2

DNA Computing: Methodologies and Challenges

Deepak Sharma and Manojkumar Ramteke

Department of Chemical Engineering, Indian Institute of Technology Delhi, New Delhi, Hauz Khas, New Delhi 110016, India

2.1 Introduction to DNA Computing Methodologies

Humans are looking for new approaches to computing since the starting of civilization. Over the years, researchers have invented many systems for computation, from "counting with abacus" to "complex computing by using modern-day computers." According to Moore's observation [1], the number of transistors on a silicon chip is found to be doubling in every 18–24 months, which results in the development of faster computing devices. However, in the coming decades, producing such faster computing devices will be more challenging as the size of the transistor is already approaching to a molecular level. Moreover, engineering such silicon chips is gradually becoming more complex and less cost effective. This compelled the researchers to look for alternative computing devices and methodologies. Biomolecular computing is one such excellent alternative to traditional silicon-based computing methods.

Biomolecular computing is illustrated for the first time by Adleman in 1994 [2] to solve the Hamiltonian path problem (HPP) using deoxyribonucleic acid (DNA). In such DNA-based computing (*referred to* as DNA computing), enzymatic reactions and manipulations such as addition, amplification, and cutting of the DNA are used for performing the computing. Subsequently, DNA computing is used by several researchers [3–8] to solve a variety of combinatorial problems. These studies have exploited high parallelism of DNA reactions over sequential operations occurring in silicon-based computers to solve the computationally intractable problems. Such parallel processing in DNA computer builds the confidence for solving the problems that are presently not solvable with silicon-based computers. Moreover, DNA became an effective and efficient material for faster computation, storage, and information processing owing to the significant advancements in biomolecular techniques such as gel electrophoresis, polymerase chain reaction (PCR), affinity separation, restriction enzyme digestion, etc. [9, 10].

Despite initial success, the increase in problem size leads to a significant bottleneck in scaling the existing DNA computing procedures for large size problem

DNA- and RNA-Based Computing Systems, First Edition. Edited by Evgeny Katz.

formulations. This is primarily because the amount of DNA required increases exponentially with size even though the number of biochemical steps required increases with a polynomial function. Further, there is a significant compounding of experimental errors involved, which makes these procedures redundant for solving the bigger size formulations. Also, the real-life problems often have continuous search spaces and multiple optimal solutions, whereas the existing DNA computing procedures are mostly developed for discrete search spaces involving a single optimal solution.

2.2 Key Developments in DNA Computing

In 1994, Adleman [2] used the DNA for computation in the first-ever molecular experiment performed to solve the HPP. Molecular biology experiments were performed to address the instance graph having seven vertices and fourteen edges. A year later, Lipton [3] presented a new model for solving another NP-complete (nondeterministic polynomial time complete) problem known as satisfiability (SAT) problem using a DNA computer. SAT problem is also solved by Smith et al. [4] and Liu et al. [6] with new surface-based DNA computing models.

Along with these models, other researchers [7, 11, 12] also solved NP-complete problems with a different approach. Sakamoto et al. [5] used DNA hairpin formation to solve the SAT problem. Chao et al. [13] developed a single-molecule "DNA navigator" to solve a maze. These models of DNA computing are discussed in the next section.

2.2.1 Adleman Model

Adleman's DNA computer [2] solved a small instance of a hard computational problem, the HPP. For a given graph, this problem asks for a path with a fixed start and end vertices that visits every vertex exactly once (Figure 2.1a). In his representation, each vertex is encoded using 20 bp nucleotides. Edges are encoded as 20-mers, with the first 10 nucleotides complementary to the last 10 nucleotides of the start vertex and the last 10 complementary to the first 10 of the end vertex (Figure 2.1b). By mixing and ligating oligonucleotides corresponding to vertices and edges, concatenates are formed that represent paths through the network (Figure 2.1c). A path through the graph involves all vertices such that each vertex represented only once is a correct solution to the problem and is referred to as the Hamiltonian path. However, the random nature of ligations leads to the formation of other incorrect paths that do not meet the required condition of the Hamiltonian path. Therefore, several biochemical steps are required (Figure 2.1d,e) to extract the correct path from the set of all paths generated in the ligation process. First, such biochemical step is the PCR, which amplifies only those paths that start and end with right vertices. In the second step, only the paths of correct length are extracted using gel electrophoresis. Finally, the presence of every vertex sequence is confirmed using affinity separation. If any DNA remains after the last separation, this must correspond to a Hamiltonian path.

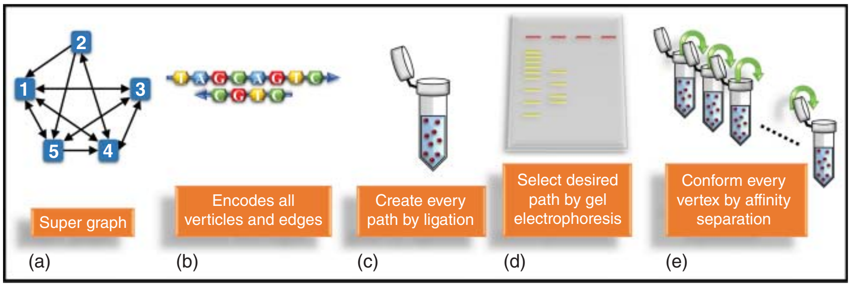
Super graph
Encodes all verticles and edges
Create every path by ligation
Select desired path by gel electrophoresis
Conform every vertex by affinity separation
(a)
(b)
(c)
(d)
(e)

Figure 2.1 Adleman's DNA computing procedure [2]. For (a) given super graph, vertices and edges are encoded using the (b) encoding strategy, and (c) mixed to generate all possible paths through the graph using ligation. The correct length path is selected using (d) gel electrophoresis. All correct length paths are further screened to confirm the presence of sequence corresponding to each vertex one by one starting from first to the last vertex using (e) affinity chromatography.

Consider a supergraph shown in Figure 2.1a involving five vertices. The objective is to obtain a path that starts from vertex 1 and ends at vertex 5 such that each vertex is represented only once. Initially, the DNA sequences of 20 bp are designed for each vertex and edge such that these should lead to linear DNA. For starting and ending edge, the complementary sequence of all 20 bp of starting and ending vertex is used instead of just 10 complementary base pairs. All encoded single-stranded DNA (ssDNA) for each vertex and edge are mixed together to form double-stranded DNA (dsDNA) representing all possible paths in the given supergraph in the ligation step. Since each vertex must be visited exactly once in the Hamiltonian path, a graph of N (= 5) vertices having each vertex encoded with L (= 20 bp) nucleotides must comprise total $N \times L$ (= 100 bp) nucleotides. Therefore, all dsDNA molecules of different lengths obtained after the ligation step are separated using gel electrophoresis. In this step, the gel slice corresponding to the band of the desired length (= 100 bp) is then separated from the gel by a cutter as the correct DNA sequence corresponding to an optimal solution is expected to be present only in this slice. Further, this band may comprise some undesired solutions with the correct length of 100 bp. The DNA solutions are extracted from the gel slice and are amplified using PCR to generate enough number of desired sequences of DNA representing the solution to the given HPP beginning with one and ending with five. Subsequently, the DNA solution undergoes affinity separation process using streptavidin–biotin magnetic beads to check the presence of vertices 1–5 sequentially one after another in tubes 1–5. The PCR products of these tubes are then analyzed by gel electrophoresis where the bands of respective lengths are obtained, signifying the location of these vertices in the entire sequence. If these tubes give the bands of 20, 40, 60, 80, and 100 bp, then it confirms that all N (= 5) vertex are visited, and depending on the location of the primer, the location of the vertex in the entire sequence is also determined. For a given example, the Hamiltonian path is 1–3–4–2–5, which corresponds to the bands of 20, 40, 60, 80, and 100 bp, respectively, on the gel image.

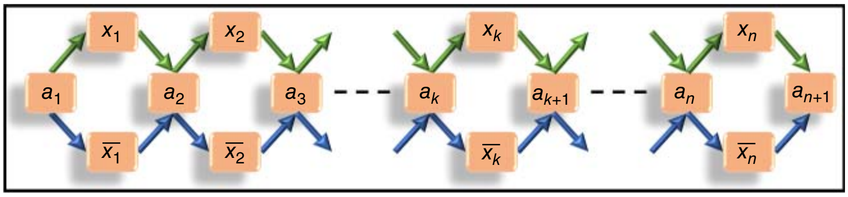

Figure 2.2 Lipton's graph [3] for constructing a binary number for a general variable string ($x_1\ x_2\ x_3 \ldots x_n$).

2.2.2 Lipton's Model

Lipton [3] solved the SAT problem with a linear complexity of biomolecular operations. In SAT problem the operators logical OR (∨), logical AND (∧), and logical NOT (¬) along with the variables (x_1, x_2, x_3, …, x_n) constitute a Boolean formula, e.g. (x_1∨x_2) ∧ (¬x_2∨x_3). In this formula, a literal can be either a variable or the negation of the variable, e.g. x_1 or ¬ x_1. The clause (C_i) is a disjunction of literals, e.g. (x_1∨x_2). Finally, a Boolean formula is a conjunction of clauses [e.g. (x_1∨x_2) ∧ (¬x_2∨x_3)] such as $C_1 \wedge C_2 \wedge C_3 \wedge \ldots.\ C_m$ for m clauses. In a SAT problem, the objective is to confirm whether there exists a set of input conditions (i.e. the input variables, $x_1, x_2, x_3, \ldots, x_n \in [0, 1]$) for which a given Boolean formula is satisfiable, i.e. the output is 1.

To solve the SAT problem, first, the variables are arranged in a string as $x_1\ x_2\ x_3 \ldots x_n$. In Lipton's model, this variable string, $x_1\ x_2\ x_3 \ldots x_n$, is represented in a graph form. Since the variables can have 0 or 1 values, the vertices with no bars (e.g. x_1, x_2, etc.) represent the 1 value, whereas those with bars (e.g. $\overline{x}_1$, $\overline{x}_2$, etc.) represent 0 value. Further to aid the graphical representation, vertices $\text{a}_1 - \text{a}_{n+1}$ are added in the sequence as a_1 (x_1 or $\overline{x}_1$) a_2 (x_2 or $\overline{x}_2$) a_3 (x_3 or $\overline{x}_3$) … a_n (x_n or $\overline{x}_n$) a_{n+1} as shown in Figure 2.2a. It is to be noted that $\text{a}_1 - \text{a}_{n+1}$ are the additional vertices included in the graph to commonly connect x_1 and $\overline{x}_1 - x_n$ and $\overline{x}_n$, respectively, and facilitate in representing all possible combinations for the string $x_1\ x_2\ x_3 \ldots x_n$. The graphical form a_1 (x_1 or $\overline{x}_1$) a_2 (x_2 or $\overline{x}_2$) a_3 (x_3 or $\overline{x}_3$) … a_n (x_n or $\overline{x}_n$) a_{n+1} is represented in the DNA world by using the sequences for each vertex and edge in the same manner as that explained in Adleman's model. These DNA solutions can be amplified and separated easily based on the specific sequence represented for each variable using the biochemical steps of PCR, gel electrophoresis, and affinity separation described earlier.

The objective is to extract a binary sequence for $x_1\ x_2\ x_3 \ldots x_n$ from all possible combinations that satisfy all the clauses $C_1, C_2, C_3, \ldots, C_m$. To solve this problem, the ssDNA sequences of $x_1, \overline{x}_1, x_2, \overline{x}_2, x_3, \overline{x}_3, \ldots, x_n$ and $\overline{x}_n$ along with sequences of $\text{a}_1 - \text{a}_{n+1}$ are added in the first test tube. Here all the sequences are selected such that the ligation represents the path between the vertices like Adleman's model. All feasible paths are represented in the first test tube. These paths are constituted by the edges from a_k to both x_k and $\overline{x}_k$ and from both x_k and $\overline{x}_k$ to a_{k+1} for any kth variable. If the vertex takes the x_k label, it encodes the value "1," and if it takes the $\overline{x}_k$ label, it encodes the value "0." For example, the path $\text{a}_1\overline{x}_1\text{a}_2x_2\text{a}_3x_3\text{a}_4$ encodes binary number 011. The next operation is the extraction in which the solution that satisfies the Boolean formula has to be extracted from all feasible solutions present in the first test tube. For this, the DNA solutions are operated by an extraction operator E (t, i, v) in several sequential steps in which t represents the sequences of the tube where an ith bit of variable string $x_1\ x_2\ x_3\ \ldots\ x_n$ is equal to v {0, 1}. Here the extraction operator E (t, i, v) is selected sequentially such that clauses C_1, C_2, C_3, …., C_m are satisfied one after another. Thus, the first extraction operation is selected to make C_1 satisfiable. All DNA solutions satisfying C_1 are then subjected to the next extraction operation, which makes C_2 satisfiable and so on. If any solution is left in the tube after sequentially satisfying C_1, C_2, C_3, …., C_m clauses, then it implies that the given Boolean formula is satisfiable for the sequence remained in the tube. If no solution is left, then it implies that the given Boolean formula is not satisfiable.

Consider a simple example of $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$ for illustration. In this formula, variables x_1, x_2, and x_3 are present. The objective is to find a binary string for $x_1\ x_2\ x_3$, which satisfies the clauses $C_1 = (x_1 \vee x_2)$ and $C_2 = (\neg x_2 \vee x_3)$. Boolean formula $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$ will be solved by making the test tubes for all possibilities, as given in Table 2.1. There are three variables, x_1, x_2, and x_3, represented by "0" or "1," which leads to total eight possibilities ($2^3 = 8$). These eight possible ways are encoded in the form of DNA molecules just by pouring all individual sequences of vertices and edges into the test tube t_0 and performing the ligation. Next, the extraction operation is performed on this tube t_0. The first extraction operator is E (t_0, 1, 1), which extract values having the first bit as "1" since this makes first clause $C_1 = (x_1 \vee x_2)$ true. Also, the second bit corresponding to x_2 can make $C_1 = (x_1 \vee x_2)$ true. Therefore, all remaining solutions that do not satisfy the E (t_0, 1, 1) are extracted in the tube $t_1{}'$. These solutions from $t_1{}'$ are then subjected to the second condition where x_2 becomes 1. Thus, the next extraction operation, t_2, is $E(t_1', 2, 1)$. The solutions extracted by t_1 and t_2 represent the solutions that satisfy $C_1 = (x_1 \vee x_2)$. These solutions are stored in tube t_3 and subjected to the next extraction operation, which makes $C_2 = (\neg x_2 \vee x_3)$ satisfiable. In the next operation, $E(t_3, 2, 0)$ is performed to extract the solution having $\neg x_2$ equal to 1, which satisfies $C_2 = (\neg x_2 \vee x_3)$. Further, the SAT of $C_2 = (\neg x_2 \vee x_3)$ depends on x_3. Therefore, all the remaining solutions stored in t_4' are subjected to $E(t_4', 3, 1)$. This extracts the solutions having x_3 equal to 1. These extracted solutions are stored in t_5. Finally, the solutions left in t_4 and t_5 are the solutions that make the given Boolean formula satisfiable. Similarly, the sequence of extraction steps for evaluating the SAT of any Boolean formula can be designed efficiently, as illustrated in the above example.

2.2.3 Smith's Model

The surface-based DNA computing model was introduced by Smith et al. [4]. In this model, DNA molecules are attached to a solid surface, instead of DNA molecules floating in a solution. Solid-phase nucleotide synthesis is used for the immobilization of the nucleotides on the solid surface. The procedure given by Smith et al. [4] involves following six steps: (i) make, (ii) attach, (iii) mark, (iv) destroy, (v) unmark, and (vi) read out as shown in Figure 2.3.

Table 2.1 The sequence of extraction operations for a given illustrative SAT problem $[(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)]$.

Test tube	Operator	Values present
t_0		000, 001, 010, 011, 100, 101, 110, 111
t_1	$E(t_0, 1, 1)$	100, 101, 110, 111
t_1'	$t_0 - t_1$	000, 001, 010, 011
t_2	$E(t_1', 2, 1)$	010, 011
t_3	$t_1 + t_2$	010, 011, 100, 101, 110, 111
t_4	$E(t_3, 2, 0)$	100, 101
t_4'	$t_3 - t_4$	010, 011, 110, 111
t_5	$E(t_4', 3, 1)$	011, 111
t_6	$t_4 + t_5$	100, 011, 101, 111

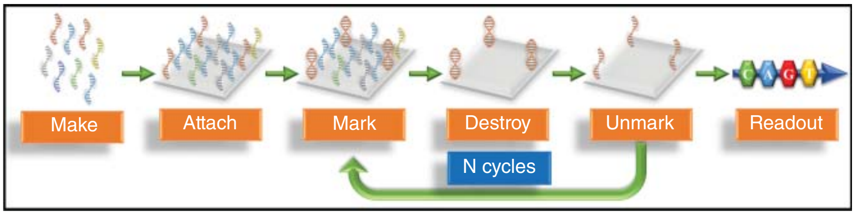

Figure 2.3 Representation of surface-based DNA computing method [4].

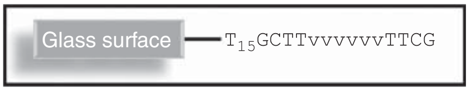

Figure 2.4 Representation of surface-bound DNA sequence.

In this procedure, first, the sequence corresponding all *possible combinations* of variables is designed in step (i). To represent each combination of variables in a SAT problem, Smith et al. [4] used the DNA sequences consisting of unique DNA sequences at each end and variable DNA sequences in the middle for hybridization such as T_{15}GCTTvvvvvvTTCG. In this, the variable DNA sequences are represented by "vvvvvv." The one end of these sequences has a spacer [15 "T" nucleotides (T_{15})] that attaches to the surface. In step (ii), the sequences generated for all combinations are attached to the solid surface, as shown in Figure 2.4.

In step (iii), the sequences corresponding to the satisfaction of each clause are marked by hybridizing these with the complementary sequences corresponding to "vvvvvv." In step (iv), all single-stranded sequences remaining after the hybridization are destroyed by treating with *Escherichia coli* Exonuclease I. In step (v), all hybridized sequences are unmarked to get the single-stranded molecules for all the remaining surface-bound sequences. Steps (iii)–(v) are repeated for all the clauses one after another. The unmarked sequences remaining at the end are analyzed in a readout operation using PCR in step (vi).

The procedure is explained in the context of the same illustrative SAT problem $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$ of three variables (x_1, x_2, x_3) described earlier. This problem has a solution space of size 8 (each variable can be either "1," or "0," total solution space = 2^3). All combinations for this problem are shown in the second row (corresponding to the test tube t_0) of Table 2.1. The problem involves two clauses: $C_1 = (x_1 \vee x_2)$ and $C_2 = (\neg x_2 \vee x_3)$. The SAT is checked for these clauses one by one. $C_1 = (x_1 \vee x_2)$ is not satisfied only if $x_1 x_2 x_3$ is represented by {000} and {001}. Therefore, the complementary sequences for all "vvvvvv" except for the above two clauses are hybridized. This eliminates the above two combinations. The remaining hybridized combinations are unmarked. Next, the SAT of $C_2 = (\neg x_2 \vee x_3)$ is checked. $C_2 = (\neg x_2 \vee x_3)$ is not satisfiable for {010} and {110}. Except these, all the sequences are allowed to hybridize. This leads to hybridization of {011}, {100}, {101}, and {111}. These are unmarked and identified in a readout step using PCR.

2.2.4 Sakamoto's Model

Sakamoto et al. [5] introduced a hairpin formation model for solving an SAT problem using molecular biology techniques. For a given illustrative SAT problem $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$, literal strings $(x_1, \neg x_2)$, (x_1, x_3), $(x_2, \neg x_2)$, and (x_2, x_3) are formed. A literal string is a string used to encode the given formula with conjunctions of the literals selected from each clause. The literal strings are obtained by concatenating of DNA sequences corresponding to each literal in a ligation step. In these literal strings, if a variable is represented in both original and negation form, then it violates the SAT condition of the given SAT problem. The literal strings without such violation in which a variable is represented only and at least in one form (either actual or negation) constitute a satisfiable solution to the given SAT problem. In Sakamoto's model, possible literal strings are first obtained by ligation. A length of the literal string equals to the number of clauses × nucleotides used for each literal. Subsequently, the obtained literal strings are subjected to temperature variation, which leads to a hairpin formation if a variable is represented in its original and negation form. The restriction enzyme destroys all such hairpins. These solutions are readily eliminated in the subsequent gel electrophoresis operation where only the literal strings with the desired length are separated. All the literal strings separated are analyzed using the sequencer, and the solution of the given SAT problem is obtained. It is to be noted that the given procedure eliminates a large number of unsatisfying literal strings, which makes it easier to deduce the correct solution from the analysis of the remaining satisfying literal strings. The procedure is useful for large size problems. However, it also has a risk of missing some literal strings due to experimental errors that may lead to an erroneous solution to the given SAT problem.

For the given illustrative SAT problem $[(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)]$, a single-stranded sequence for all literals is ligated to form the literal strings $(x_1, \neg x_2)$, (x_1, x_3), $(x_2, \neg x_2)$, and (x_2, x_3). The ligated strings are shown in Figure 2.5. These ligated strings will be subjected to restriction enzyme digestion, which eliminates the string $(x_2, \neg x_2)$, and finally, the remaining literal strings are $(x_1, \neg x_2)$, (x_1, x_3), and (x_2, x_3). From these remaining three literal strings, a solution to the given SAT

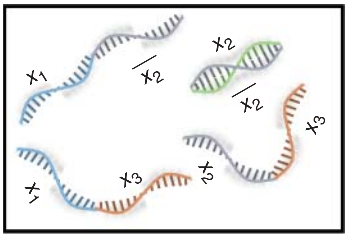

Figure 2.5 Illustration of four literal strings for the DNA hairpin formation-based computation.

problem is deduced by *mathematical analysis*. It is to be mentioned that only one literal string is eliminated for the given problem as it has only $(2)^2$ [= (number of literals in each clause)$^{\text{number of clauses}}$] equal to four literal strings. Though for the given illustrative example the search space is reduced only by 25% (as it removes only one literal string), for bigger problems the reduction in search space is significant. Sakamoto et al. [5] illustrated the benefit of the method by solving 6-variable, 10-clause problem [= $(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee x_5) \wedge (x_1 \vee x_4 \vee \neg x_6) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4)$] having three literals in each clause. This example has $3^{10} = 59\,049$ literal strings out of which only 24 literal strings were found to be satisfying the condition. Thus, ~99.95% of the search space is eliminated using the above methodology to finally obtain the correct solution just by analyzing 24 literal strings. One of the advantages of this methodology is the use of just one step, unlike sequential elimination steps used in earlier models.

2.2.5 Ouyang's Model

Ouyang et al. [11] solved a maximal clique problem using DNA computing method. A maximal clique is the largest subset in the graph in which each vertex is connected to every other vertex in the subset. In maximal clique problem, the maximum size of the clique in terms of the vertices has to be evaluated. For example (Figure 2.6a), the graph has five vertices, and eight edges where the vertices (5, 4, 2, 1) is the largest clique; thus the maximum size of the clique is four. Ouyang et al. [11] solved the maximal clique problem using DNA computing as follows.

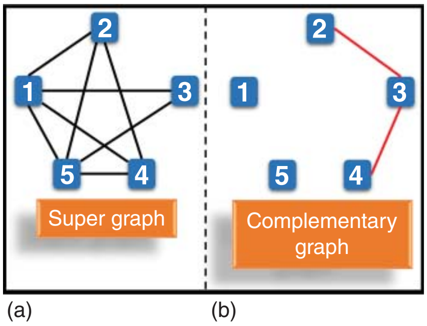

Figure 2.6 (a) The five-node graph and (b) its complementary graph used to solve the maximal clique problem.

First, all possible cliques in the graph of N vertices are represented by an N-digit binary string. In the clique, if the vertex is present, then it is represented by "1," and if the vertex is absent, then it is represented by "0." For the case of 5-vertex graph (shown in Figure 2.6a), a clique involving {5, 4, 2} vertices is represented by a binary string as {11010}. Initially, all possible combinations of this N-digit binary number are generated. Some of the vertices in the graph are not connected by the edges. A graph of such unconnected vertices is referred to as the complementary graph (see Figure 2.6b). In the next step, the combinations comprising the edges present in the complementary graph are removed. For the given illustrative example (Figure 2.6b), the combinations with {cc11c} and {c11cc} are removed from the data pool (c ε {0, 1}). Lastly, find out the binary number having the largest number of "1," which represents the size of the maximal clique. This procedure is performed using the DNA sequences as follows.

Each bit in the above binary string corresponds to a vertex and is represented as a DNA sequence having three parts. In this, the second part corresponds to the vertex number, whereas the first and the third part correspond to the position number. Further, these vertices have to be connected in sequential order (e.g. V_1–V_N) as represented in the binary string. In order to have such connection, the value "0" for the first vertex is represented as $P_1V_1{}^0P_2$, whereas the value "1" is represented by $P_1V_1{}^1P_2$. Since the next vertex V_2 has to be connected to V_1, its "0" value is represented by the complementary sequence $\overline{P_3V_2^0P_2}$, whereas its value "1" is represented by the sequence $\overline{P_3V_2^1P_2}$. Similarly, the next vertex V_3 will be connected to V_2 by $P_3V_3{}^0P_4$ and $P_3V_3{}^1P_4$ for "0" and "1" values. Thus, all odd-numbered vertices are represented by $P_iV_i{}^0P_{i+1}$ and $P_iV_i{}^1P_{i+1}$ for "0" and "1" values, whereas those for even-numbered vertices are represented by $\overline{P_{i+1}V_i^0P_i}$ and $\overline{P_{i+1}V_i^1P_i}$, respectively. Further, all sequences with V_i^0 have 10 nucleotides representing the second part, whereas these are absent in V_i^1. The ssDNA sequences corresponding to each bit combine in a sequential manner to form a dsDNA representing all combination of 0 or 1 for each vertex. Initially, all the sequences starting with P_1 and ending with P_N are amplified using PCR. The complementary graph is removed from the data pool by treating the DNA solutions using the restriction enzyme repeatedly for each edge present in the corresponding graph. For each edge, the DNA solution is divided into two equal parts in two tubes, and the restriction enzymes corresponding to the constituting vertices are added in the respective tubes. After restriction enzyme digestion, solutions of both tubes are added, and the process is repeated for all the edges of the corresponding graph. After this, a PCR is performed with starting P_1 and ending P_N. The amplified solutions are then analyzed using gel electrophoresis to obtain the maximal clique for the given supergraph. On the gel electrophoresis, the largest clique is corresponding to the shortest DNA strands. This is primarily because all the sequences comprising maximum V_i^1 will lead to the maximum "1"s in the binary string and will have the shortest length in base pairs. Such strands are analyzed by cloning and sequencing to obtain the maximal clique. For the given illustrative example (Figure 1.12b), {cc11c} and {c11cc} sequences will be removed. Therefore, the sequences that will lead to the maximal clique will be {11011}. From these, the maximal size of the clique for the given illustrative example is four, which corresponds to the clique V_1–V_2–V_4–V_5.

2.2.6 Chao's Model

Chao et al. [13] developed a single-molecule "DNA navigator" to solve a maze (tree graph) of 10 vertices with three junctions. In this, the desired path is explored out of all possible paths of the maze present on an origami that is used as a substrate. On this origami, some sites are specifically used for the binding of the vertices of the tree graph. This helps in the propagation of the path on the origami.

The DNA used for the process is very specific in size and design. The designing of DNA is described next. Hairpin DNA Y is attached to the origami and has a typical sequence layout of the structure $5' - toehold - stem - loop - \overline{stem} - 3'(5' - t_h - s_t - l - \overline{s_t} - 3')$, as shown in Figure 2.7. Another type of DNA used is hairpin DNA Z, which is present freely in the solution. Its structure is like $5' - \overline{s_t} - \overline{t_h} - s_t - \overline{l} - 3'$. Additionally, an initiator DNA that has a sequence $\overline{s_t\, t'_h}$ is used to start the process. This initiator DNA binds to the entry vertex. Similarly, an exit DNA is present, which does not have a loop to form a hairpin.

After the binding of the initiator, a hairpin loop of the DNA Y opens to make it free to bind to DNA Z and vice versa as both have complementary sites for free form of each other. This type of hybridization continues until it reaches the exit DNA. This hybridization chain also produces those paths that are not the solution to the maze. The exit DNA corresponding to an end vertex of the maze is biotin labeled. If the path is correct, then this biotin-labeled DNA is free from the exit vertex; otherwise, biotin remains attached to the DNA corresponding to the exit vertex. All the biotin-labeled sequences are then removed from the solution by streptavidin magnetic beads. The correct path sequence remained in the solution as it is not attached to the biotin. This solution is then analyzed by AFM for identifying the final path.

2.2.7 DNA Origami

Self-assembly of DNA is one of the most popular techniques used to create a variety of two-dimensional and three-dimensional structures. The DNA origami is an excellent example of self-assembly of DNA, which was introduced by Rothemund [14]. The procedure of DNA origami is shown in Figure 2.8. DNA origami is generated from a thousand base pair long ssDNA, known as a scaffold. Some shorter and linear ssDNA called staples are mixed with the scaffold to create DNA origami. For this purpose, the solution mixture is first heated to 95 °C. The solution is then cooled down to room temperature from 95 °C; throughout the cooling, the staples bind to the scaffold through Watson–Crick complementary base pairing. The scaffold then becomes a static structure or pattern of DNA. The sequence of DNA staples results in the formation of various shapes, for example, squares, smiley faces, cube, hexagon, star, and many more.

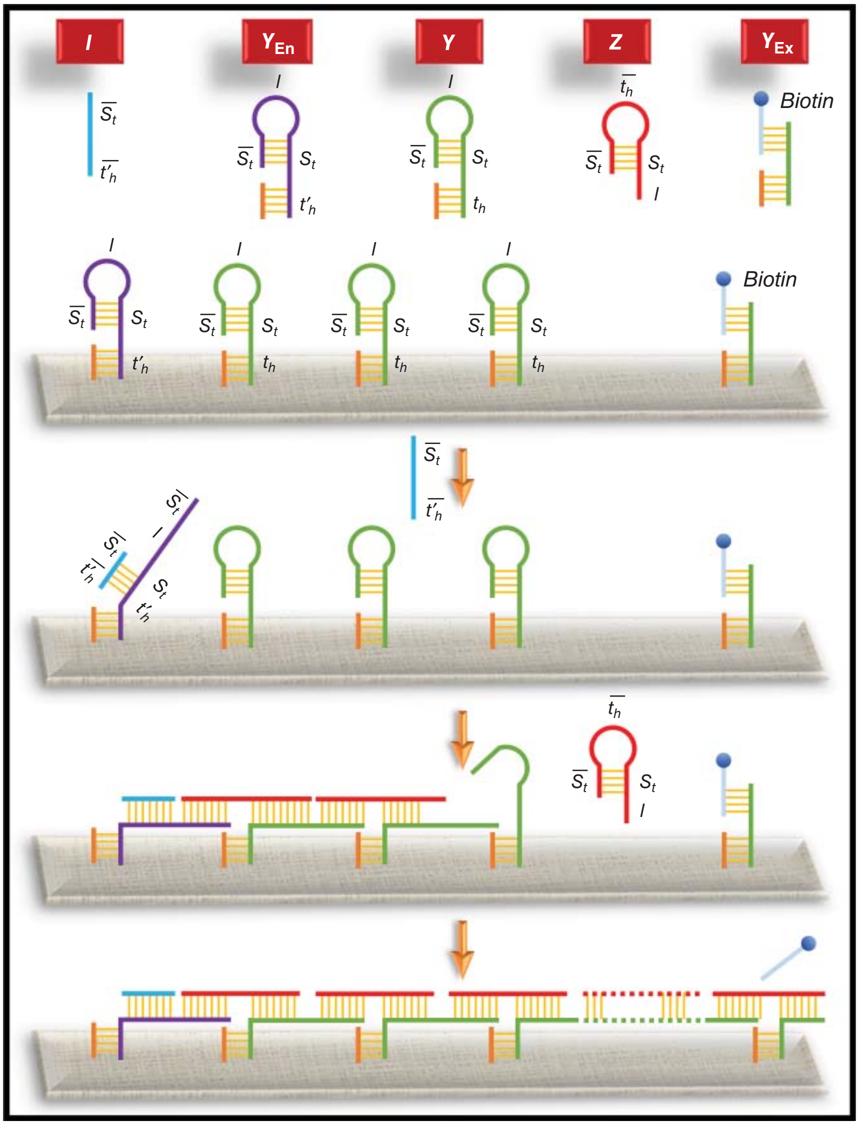

Figure 2.7 Chao's single-molecule DNA navigator [13] for solving the maze.

Based on the structure or pattern of the DNA origami, it has multiple applications in different fields of biology, chemistry, physics, computer science, and materials science. Broad application of DNA origami can be seen as in molecular robotics [15, 16], DNA walkers [17], protein function [18, 19] and structure determination [20, 21], single-molecule force spectroscopy [22, 23], nanopore construction [24–26], drug delivery [27, 28], nucleic acid analysis [29–31], enzymatic nanostructure formations [32, 33], and many more.

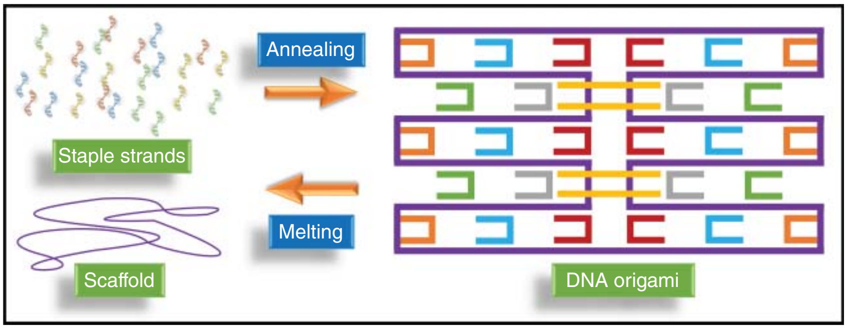

Figure 2.8 DNA origami is a group of linked dsDNA. The structure consists of one scaffold and many staple strands that are complementary to one or two domains on the scaffold.

2.2.8 DNA-Based Data Storage

DNA carries all the genetic information such as the color of eyes and hairs in the form of A, T, G, and C, which is transferred from one generation to another. Inspired by this, the researchers are now looking at the capabilities of the DNA to store the data. Today we are dealing with a large amount of data that is increasing day by day. DNA appears to be the right choice as an alternative for storing the information. In DNA-based data storage, first, the digital information present in the form of bits is encoded in DNA. This encoded information is then processed and decoded back to original binary data.

Church et al. [34] performed an experiment for storing 5.27 MB data generated from his book (contains 53 426 words, 11 images, and 1 JavaScript program) on the DNA. A simple encoding of one bit to one base is used to represent the data on the DNA. The result of this experiment clearly showed that DNA is a good material for storing digital information in addition to other storage media. Goldman et al. [35] stored over five million bits of digital information of the DNA that is later retrieved and reproduced the information with an accuracy between 99.99% and 100%.

Researchers are now looking for high data storage with high accuracy in recovery. Blawat et al. [36] stored and recovered 22 MB of a MPEG compressed movie from DNA with zero errors. Erlich and Zielinski [37] reported another encoding method that can store the 215 petabytes digital data in one gram of DNA. Recently, Organick et al. [38] demonstrated approximately 220 MB digital data storage with random access on the DNA with successful retrieval from it. In 2019, researchers from Microsoft and the University of Washington have demonstrated a fully automated system to encode and decode data in DNA [39].

2.3 Challenges

Though the use of DNA for computing can have the benefit of performing millions of operations simultaneously with very high energy efficiency, it also has several challenges. The amount of DNA required is substantial even for a simple formulation. Therefore, solving the large size problems becomes impractical owing to the requirement of a large amount of DNA. Unlike the traditional silicon-based computers in which memory reallocation is performed readily, reuse of DNA material is challenging in DNA computing as specific designs of DNA are required.

The success of DNA computing procedures is based on error-free operations of biochemical steps involved. However, the practical operations do involve experimental errors that increase with the increase in the number of steps. Further, the operations involve human interventions during the process. Therefore, solving the bigger size formulations also involves the higher probability of missing the correct answers. Further, increase in formulation size requires extracting the correct solution from the pool involving large number of incorrect solutions. Therefore, the extraction efficiency decreases with increase in formulation size. Also, the large amount of DNA representing the incorrect solutions is discarded as waste. Complete automation of all the biochemical steps is required for building a reliable DNA computer.

Another challenge for DNA computing is its application to real-world problems. Since the data of every problem has to be represented in the form of DNA, the design of DNA is specific to each problem and cannot be used for other problems. Further, for error-free biochemical operations, the DNA designs should have specific GC content with unique (noncomplementary) nucleotide sequence and should lead to a specific structure (i.e. hairpin or linear formation). These requirements reduce the designing flexibility and therefore restrict the application to bigger size formulations. Moreover, real-world problems often involve continuous search spaces with multiple optimal solutions. For such problems, the existing DNA computing procedures that are originally developed for solving the combinatorial problems involving the discrete search space need to be modified.

In conclusion, DNA computing shows great potential and has many advantages in the field of computing and data storage over conventional computing, primarily due to its ability to perform millions of calculations simultaneously using molecules. Despite this, the DNA computer is far from matching the reliability of conventional silicon-based computer owing to several challenges such as poor scaling and limited ability to handle real-world problems. The comparative analysis of existing DNA computing and data storage models illustrated their pros and cons, which is opening up new directions in materials science and biomedical applications.

Acknowledgment

This chapter is a part of the PhD thesis titled "Computing using Biomolecules" of Mr. Deepak Sharma, which is under consideration for the award of PhD degree at Indian Institute of Technology Delhi, India.

References

1 Moore, G.E. (1975) Progress in digital integrated circuits. Electron Devices Meeting, 11–13.

2 Adleman, L.M. (1994). Molecular computation of solutions to combinatorial problems. *Science* 266 (5187): 1021–1024.

3 Lipton, R.J. (1995). DNA solution for hard computational problems. *Science* 268 (5210): 542–545.

4 Smith, L.M., Corn, R.M., Condon, A.E. et al. (1998). A surface-based approach to DNA computation. *J Comput Biol* 5 (2): 255–267.

5 Sakamoto, K., Gouzu, H., Komiya, K. et al. (2000). Molecular computation by DNA hairpin formation. *Science* 288 (5469): 1223–1226.

6 Liu, Q., Wang, L., Frutos, A.G. et al. (2000). DNA computing on surfaces. *Nature* 403 (6766): 175–179.

7 Braich, R.S., Chelyapov, N., Johnson, C. et al. (2002). Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* 296 (5567): 499–502.

8 Kahan, M., Gil, B., Adar, R., and Shapiro, E. (2008). Towards molecular computers that operate in a biological environment. *Physica D* 237 (9): 1165–1172.

9 Bell, S.A., McLean, M.E., Oh, S.K. et al. (2003). Synthesis and characterization of covalently linked single-stranded DNA oligonucleotide-dendron conjugates. *Bioconjug Chem* 14 (2): 488–493.

10 Shagin, D.A., Shagina, I.A., Zaretsky, A.R. et al. (2017). A high-throughput assay for quantitative measurement of PCR errors. *Sci Rep* 7 (1): 1–11.

11 Ouyang, Q., Kaplan, P.D., Liu, S., and Libchaber, A. (1997). DNA solution of the maximal clique problem. *Science* 278 (5337): 446–449.

12 Faulhammer, D., Cukras, A.R., Lipton, R.J., and Landweber, L.F. (2000). Molecular computation: RNA solutions to chess problems. *Proc Natl Acad Sci* 97 (4): 1385–1389.

13 Chao, J., Wang, J., Wang, F. et al. (2019). Solving mazes with single-molecule DNA navigators. *Nat Mater* 18 (3): 273–279.

14 Rothemund, P.W. (2006). Folding DNA to create nanoscale shapes and patterns. *Nature* 440 (7082): 297–302.

15 Krishnan, Y. and Simmel, F.C. (2011). Nucleic acid based molecular devices. *Angew Chem Int Ed* 50 (14): 3124–3156.

16 Yurke, B., Turber, A.J., Mills, A.P. Jr. et al. (2000). A DNA-fuelled molecular machine made of DNA. *Nature* 406: 605–608.

17 He, Y. and Liu, D.R. (2010). Autonomous multistep organic synthesis in a single isothermal solution mediated by a DNA walker. *Nat Nanotechnol* 5 (11): 778–782.

18 Funke, J.J. and Dietz, H. (2016). Placing molecules with Bohr radius resolution using DNA origami. *Nat Nanotechnol* 11 (1): 47–52.

19 Hariadi, R.F., Appukutty, A.J., and Sivaramakrishnan, S. (2016). Engineering circular gliding of actin filaments along myosin-patterned DNA nanotube rings to study long-term actin-myosin behaviors. *ACS Nano* 10 (9): 8281–8288.

20 Douglas, S.M., Chou, J.J., and Shih, W.M. (2007). DNA-nanotube-induced alignment of membrane proteins for NMR structure determination. *Proc Natl Acad Sci U S A* 104 (16): 6644–6648.

21 Martin, T.G., Bharat, T.A.M., Joerger, A.C. et al. (2016). Design of a molecular support for cryo-EM structure determination. *Proc Natl Acad Sci U S A* 113 (47): E7456–E7463.

22 Shrestha, P., Jonchhe, S., Emura, T. et al. (2017). Confined space facilitates G-quadruplex formation. *Nat Nanotechnol* 12 (6): 582–588.

23 Kilchherr, F., Wachauf, C., Pelz, B. et al. (2016). Single-molecule dissection of stacking forces in DNA. *Science* 353 (6304): aaf5508.

24 Venkatesan, B.M. and Bashir, R. (2011). Nanopore sensors for nucleic acid analysis. *Nat Nanotechnol* 6 (10): 615–624.

25 Bell, N.A.W., Keyser, U.F., Puchner, E.M. et al. (2014). Nanopores formed by DNA origami: a review. *Fed Eur Biochem Soc* 588: 3564–3570.

26 Krishnan, S., Ziegler, D., Arnaut, V. et al. (2016). Molecular transport through large-diameter DNA nanopores. *Nat Commun* 7: 1–7.

27 Ouyang, X., Li, J., Liu, H. et al. (2013). Rolling circle amplification-based DNA origami nanostructrures for intracellular delivery of immunostimulatory drugs. *Small* 9 (18): 3082–3087.

28 Ora, A., Järvihaavisto, E., Zhang, H. et al. (2016). Cellular delivery of enzyme-loaded DNA origami. *Chem Commun* 52 (98): 14161–14164.

29 Endo, M. and Sugiyama, H. (2014). Single-molecule imaging of dynamic motions of biomolecules in DNA origami nanostructures using high-speed atomic force microscopy. *Acc Chem Res* 47 (6): 1645–1653.

30 Rajendran, A., Endo, M., and Sugiyama, H. (2012). Single-molecule analysis using DNA origami. *Angew Chem Int Ed* 51 (4): 874–890.

31 Wang, D., Fu, Y., Yan, J. et al. (2014). Molecular logic gates on DNA origami nanostructures for microRNA diagnostics. *Anal Chem* 86 (4): 1932–1936.

32 Linko, V., Nummelin, S., Aarnos, L. et al. (2016). DNA-based enzyme reactors and systems. *Nanomaterials* 6 (8): 139.

33 Fu, J., Liu, M., Liu, Y., and Yan, H. (2012). Spatially-interactive biomolecular networks organized by nucleic acid nanostructures. *Acc Chem Res* 45 (8): 1215–1226.

34 Church, G.M., Gao, Y., and Kosuri, S. (2012). Next-generation digital information storage in DNA. *Science* 337 (6102): 1628–1628.

35 Goldman, N., Bertone, P., Chen, S. et al. (2013). Toward practical high-capacity low-maintenance storage of digital information in synthesised DNA. *Nature* 494 (7435): 77–80.

36 Blawat, M., Gaedke, K., Hütter, I. et al. (2016). Forward error correction for DNA data storage. *Procedia Comput Sci* 80: 1011–1022.

37 Erlich, Y. and Zielinski, D. (2017). DNA fountain enables a robust and efficient storage architecture. *Science* 355 (6328): 950–954.

38 Organick, L., Ang, S.D., Chen, Y.-J. et al., and others (2018). Random access in large-scale DNA data storage. *Nat Biotechnol* 36 (3): 242.

39 Takahashi, C.N., Nguyen, B.H., Strauss, K., and Ceze, L. (2019). Demonstration of end-to-end automation of DNA data storage. *Sci Rep* 9 (1): 1–5.