cost and time efficient comparing with the method that decodes the strands in the pool one at a time using electron microscope. The flow diagram of the new method is illustrated in Figure 6. The function of each box in this figure is a filter based on the gel-based



Circled numbers represent places where valves are placed

Fig. 6: AUTOMATED "*DECODING*" PROCESS WITH 3n FILTER

separation. The detailed structure of each box is shown in Figure 7. The filter function is given below: before the input buffer is filled, the capture layer is filled with small segments of DNA strands. Each filter is named as Fkc, where $k \in \{1, 2, \cdots, n\}$ and $c \in \{R, G, B\}$, the capture layers contains the DNA strand segments that represent the color pattern complements to color c for node k. The temperature is cooled down first. Then, the input buffer lets the input pool flow into the capture layer and valve A is opened. All strands that contains the segment representing color c for node k are capture in the layer. The rest of the strands in the input pool will pass the layer and go through valve A. When this process is finished, valve B is opened and the temperature of the container is increased. All strands containing the segment that represents node k being colored with color c is *separated* from the rest of the pool. The order of the operations are indicated in Figure 6. For example, $F1r$ will divide the input pool into two pools where they contain strands representing color patterns $N_n N_{n-1} \cdots N_2 N_1 = \{XX \cdots XR\}$ and $N_n N_{n-1} \cdots N_2 N_1 = \{XX \cdots X\bar{R}\}$ where $X \in \{R, G, B\}$, $\bar{R} = \{G, B\}$. If two filters $F1R$, and $F2R$ are connected serially, it is easy to *separate* out the strands

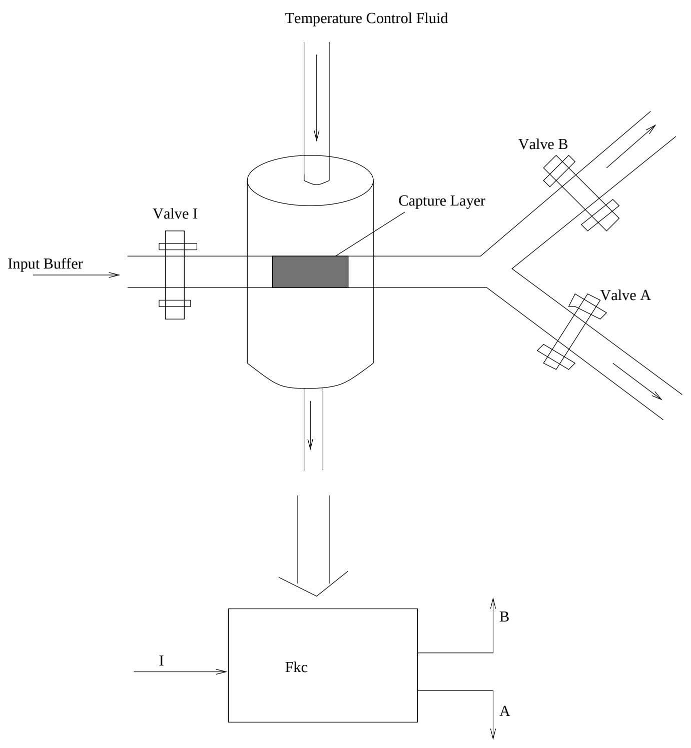


Fig. 7: AN EXAMPLE OF A FILTER FOR THE *KTH NODE WITH COLOR C*

with patterns $N_nN_{n-1}\cdots N_2N_1$ $=\{XX\cdots RR\}$. If n filters Fir where $i \in \{1, 2, \cdots, n\}$ are connected serially as shown at the left column in Figure 6, at the time strands come out from Fnr, these strands, if any, must be representing patterns $N_nN_{n-1}\cdots N_2N_1$ $=\{RR\cdots XR\}$. If no strand comes out from this filter, then the pattern $N_nN_{n-1}\cdots N_2N_1$ $=\{RR\cdots XR\}$ is not a color pattern that can color the given graph. At the time t_i, all valves labeled i in Figure 6 are opened and the temperature of the corresponding container should have been cooled down or warmed up. Eventually, x, y and z will provide some output sets. At time t_{n+1}, x should output a set. This set only has strands representing color combination of $N_nN_{n-1}\cdots N_1 = RR\cdots R$ for n nodes. The following time t_{n+2}, t_{n+3}, $\cdots$, t_{n+3^n-2}, other sets containing strands representing the color combinations $N_nN_{n-1}\cdots N_1$ $=$ $\{RR\cdots RG, RR\cdots RB, RR\cdots GR, RR\cdots GG, \cdots,$ $RB\cdots BB\}$ are outputed from x. The color combinations represented by strands are outputed from y and z in the following order: $N_nN_{n-1}\cdots N_1$ $=$ $\{GR\cdots RR, GR\cdots RG, GR\cdots RB, GR\cdots GR, GR\cdots GG,$ $\cdots,$ $GB\cdots BB\}$ and $N_nN_{n-1}\cdots N_1$ $=$ $\{BR\cdots RR,$ $BR\cdots RG$, $BR\cdots RB$, $BR\cdots GR$, $BR\cdots GG, \cdots,$ $BB\cdots BB\}$.

The *decoding* process has been simplified by using the *separate* and *detect* operations. At the time a set is outputed from x, y or z, the *detect* operation will check if it is empty. If not, the corresponding color combination is a good one for coloring the graph with no color conflict along any edge. Otherwise, the set is empty and the corresponding color combination can not be used to color the graph.

Now, let's check the extra space and effort that are necessary for efficiently decoding the strands in the final set. At the beginning, it seems that $3n$ different filters are needed. When the algorithm for generating the final set is implemented, it can be clearly seen that all the filters are already generated in order to *separate* the initial pool containing strands representing all color combinations. The extra effort is needed to reorder these filters. After the filters are connected together, the valves and temperature of the containers can be controled by electronic microcontroller automatically. The automation will greatly reduced the involvement of human being and it will make the DNA computing more error resistant. In addition, all filters on the far right column in Figure 6 are not needed because all strands coming into these strands will pass through the filter together. There is no filter function necessary here. Storage buffers can be used to replace these filters for temporarily storage in order to simplify the system. The other additional effort that is needed is the *detect* operation. This step can be accomplished very effectively and quickly.

IV. Conclusion

In this paper, a new model for DNA computing is introduced. Based on the new model, our new algorithms for the 3-Coloring problems have been presented. The new algorithms have the advantages of decoding all answers for the problem represented by DNA strands over the methods that can locate only a few answers in the whole set. The *decoding* process of the newly introduced algorithm is very fast and efficient comparing with the existing method using electron microscopy. Instead of only providing the "yes" or "no" answer, the new model can provide exact answers for the problem. Based on the *separate* operation, the new method can decode all the strands in a set with little extra cost. These new algorithms represent a huge improvement over naive search used in the existing algorithms. This will make DNA computing more attractive to potential users who want to solve problems currently unsolvable.

References

[1] Len Adleman. Molecular computation of solutions to combinatorial problems. *Science*, pages 1021–1024, November 1994.

[2] Len Adleman. On constructing a molecular computer. Manuscript, 1995.

[3] Eric Bach and Anne Condon. DNA models and algorithms for NP-complete problems. *Journal of Computer and System Sciences*, 57:172–186, 1996.

[4] Ravinderjit S. Braich, Cliff Johnson, Paul W. K. Rothemund, Darryl Hwang, Nicholas Chelyapov, and Leonard M. Adleman. Solution of a satisfiability problem on a gel-based DNA computer. In *Sixth International Meeting on DNA Based Computers*, pages 31–42, June 2000.

[5] Nicos Christofides. *Graph Theory: An Algorithmic Approach.* Academic Press, 1975.

[6] John Clark and Derek Allan Holton. *A First Look at Graph Theory.* World Scientific, 1991.

[7] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[8] Y. Gao, M. Garzon, R. C. Murphy, J. A. Rose, R. Deaton, D. R. Franceschetti, and S. E. Stevens Jr. DNA implementattion of nondeterminism. In *3rd DIMACS Workshop on DNA Based Computers*, pages 204–211, June 1997.

[9] Gre Gloor, Lila Kari, Michelle Gaasenbeek, and Sheng Yu. Towards a DNA solution to the shortest common superstring problem. In *Fourth International Meeting on DNA Based Computers*, pages 111–116, June 1998.

[10] Vineet Gupta, Srinivasan Parthasarathy, and Mohammed J. Zaki. Arithmetic and logic operation with DNA. In *3rd DIMACS Workshop on DNA Based Computers*, pages 212–222, June 1997.

[11] Peter Kaplan, David Thaler, and Albert Libchaber. Parallel overlap assembly of paths through a directed graph. In *3rd DIMACS Workshop on DNA Based Computers*, pages 127–141, June 1997.

[12] Peter D. Kaplan, G. Cecchi, and A. Libchaber. DNA-based molecular computation: template-template interactions in pcr. In *Second Annual Meeting on DNA Based Computers*, pages 159–171, June 1996.

[13] Richard Lipton. Using DNA to solve SAT. Unpulished Draft, 1995.

[14] Z. Frank Qiu and Mi Lu. Arithmetic and logic operations for DNA computers. In *Parallel and Distributed Computing and Networks (PDCN'98)*, pages 481–486. IASTED, December 1998.

[15] Sam Roweis, Erik Winfree, Richard Burgoyne, Nickolas Chelyapov, Myron Goodman, Paul Rothemund, and Leonard Adleman. A sticker based architecture for DNA computation. In *Second Annual Meeting on DNA Based Computers*, pages 1–27, June 1996.