13

DNA Encoding Methods in the Field of DNA Computing

Aili Han[1,2] and Daming Zhu[2]

[1] Department of Computer Science and Technology, Shandong University at Weihai, Weihai 264209, China
[2] School of Computer Science and Technology, Shandong University, Jinan 250061, China
hanal@sdu.edu.cn

Summary. Bioinformatics studies the acquisition, process, store, distribution, analysis, etc of biological information so as to understand the meanings of biological data by means of mathematics, computer science and biological techniques. Some researches on Bioinformatics, such as the properties of DNA and the Watson-Crick's law, provide a probability of computing with DNA molecules. DNA computing is a new computational paradigm that executes parallel computation with DNA molecules based on the Watson-Crick's law. The procedure of DNA computing can be divided into three stages: encoding information, computation (molecular operations) and extraction of solution. The stage of encoding information is the first and most important step, which directly affects the formation of optimal solution. The methods of encoding information can be divided into two classes: the methods of encoding information in graphs without weights and the methods of encoding information in graphs with weights. The previous researches, which belong to the first class, such as Adleman's encoding method [1] for the directed Hamiltonian path problem, Lipton's encoding method [2] for the SAT problem, and Ouyang's encoding method [3] for the maximal clique problem, do not require the consideration of weight representation in DNA strands. However, there are many practical applications related to weights. Therefore, weight representation in DNA strand is an important issue toward expanding the capability of DNA computing to solve optimization problems. Narayanan et al [6] presented a method of encoding weights by the lengths of DNA strands. Shin et al [6] proposed a method of encoding weights by the number of hydrogen bonds in fixed-length DNA strand. Yamamoto et al [7] proposed a method of encoding weights by the concentrations of DNA strands. Lee et al [9] proposed a method of encoding weights by the melting temperatures of fixed-length DNA strands. Han et al [10, 11] proposed a method of encoding weights by means of the general line graph. They also gave a method of encoding weights [12] by means of the relative length graph and several improved DNA encoding methods [13–16] for the maximal weight clique problem, the traveling salesman problem, the minimum spanning tree problem and the 0/1 knapsack problem. In this chapter, I collect and classify the present methods of encoding information in DNA strands, which will benefit the further research on DNA computing.

Supported by the Natural Science Foundation of Shandong Province of China.

A. Han and D. Zhu: *DNA Encoding Methods in the Field of DNA Computing*, Studies in Computational Intelligence (SCI) **94**, 293–322 (2008)
www.springerlink.com

13.1 Introduction

Bioinformatics studies the acquisition, process, store, distribution, analysis, etc of biological information so as to understand the meanings of biological data by means of mathematics, computer science and biological techniques. Some researches on Bioinformatics, such as the properties of DNA, the Watson-Crick's law, provide a probability of computing with DNA molecules, so DNA computing is an applied branch of Bioinformatics. The results of researches on Bioinformatics will improve the capabilities of DNA computing.

DNA computing is a computational paradigm that uses synthetic or natural DNA molecules as information storage media, in which the techniques of molecular biology, such as polymerase chain reaction, gel electrophoresis, and enzymatic reaction, are used as computational operators for copying, sorting, and splitting/concatenating information, respectively. Based on the massive parallelism of DNA computing, many researchers tried to solve a large number of difficult problems. In 1994, Adleman [1] solved a 7-vertex instance of the directed Hamiltonian path problem by means of the techniques of molecular biology. This creative research opened up a new way to computation with DNA molecules. A major goal of subsequent research in the field of DNA computing is to understand how to solve NP-complete problems. To address this goal, Lipton [2] abstracted a parallel molecular model on the basis of Adleman's experiment and applied it to solve the SAT problem; Ouyang *et al* [3] solved the maximal clique problem by means of DNA molecules; Head *et al* [4] solved the maximal independent set problem using operations on DNA plasmids; Sakamoto *et al* [5] presented a molecular algorithm of Boolean calculation by means of DNA hairpin formation. These previous researches on DNA computing do not require the consideration of weight representation in DNA strands.

However, there are many practical applications related to weights, such as the shortest path problem, the traveling salesman problem, the maximal weight clique problem, the Chinese postman problem, and the minimum spanning tree problem. Therefore, weight representation in DNA strand is an important issue toward expanding the capability of DNA computing to solve optimization problems. There exist previous works to represent weights in DNA molecules. Narayanan et al [6] presented a method of encoding weights by the lengths of DNA strands. Shin et al [6] proposed a method of encoding weights by the number of hydrogen bonds in fixed-length DNA strand. Yamamoto et al [7] proposed a method of encoding weights by the concentrations of DNA strands. Lee et al [9] proposed a method of encoding weights by the melting temperatures of fixed-length DNA strands. Han et al [10, 11] proposed a method of encoding weights by means of the general line graph. They also gave a method of encoding weights [12] by means of the relative length graph and several improved DNA encoding methods [13–16] for the maximal weight clique problem, the traveling salesman problem, the minimum spanning tree problem and the 0/1 knapsack problem.

In this chapter, we collect and classify the present DNA encoding methods in the field of DNA computing, which will benefit the further research on DNA computing.

13.2 Preliminaries to DNA Computing

In order to easily understand DNA encoding methods and the corresponding DNA algorithms, we first present some basic knowledge related to DNA computing.

13.2.1 Orientation of DNA Molecule

When DNA molecules combine with each other to form a DNA strand, $5'$-phosphate group of one nucleotide always combine with $3'$-hydroxyl group of another nucleotide by phosphodiester bonds, shown as P in Fig. 13.1. This is called as $5'$-$3'$ orientation or $3'$-$5'$ orientation [17, 18]. The nucleotide with $5'$ free-end being located at the most left end and $3'$ free-end being located at the most right end is marked as $5'$-$X_1X_2\ldots X_n$-$3'$, and the nucleotide with $3'$ free-end being located at the most left end and $5'$ free-end being located at the most right end is marked as $3'$-$X_1X_2\ldots X_n$-$5'$, where X_i denotes one letter in the alphabet $\{A, G, C, T\}$. Take Fig. 13.1 as an example. The DNA strand shown in Fig. 13.1(a) is marked as $5'$-*AGC*-$3'$, and the DNA strand shown in Fig. 13.1(b) is marked as $3'$-*CGA*-$5'$. Note that $5'$-*AGC*-$3'$ and $3'$-*CGA*-$5'$ are the same DNA molecules. In this chapter, we use the following representation: The DNA molecule $5'$-$X_1X_2\ldots X_n$-$3'$ is written as $X_1X_2\ldots X_n$, and $3'$-$X_1X_2\ldots X_n$-$5'$ is written as $-X_1X_2\ldots X_n$ [10, 11, 13, 14]. Note that $-X_1X_2\ldots X_n = X_nX_{n-1}\ldots X_1$.

Definition 13.2.1.1 For any DNA strand s, let h represent a mapping function from each base to its complement, or $h(A) = T$, $h(G) = C$, $h(C) = G$, $h(T) = A$. The obtained DNA strand $h(s)$ is called the complement of s, and its reversal $-h(s)$ is called the reverse complement of s. The mapping function h is called the complementary mapping from s to s' [10, 11, 14].

Take the DNA strand $s = AGC$ as an example. The complement of s is $h(s) = TCG$, and the reverse complement is $-h(s) = -TCG = GCT$. Obviously, the DNA strand AGC can combine with $-TCG$ to form a double-stranded DNA (dsDNA) through hydrogen bonds, as shown in Fig. 13.2. It can be concluded that any DNA strand s can combine with its reverse complement $-h(s)$ to form a dsDNA through

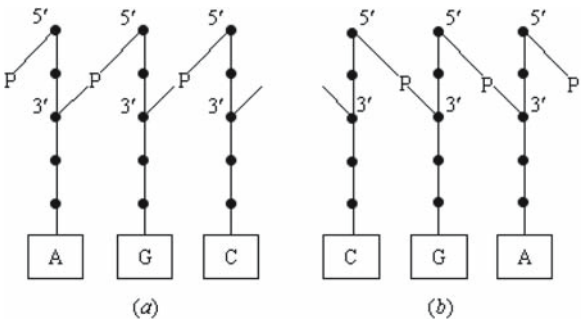

Fig. 13.1. Orientation of DNA molecule. (a) $5'$-$3'$ orientation and (b) $3'$-$5'$ orientation

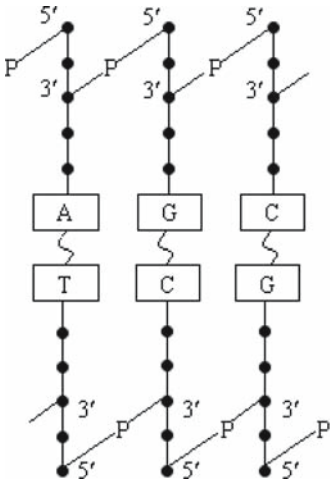

Fig. 13.2. A double-stranded DNA

hydrogen bonds. Note that DNA double strand consists of two DNA sequences $\alpha_1\alpha_2\ldots\alpha_k$ and $-\beta_1\beta_2\ldots\beta_k$ that satisfy the Watson-Crick's law, that is, for each $i = 1, 2, \ldots, k$, $k \in Z$, α_i and β_i must be complements, and the two complementary sequences anneal in an antiparallel fashion.

13.2.2 Basic Operations in DNA Computing

A tube [17] is defined as a multiset of words on the alphabet $\{A, C, G, T\}$. A multiset means a set in which the repeated words are regarded as different elements. For example, the multiset $\{AGC, AGC, GCTA\}$ has three elements, and the set $\{AGC, AGC, GCTA\}$ has two elements since the repeated words in a set are regarded as one element. Actually, a tube is a multiset of DNA strands. The basic operations [17] in DNA computing are as follows.

(1) *Merge*: For two tubes N_1 and N_2, it forms the unite $N_1 \cup N_2$ (multiset).

(2) *Amplify*: For a given tube N, it copies N into two shares (only for multiset).

(3) *Separate*: For a given tube N and a word w, $w \in \{A, C, G, T\}^*$, it generates two tubes: $+(N, w)$ and $-(N, w)$, where $+(N, w)$ consists of all the strands including w, and $-(N, w)$ consists of all the strands excluding w.

(4) $(N, \leq n)$: For a given tube N and an integer n, it generates a tube of all strands in N whose lengths are less than or equal to n.

(5) $B(N, w)$ and $E(N, w)$: For a given tube N and a word w, $B(N, w)$ generates a tube of all strands in N that begin with w, and $E(N, w)$ generates a tube of all strands in N that end with w.

(6) *Detect*: For a given tube N, it returns true if there exists at least one DNA strand in N, otherwise it returns false. This can be done through gel electrophoresis.

All the above operations can be implemented by means of the present biological techniques.

13.3 DNA Encoding Methods for the Problems Related to Graphs without Weights

Based on the massive parallelism of DNA computing, the previous researchers focused on NP-complete problems. These previous researches do not require the consideration of weight representation in DNA strands. Some of them are given in the following.

13.3.1 DNA Encoding Method for the Hamiltonian Path Problem

In 1994, Adleman [1] solved an instance of the directed Hamiltonian path problem by means of the molecular biology techniques. A 7-vertex graph was encoded in DNA strands and the operations were performed with standard DNA protocols and enzymes. This experiment demonstrates the feasibility of carrying out computations at the molecular level.

The Hamiltonian Path Problem

For a directed graph $G = (V, E)$, a path is called a Hamiltonian path if and only if it contains each vertex in G exactly once. A directed graph G with designated vertices v_{in} and v_{out} is said to have a Hamiltonian path if and only if there exists a path $e_1 e_2 \ldots e_n$ that begins with v_{in}, ends with v_{out} and enters every other vertex exactly once.

Fig. 13.3 shows a graph which for $v_{in} = 0$ and $v_{out} = 6$ has a Hamiltonian path, given by the edges 0→1, 1→2, 2→3, 3→4, 4→5, 5→6. If the edge 2→3 were removed from the graph, the resulting graph with the same designated vertices would not have a Hamiltonian path. Similarly, if the designated vertices were changed to $v_{in} = 3$ and $v_{out} = 5$, there would be no Hamiltonian path.

For a directed graph $G = (V, E)$, the Hamiltonian path problem (HPP) is to determine whether there exists a Hamiltonian path in it, that is, to find a directed path that starts with a given vertex, ends with another one, and visits every other vertex exactly once. HPP has been proved to be NP-complete. There are well known algorithms for deciding whether a directed graph with designated vertices has a Hamiltonian path, but all known deterministic algorithms for HPP have exponential worst-case complexity. In 1994, Adleman [1] designed a non-deterministic DNA algorithm for HPP which runs in a polynomial time.

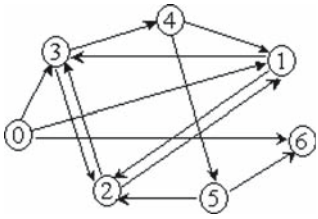

Fig. 13.3. The directed graph solved by Adleman

DNA Encoding Method

Given a directed graph $G = (V,E)$ with designated vertices $v_{in} = v_0$ and $v_{out} = v_6$, as shown in Fig. 13.3, Adleman [1] proposed the following DNA encoding method for solving the Hamiltonian path problem.

(1) Each vertex v_i in G was associated with a random 20-mer DNA strand denoted by s_i.

(2) For each edge $e_{ij} = (v_i, v_j)$ in G, a DNA strand s_{ij} was created which was $3'$ 10-mer of s_i (unless $i = 0$ in which case it was all of s_i) followed by $5'$ 10-mer of s_j (unless $j = 6$ in which case it was all of s_j).

The choice of random 20-mer DNA strand for encoding vertices in G is based on the following rationale [1]. First, choosing randomly several DNA strands from 4^{20} 20-mer DNA strands would unlikely share long common subsequences, which might result in unintended binding during the ligation step. Second, some deleterious features such as hairpin loops would unlikely arise in the several 20-mer DNA strands selected from 4^{20} 20-mer DNA strands. Finally, choosing 20-mers assured that binding between splint strands and edge strands would involve ten base pairs and would consequently be stable at room temperature.

DNA Algorithm

Given a directed graph $G = (V,E)$ with designated vertices $v_{in} = v_0$ and $v_{out} = v_t$. Based on the DNA encoding method, Adleman [1] designed the following DNA algorithm to solve the Hamiltonian path problem.

(1) *Merge*: All the DNA strands $h(s_i)$ and s_{ij} are mixed together in a single ligation reaction. Based on the Watson-Crick's law, generate random paths through the graph G.

(2) $Amplify$: The product of step 1 was amplified by polymerase chain reaction (PCR) using primers s_0 and $h(s_t)$. Thus, only those DNA molecules encoding paths that begin with v_0 and end with v_t were amplified. Keep only those paths that begin with v_{in} and end with v_{out} through the operations of $B(N, h(s_0))$ and $E(N, h(s_t))$.

(3) $(N, \leq 20n)$: Keep only those paths that enter exactly n vertices.

(4) $+(N, h(s_i))$: Keep only those paths that enter all the vertices at least once through the operation $+(N, h(s_i))$, $1 \leq i \leq n$, where n is the number of vertices in G

(5) $Detect$: If any paths remain, return $true$, otherwise return $false$. See the basic operations in section 13.2.2.

For more details, please see the reference [1].

13.3.2 DNA Encoding Method for the SAT Problem

Based on Adleman's experiment, Lipton [2] showed how to solve another famous NP-complete problem, the SAT problem. The advantage of the results is the huge parallelism inherent in DNA computing, which has the potential to yield vast speedups over conventional silicon-based computers.

The SAT Problem

Consider the Boolean formula $F = (x \vee y) \wedge (\neg x \vee \neg y)$, where the variables x and y are allowed to range only over the two values 0 and 1, $\vee$ is the logical OR operation, $\wedge$ is the logical AND operation, and $\neg x$ denotes the negation of x. Usually, one thinks of 0 as $false$ and 1 as $true$. The SAT problem is to find the Boolean values for x and y that make the formula F true.

In general, a Boolean formula is of the form $C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where $C_i (1 \leq i \leq m)$ is a clause; a clause is of the form $x_1 \vee x_2 \vee \cdots \vee x_k$, where x_i is a Boolean variable or its negation. The SAT problem is to find the values for the variables that make the formula have the value 1, that is, to find the values for the variables that make each clause have the value 1. The SAT problem has been proved to be NP-complete.

DNA Encoding Method

Given a Boolean formula F containing n variables $x_1, x_2, \ldots, x_n$, Lipton [2] designed the following DNA encoding method for the SAT problem.

(1) Construct a graph G with vertices a_1, x_1, x_1', a_2, x_2, x_2', $\ldots$, a_{n+1}, and with edges from a_i to both x_i and x_i' and from both x_i and x_i' to a_{i+1}, as shown in Fig. 13.4.

For the constructed graph G, each stage of a path has exactly two choices: If it takes the vertex with an unprimed label, or x_i, it encodes a 1; if it takes the vertex with a primed label, or x_i', it encodes a 0. For example, the path $a_1 x_1' a_2 x_2 a_3$ encodes the binary number 01. Obviously, each path starting with a_1 and ending with a_{n+1} in G encodes an n-bit binary number.

(2) Each vertex v_i $(1 \leq i \leq 3n+1)$ in G is assigned a random DNA strand s_i of length 20.

The DNA strand s_i corresponding to vertex v_i has two parts. The first half is denoted by s_i' and the second half is denoted by s_i''. That is, $s_i' s_i''$ is the code corresponding to vertex v_i, $v_i \in \{a_1, x_1, x_1', a_2, x_2, x_2', \ldots, a_{n+1}\}$.

(3) For each edge $e_{ij} = (v_i, v_j)$, $-h(s_i'' s_j')$ is used to encode it, where $-h(x)$ denotes the reverse complement of x.

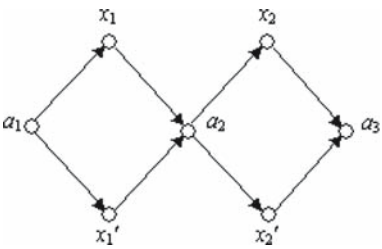

Fig. 13.4. The graph corresponding to a Boolean formula that contains 2 variables x_1, x_2

DNA Algorithm

For the DNA algorithm proposed by Lipton [2], let $S(t,i,a)$ denote all the sequences in tube t for which the ith bit is equal to a, $a \in \{0,1\}$. This is done by performing one extraction operation that checks for the sequence corresponding to x_i {if $a = 1$} or x_i' {if $a = 0$}. Consider the Boolean formula $F = (x \vee y) \wedge (\neg x \vee \neg y)$. The following steps can solve it.

(1) Construct a graph G with vertices a_1, x_1, x_1', a_2, x_2, x_2', a_3, and with edges from a_i to both x_i and x_i' and from both x_i and x_i' to a_{i+1}, $1 \leq i \leq 2$. Each vertex in G is assigned a random DNA strand of length 20. For each edge $e_{ij} = (v_i, v_j)$, $-h(s_i'' s_j')$ is used to encode it.

(2) Let t_0 be the tube including the sequences corresponding to $a_1x_1a_2x_2a_3$, $a_1x_1'a_2x_2a_3$, $a_1x_1a_2x_2'a_3$, $a_1x_1'a_2x_2'a_3$.

(3) Let tt_1 be the tube corresponding to $S(t_0,1,1)$. Let the reminder be tt_1', and tt_2 be $S(tt_1',2,1)$. Pour tt_1 and tt_2 together to form t_1.

(4) Let tt_3 be the tube corresponding to $S(t_1,1,0)$. Let the reminder be tt_3' and tt_4 be $S(tt_3',2,0)$. Again pour tt_3 and tt_4 together to form t_2.

(5) Detect DNA in the last tube t_2. If there is any DNA in t_2, the formula is satisfiable.

Now consider the SAT problem on n variables and m clauses. Suppose that, as is usual, each clause consists of a fixed number of variables or their negations. Let C_1, C_2, …, C_m be the m clauses. A series of tubes t_0, t_1, …, t_m are constructed so as to let t_k be the set of n-bit numbers in which each element x satisfies that $C_1(x) = C_2(x) = \cdots = C_k(x) = 1$, where $C_i(x)$ is the value of C_i on x.

(1) Construct a graph G with vertices a_1, x_1, x_1', a_2, x_2, x_2', …, a_{n+1}, and with edges from a_i to both x_i and x_i' and from both x_i and x_i' to a_{i+1}, $1 \leq i \leq n$. Each vertex in G is assigned a random DNA strand of length 20. For each edge $e_{ij} = (v_i, v_j)$, the DNA strand $-h(s_i'' s_j')$ is used to encode it.

(2) Let t_0 be the tube including all n-bit sequences.

(3) Construct t_{k+1}, $k = 0, 1, \ldots, m-1$, step by step. Let C_{k+1} be the clause $x_1 \vee x_2 \vee \ldots x_l$, where x_i is a literal or its negation. For each literal x_i, if x_i is equal to x_j, then form $S(t_k, j, 1)$; if it is equal to $\neg x_j$, then form $S(t_k, j, 0)$. The reminder of each extraction is used for the next step. Pour all the reminders together to form t_{k+1}.

(4) Detect DNA in the tube t_m. If there is any DNA in t_m, the formula is satisfiable.

For more details, please see the reference [2].

13.3.3 DNA Encoding Method for the Maximal Clique Problem

In 1997, Ouyang [3] solved the maximal clique problem using the techniques of molecular biology. A pool of DNA molecules corresponding to the ensemble of six-vertex cliques was built, followed by a series of selection processes. The algorithm is highly parallel and has satisfactory fidelity.

The maximal Clique Problem

Mathematically, a clique is defined as a subset of vertices in a graph, in which each vertex is connected to all other vertices in the subset. The clique including the most vertices is called the maximal clique. The maximal clique problem asks: Given an undirected graph with n vertices and m edges, how many vertices are in the maximal clique? The corresponding decision problem has been proved to be NP-complete. Take the graph shown in Fig. 13.5(a) as an example. The vertices $(5, 4, 3, 2)$ form the maximal clique, that is, the size of the maximal clique is four.

DNA Encoding Method

For an undirected graph $G = (V, E)$ with n vertices, Ouyang [3] designed the following DNA encoding method to solve the maximal clique problem.

(1) Each clique in G is represented as an n-bit binary number. A bit set to 1 represents the vertex being in the clique, and a bit set to 0 represents the vertex being out of the clique. Thus, the set of all the cliques in G is transformed into the ensemble of n-bit binary numbers, which is called the complete data pool [3].

(2) Each bit in a binary number corresponds to two DNA sections: one for the bit's value (V_i) and another for its position (P_i). The length of V_i is set to 10 base pairs if the value of V_i is equal to 0, and 0 base pair if the value of V_i is equal to 1. Thus, the ensemble of DNA strands representing the complete data pool is constructed.

(3) The restriction sequence is embedded within V_i if the value of V_i is equal to 1.

Thus, for a DNA strand representing an n-bit binary number, there are n value sections (V_0 to V_{n-1}) sandwiched sequentially between $n+1$ position sections (P_0 to P_n). The last position section P_n is needed for PCR amplification.

DNA Algorithm

In order to easily understand the DNA algorithm for the maximal clique problem, the definition of complementary graph [3] is given as follows.

Definition 13.3.3.1 For an undirected graph $G = (V, E)$, the graph $G' = (V, E')$ only containing all the connections absent in G is called the complementary graph of G.

Take the graph shown in Fig. 13.5(a) as an example, Fig. 13.5(b) is its complementary graph. According to definition 13.3.3.1, any two vertices connected in the

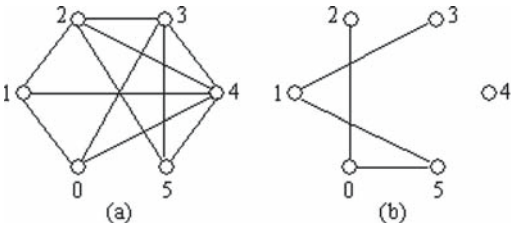

Fig. 13.5. A graph and its complementary graph. (a) An undirected graph G and (b) The complementary graph G' of G

complementary graph are disconnected in the original graph and therefore cannot be members of the same clique; this means that the corresponding bits cannot both be set to 1. Based on this, Ouyang [3] devised the following DNA algorithm for the maximal clique problem.

(1) Generate the random sequence of each P_i and V_i, and then construct the complete data pool by means of the technique of parallel overlap assembly (POA) [3].

The construction starts with $2n$ DNA strands: $P_0V_0^0P_1$, $P_0V_0^1P_1$, $-h(P_1V_1^0P_2)$, $-h(P_1V_1^1P_2)$, $P_2V_2^0P_3$, $P_2V_2^1P_3$, $-h(P_3V_3^0P_4)$, $-h(P_3V_3^1P_4)$, etc. Each DNA strand consists of two position motifs and one value motif, $P_iV_iP_{i+1}$ for even i and $-h(P_iV_iP_{i+1})$ for odd i, where the value of V_i can be 0 or 1. The $2n$ DNA strands were mixed together for thermal cycling [3]. During each thermal cycle, the position string in one DNA strand is annealed to the reverse complement of the next DNA strand. After a few thermal cycles, a data pool with all combinations of $V_0V_1V_2\ldots V_{n-1}$ was built. The POA procedure was followed by PCR, and the molecules with P_0 and P_n at their ends were exponentially amplified.

(2) Eliminate all the numbers containing connections in the complementary graph from the complete data pool. The remainder corresponds to all the cliques in the original graph.

Guided by the complementary graph, the data pool was digested with restriction enzymes. These enzymes break DNA at specific restriction sites, which were embedded within the sequences for $V_i = 1$. Consider 0-2 connection in the complementary graph, that is, there were xx…x0x0, xx…x0x1, xx…x1x0, and xx…x1x1 in the data pool. The data pool was divided into two tubes, t_0 and t_1. In t_0, the DNA strands containing $V_0 = 1$ were cut with Afl II. Thus, t_0 contained only xx…x0x0 and xx…x1x0. In t_1, the DNA strands containing $V_2 = 1$ were cut with Spe I. Thus, t_1 contained only xx…x0x0 and xx…x0x1. And then, t_0 and t_1 were put into tube t, which contained xx…x0x0, xx…x1x0 and xx…x0x1. That is, tube t did not contain xxx1x1.

(3) Sort the remaining data pool to find the data containing the largest number of 1's. The clique with the largest number of 1's tells us the size of the maximal clique.

For more details, please see the reference [3].

13.4 DNA Encoding Methods for the Problems Related to Graphs with Weights

The previous works deal with the problems related to graphs without weights. The significance of these researches is that they demonstrate how DNA can be used for representing information and solving the problems in the complexity class NP. However, there are many practical applications related to weights. Representation of weight information is one of the most important but also challenging problems in DNA computing. Some of the methods representing weights in DNA strands are given in the following.

13.4.1 Encoding Weights by the Lengths of DNA Strands

In 1998, Narayanan *et al* [6] proposed a DNA encoding method of representing weights for the shortest path problem. For a connected, weighted graph $G = (V, E)$, the shortest path problem is to find a path with minimum cost (weight) that begin with a specified vertex and end with another one.

DNA Encoding Method

For a connected, weighted graph $G = (V, E)$ with n vertices and m edges. Narayanan *et al* [6] designed the following DNA encoding method for the shortest path problem.

(1) For each vertex v_i ($i = 1, 2, \ldots, n$), assign a unique DNA sequence s_i with fixed length to encode it.

(2) Sort all the edges in G by distance (weight), and put their distances into a vector **D**. For each distance d in **D**, a DNA sequence s_d is randomly selected whose length l is associated with the location of distance d and a constant factor k. Consider $k = 3$ and $D = \{2, 5, 9, 10\}$, 2 is represented by a strand of length 3, 5 is represented by a strand of length 6, and so on.

(3) For each edge $e_{ij} = (v_i, v_j)$, the DNA strand s_{idj} is created in the following way: if $i = 1$, create the strand s_{idj} as ALL s_i + ALL s_d + HL s_j; if $i > 1$, create the strand s_{idj} as HR s_i + ALL s_d + HL s_j, where ALL represents the whole DNA strand, HL the left half, HR the right half, and + the join operation.

DNA Algorithm

For a connected, weighted graph $G = (V, E)$, the following steps adapted from Adleman's DNA algorithm [1] can extract the shortest path between the initial vertex v_1 and the destination vertex v_t.

(1) *Merge*: Put all the DNA strands $h(s_i)$ and s_{idj} into a tube, and perform a DNA ligase reaction in which random paths through G are formed.

(2) $Amplify$: The strands beginning with v_1 are amplified through a polymerase chain reaction using primers s_1 and $h(s_t)$.

(3) $B(N, s_1)$: Keep only the strands beginning with v_1.

(4) $E(N, s_t)$: Keep only the strands ending with v_t.

(5) All the obtained strands are sorted through gel electrophoresis. The shortest strand corresponds to the desired solution.

For more details, please see the reference [6].

13.4.2 Encoding Weights by the Number of Hydrogen Bonds

Shin *et al* [7] presented an encoding method that uses fixed-length codes for representing integer and real values. In this method, the relative values of *G*/*C* contents against *A*/*T* contents are taken into account to represent weights in the graph, which is based on the fact that hybridization between *G*/*C* pairs occurs more frequent than

those between A/T pairs because there are 3 hydrogen bonds between G and C, whereas 2 hydrogen bonds between A and T. Generally, the ligation between DNA sequences is influenced by DNA length and the G/C contents [7]: The longer the sequences, the more often they get hybridized; the more G/C pairs the sequences have, the more probability they get hybridized.

Shin's method was applied to the traveling salesman problem. For a connected, weighted graph, the traveling salesman problem is to find a minimum cost (weight) path that begins with a specified vertex and ends there after passing through all other vertices exactly once.

DNA Encoding Method

For a connected, weighted graph $G = (V,E)$, the DNA encoding method proposed by Shin [7] is as follows.

(1) For each vertex v_j, which is the common vertex of edges e_{ij} and e_{jk}, the DNA sequence s_j consists of 4 components: 10 bp weight sequence $h(W''_{ij})$, 10 bp position sequence P'_j, 10 bp position sequence P''_j, and 10 bp weight sequence $h(W'_{jk})$, where x' denotes the first half of x, and x'' denotes the last half of x.

(2) For each edge $e_{ij} = (v_i, v_j)$, the DNA sequence s_{ij} also consists of 4 components: 10 bp link sequence $h(P''_i)$, 10 bp weight sequence W'_{ij}, 10 bp weight sequence W''_{ij}, and 10 bp link sequence $h(P'_j)$. The orientation of edge code is opposite to that of vertex code.

For more details, please see the reference [7].

DNA Algorithm

The DNA algorithm proposed by Shin [7] consists of two parts: a genetic algorithm for optimizing the DNA codes, and a molecular algorithm for simulating the DNA computing process.

Algorithm for genetic code optimization

The codes are optimized using the following genetic algorithm.

(1) For each vertex v_i, generate randomly the vertex position sequences P_i, $1 \leq i \leq n$.

(2) For each edge $e_{ij} = (v_i, v_j)$, generate the edge link sequences, that is, the reverse complements of the last half of the vertex position sequence P_i and the first half of the vertex position sequence P_j.

(3) For each weight w_{ij} on edge e_{ij}, generate randomly the edge weight sequences W_{ij}.

(4) Generate the vertex weight sequences according to the edge weight sequences.

(5) While (*generation* $g \leq g_{max}$) do {Evaluate the fitness of each code; Apply genetic operators to produce a new population}.

(6) Let the best code be the fittest one.

In the genetic algorithm, the amount of *G/C* contents in edge sequences is optimized in step 5 so as to let the edges with smaller weights have more *G/C* contents and thus have higher probability being contained in the final solution. The fitness function is to promote the paths with lower costs (path lengths) so that the minimum cost path could be found. Let N_{eij} denote the number of hydrogen bonds in edge e_{ij}, S_h denote the total number of hydrogen bonds in all edges, W_{eij} denote the weight on edge e_{ij}, and S_w denote the *sum* of the weight values. The fitness function is defined as follows: if $|N_{eij}/S_h - W_{eij}/S_w| \leq \theta$, then $F_i = |N_{eij}/S_h - W_{ei}/S_w|$; otherwise, $F_i = 0$, where the threshold value θ is determined by experiments.

Molecular algorithm

The molecular algorithm adopted the same as the iterative version of molecular programming [19]. The iterative molecular algorithm (IMA) iteratively evolves fitter sequences rather than simply filtering out infeasible solutions. This procedure is summarized as follows.

(1) Encoding: Determine the code sequence using the algorithm of genetic code optimization.

(2) While ($cycle\ c \leq c_{max}$) do {*Synthesis*: Produce candidate solutions by molecular operators; *Separation*: Filter out infeasible solutions by laboratory steps}.

(3) Keep only those paths that begin with V_{in} and end with V_{in}.

(4) Keep only those paths that enter exactly $n+1$ vertices, where n is the number of vertices in the graph.

(5) Keep only those paths that enter all the vertices at least once.

(6) Select the path that contains the largest amount of G/C pairs, which corresponds to the minimum cost path.

For more details, please see the reference [7].

13.4.3 Encoding Weights by the Concentrations of DNA Strands

Yamamoto *et al* [8] presented a method of encoding weights by the concentrations of DNA strands, and used it to the shortest path problem.

DNA Encoding Method

For a connected, weighted graph $G = (V,E)$, the DNA encoding method proposed by Yamamoto [8] is as follows.

(1) Each vertex v_i in G is associated with a 20-mer DNA sequence denoted by s_i.

(2) For each edge $e_{ij} = (v_i, v_j)$ in G, a DNA strand s_{ij} that is $3'$ 10-mer of s_i followed by $5'$ 10-mer of s_j is created. The relative concentration D_{ij} of s_{ij} is calculated by the following formula: $D_{ij} = (min/w_{ij})^{\alpha}$, where *min* represents the minimum weight in G, w_{ij} represents the weight on edge e_{ij}, and α is a parameter value.

DNA Algorithm

Based on the DNA encoding method, the DNA algorithm [8] for the shortest path problems is as follows.

(1) For each vertex v_i in G, set the concentration of $h(s_i)$ to a certain value. Note that the concentrations of all the DNA strands $h(s_i)$ are set to the same value.

(2) For each edge e_{ij} in G, calculate the relative concentration D_{ij} of s_{ij} according to the formula $D_{ij} = (min/w_{ij})^{\alpha}$.

(3) Put all the DNA strands $h(s_i)$ with the same concentration and all the DNA strands s_{ij} with different concentrations D_{ij} to construct random paths through G.

(4) Amplify the DNA paths that begin with the start vertex and end with the destination vertex.

(5) Determinate the DNA strand of encoding the shortest path.

For more details, please see the reference [8].

13.4.4 Encoding Weights by the Melting Temperatures of DNA Strands

Lee *et al* [9] introduced a DNA encoding method to represent weights based on the thermodynamic properties of DNA molecules, and applied it to the traveling salesman problem. This method uses DNA strands of fixed-length to encode different weights by varying the melting temperatures, in which the DNA strands for higher-cost values have higher melting temperatures than those for lower-cost values.

DNA Encoding Method

For an instance of the traveling salesman problem, Lee *et al* [9] gave the following method of encoding weights.

(1) Each city sequence is designed to have a similar melting temperature. That is, city sequences contribute equally to the thermal stability of paths.

(2) Cost sequences are designed to have various melting temperatures according to the costs. A smaller cost is represented by a DNA sequence with a lower melting temperature.

(3) Road sequences that connect two cities are generated using the sequences of departure cities, costs, and arrival cities. The first part of the road sequence is the complement of the last half of the departure city, the middle part represents the cost information, and the last part is the complement of the first half of the arrival city.

There are several empirical methods to calculate the melting temperatures. One of them is the *GC* content method that uses the content of *G* and *C* in DNA strand as a main factor determining melting temperature.

For more details, please see the reference [9].

DNA Algorithm

The DNA algorithm modifies the PCR protocol to employ temperature gradient in the denaturation step. The denaturation temperature is low at the beginning of PCR

and then it increases gradually. With help of the denaturation temperature gradient PCR (DTG-PCR), the more economical paths of lower T_m can be amplified more intensively. The DNA algorithm presented by Lee *et al* [9] for the traveling salesman problem is as follows.

(1) Generate the answer pool through the operations of hybridization and ligation.

(2) Select the paths satisfying the conditions of the traveling salesman problem through the operations of PCR with primers and affinity-separation.

(3) Amplify the more economical paths through the operation of DTG-PCR.

(4) Separate the most economical path among the candidate paths.

(5) Read the final path through sequencing.

For more details, please see the reference [9].

13.4.5 Encoding Weights by Means of the General Line Graph

Han *et al* [10,11] proposed a DNA encoding method to represent weights and applied it to the Chinese postman problem, an instance of optimization problems on weighted graphs. For a weighted, undirected graph $G = (V,E)$, Han *et al* first convert it into its general line graph $G' = (V',E')$, and then design the DNA encoding method based on G'.

The Chinese Postman Problem

The Chinese postman problem is to find a minimum cost tour that a postman sets out from the post office, walks along each street to deliver letters, and returns to the post office. If the layout of streets is an Euler graph, the Euler tour is just what we want; otherwise, he needs to walk along some streets more than once. The Chinese postman problem can be abstracted as follows: For a connected, weighted, undirected graph $G = (V,E)$, $v_i \in V$, $1 \leq i \leq n$, $e_j \in E$, $1 \leq j \leq m$, where the weight on edge e_j is w_j, $w_j \geq 0$, $w_j \in Z$, it is to find a minimum cost (weight) tour that begins with a specified vertex and ends there after passing through all the given edges. That is, the Chinese postman problem is to find a shortest tour that goes through all the edges in G.

Construction of General Line Graph

Definition 13.4.5.1 For an undirected graph $G = (V,E)$, $v_i \in V$, $1 \leq i \leq n$, $e_j \in E$, $1 \leq j \leq m$, a mapping function f is constructed to satisfy: (1) For each edge $e_j \in E$, there exists only one vertex v'_j to satisfy $f(e_j) = v'_j$; (2) If any two edges e_i and e_j are adjacent, draw an undirected edge between v'_i and v'_j; (3) If v_i is with odd degree, maximum one self-loop is added to each of the vertices which are mapped from the edges linked to v_i. The function f is called the mapping function from edges to vertices, and the obtained graph is called the general line graph of G.

Take the weighted, undirected graph G shown in Fig. 13.6(a) as an example. The procedure of mapping from edges to vertices is as follows. (1) The edges e_1, e_2, ...,

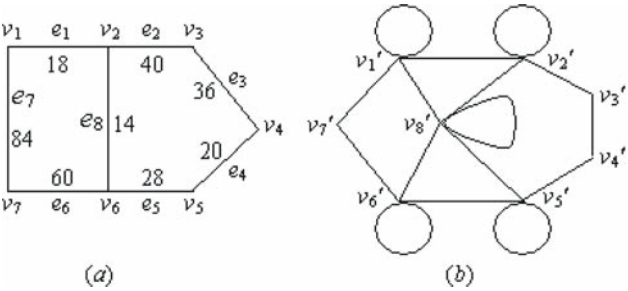

Fig. 13.6. A weighted graph G and its general line graph G'. (a) A weighted graph G and (b) The general line graph G' of G

e_8 are respectively mapped to the vertices $v'_1, v'_2, \ldots, v'_8$. (2) Adding the undirected edges. The vertex v'_1 is linked to v'_2, v'_8, v'_7 since the edge e_1 is adjacent to e_2, e_8, e_7. The vertex v'_2 is linked to v'_3, v'_8, v'_1 since the edge e_2 is adjacent to e_3, e_8, e_1. Similar operation is done for other vertices $v'_3, v'_4, \ldots, v'_8$. (3) Adding the self-loops. A self-loop is respectively added to v'_1, v'_2 and v'_8 since v_2 is with odd degree and v_2 is linked to the edges e_1, e_2 and e_8. A self-loop is respectively added to v'_5 and v'_6 since v_6 is with odd degree and v_6 is linked to the edges e_5, e_6 and e_8. The obtained general line graph G' is shown in Fig. 13.6(b).

By means of the mapping function from edges to vertices, Han *et al* [10, 11] convert the problem of searching for the shortest tour that pass through each edge at least once into that of searching for the shortest tour that pass through each vertex at least once. Note that the shortest tour may be not only one. For example, the shortest tours in Fig. 13.6(a) are $v_1e_1v_2e_2v_3e_3v_4e_4v_5e_5v_6e_8v_2e'_8v_6e_6v_7e_7v_1$ and $v_1e_1v_2e'_8v_6e_8v_2e_2v_3e_3v_4e_4v_5e_5v_6e_6v_7e_7v_1$, where e'_8 denotes the reversal of e_8. In order to easily observe, we use edge sequence to denote the shortest tour in G, such as $e_1e_2e_3e_4e_5e_8e'_8e_6e_7$, and use vertex sequence to denote the shortest tour in G', such as $v'_1v'_2v'_3v'_4v'_5v'_8v'_8v'_6v'_7$.

DNA Encoding Method

Given a connected, weighted, undirected graph $G = (V,E)$, $v_i \in V$, $1 \leq i \leq n$, $e_j \in E$, $1 \leq j \leq m$, where the weight on edge e_j is w_j, $w_j \geq 0$, $w_j \in Z$. If w_i is a real number, all the weights are multiplied by a certain integer (i.e. 10) and then they are rounded into integers. The main idea of the DNA encoding method proposed by Han *et al* [10,11] is as follows: The given graph G is firstly converted into its general line graph $G' = (V',E')$, $v'_i \in V'$, $1 \leq i \leq m$, where v'_i is mapped from e_i. For each vertex v'_i, use DNA strand s_i of length w_i to encode it. For each edge $e'_{ij} = (v'_i, v'_j)$, use the DNA strand s_{ij}, which is the reverse complement of the last half of s_i and the first half of s_j, to encode it. Note that the DNA strands to encode vertices are of different lengths. The detailed encoding method [10, 11] for the Chinese postman problem is as follows:

(1) All the edges e_j ($1 \le j \le m$) are mapped to vertices v'_j. If e_i and e_j are adjacent, an undirected edge is drawn between v'_i and v'_j. If v_i is with odd degree, maximum one self-loop is added to each of the vertices which are mapped from the edges linked to v_i.

(2) For each vertex v'_i, use DNA strand s_i of length w_i to encode it.

(3) For each edge $e'_{ij} = (v'_i, v'_j)$, use the reverse complement of the last half of s_i and the first half of s_j to encode it. Specifically, s_i and s_j are firstly divided into two substrands with equal length, or $s_i = s'_i s''_i$, $s_j = s'_j s''_j$. And then use the DNA strand $s_{ij} = -h(s''_i s'_j)$ to encode edge $e'_{ij} = (v'_i, v'_j)$, where s_{ij} is with the length of $|s_i|/2+|s_j|/2$. Here, suppose that weights in the weighted graph are all even. If there exists one or more weights are odd in a practical problem, all the weights are multiplied by 2. Thus, a half of the optimal solution is the desired results.

Note that, for any undirected edge $e'_{ij} = (v'_i, v'_j)$, if walk from v'_i to v'_j, the code s_{ij} is the reverse complement of s''_i and s'_j, or $-h(s''_i s'_j)$; if walk from v'_j to v'_i, the code s_{ji} is the reverse complement of $(-s_j)''$ and $(-s_i)'$, or $s_{ji} = -h((-s_j)''(-s_i)') = h(s''_i s'_j) = -s_{ij}$. That is, only need one code $s_{ij} = -h(s''_i s'_j)$ to encode edge $e'_{ij} = (v'_i, v'_j)$.

Take the weighted graph G shown in Fig. 13.6(a) as an example. We specifically analyze the proposed DNA encoding method. First of all, the general line graph G' is converted from the given graph G, as shown in Fig. 13.6(b). For the vertices v'_1, $v'_2, \ldots, v'_8$ in the general line graph G', the following DNA strands $s_1, s_2, \ldots, s_8$ with the lengths of 18, 40, 36, 20, 28, 60, 84, 14 are respectively selected to encode them.

$s_1 = CAGTTGACATGCAGGATC$

$s_2 = CAACCCAAAACCTGGTAGAGATATCGCGGGTTCAACGTGC$

$s_3 = TAGTACTGATCGTAGCAACCTGGTACCAAGCTTGAC$

$s_4 = CGCATGCAGGATTCGAGCTA$

...

$s_8 = TGGTTTGGACTGGT$

For each edge $e'_{ij} = (v'_i, v'_j)$, the DNA strand $s_{ij} = -h(s''_i s'_j)$ is used to encode it. For example, the code of $e'_{12} = (v'_1, v'_2)$ is as follows.

$s_{12} = -h(s_1'' s'_2) = h(TGCAGGATCCAACCCAAAACCTGGTAGAG)$

$=ACGTCCTAGGTTGGGTTTTGGACCATCTC$

Obviously, s_{12} is with the length of $18/2+40/2=29$. The joint among the DNA strands of encoding edge $e'_{12} = (v'_1, v'_2)$ and vertices v'_1, v'_2 is shown in Fig. 13.7.

On the basis of Fig. 13.7, the next is to extend rightward to s_3 or s_8, and to extend leftward to s_7 or s_8. The extension of the DNA strand $s_1 s_2$ rightward is shown in Fig. 13.8, and its extension leftward can similarly be drawn up.

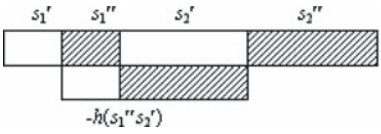

Fig. 13.7. Joint of DNA strands s_1, s_2 and s_{12}

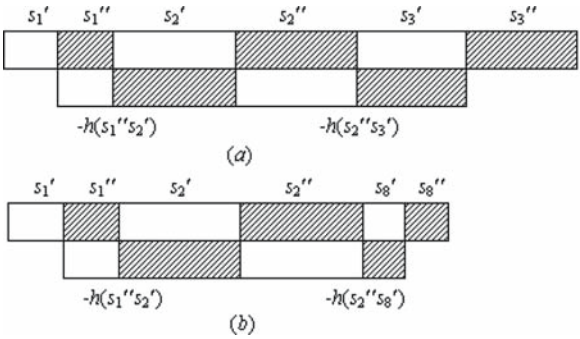

Fig. 13.8. Extension rightward of DNA strand s_1s_2. (a) Joint of DNA strands s_1, s_2, s_3, s_{12}, s_{23} and (b) Joint of DNA strands s_1, s_2, s_8, s_{12}, s_{28}

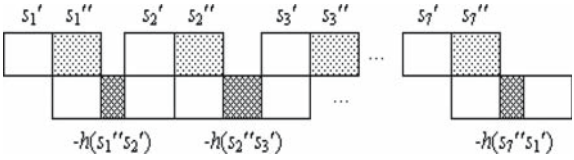

Fig. 13.9. Alternant DNA strand and double strand

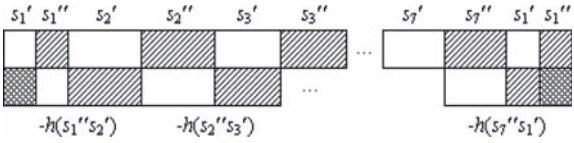

Fig. 13.10. Double-stranded DNA corresponding to the optimal solution

On the basis of Fig. 13.8(a), the next is to extend rightward to s_4, and to extend leftward to s_7 or s_8, and so on. Thus, with the help of the property of reverse complementation between vertex codes and edge codes, the DNA strands may extend continually to form various random paths including the optimal solution.

In the DNA encoding method [10, 11], the paths generated in a single ligation reaction are double-stranded DNA instead of alternant DNA strand and double strand. An alternant DNA strand and double strand is shown in Fig. 13.9, and a double-stranded DNA is shown in Fig. 13.10. It is well known that the stable structure of DNA molecules is DNA double strand. In an alternant DNA strand and double strand, the part of DNA strand can combine with other molecules through hydrogen bonds based on the Watson-Crick's law. Since DNA double strand are more stable than alternant DNA strand and double strand, the proposed DNA encoding method can more easily generate the optimal solution. In addition, the proposed DNA encoding method uses DNA strands of different lengths to encode different vertex. It also has characteristics of easy encoding and low error rate. But when the values of the weights are very large, the lengths of the DNA strands are very long which result in higher space complexity. For more details, please see the reference [10, 11].

DNA Algorithm

For the general line graph G' converted from the given graph G, suppose that v'_1 is the original vertex just as well because, for any shortest tour C going through all the edges in G', the length of the route beginning with v'_1 and ending there along C is equal to that of the route beginning with v'_i $(i \neq 1)$ and ending there along C.

In order to easily generate the optimal solution, the DNA algorithm proposed by Han *et al* [10, 11] searches for the shortest path instead of the shortest tour. The reason is that the length of a shortest tour that begins with v'_1 and end there after passing through all the edges in G' is equal to that of a shortest path that begins with v'_1 and end there after passing through all the edges in G'. Moreover, the polymerase chain reaction (PCR) in the biological techniques is generally carried out on a linear template; there is no circular template so far. The detailed DNA algorithm [10, 11] for the Chinese postman problem is as follows.

(1) *Merge*: The DNA strands s_i and s_{ij} $(1 \leq i, j \leq m)$ are mixed together in a single ligation reaction. Based on the Watson-Crick's law, generate various DNA molecules corresponding to the random paths.

(2) $Amplify$: The product of step 1 is amplified by polymerase chain reaction (PCR) using primers $-h(s''_1)$ and $-h(s'_1)$. Thus, only those DNA molecules encoding paths that begin with v'_1 and end with v'_1 were amplified.

(3) $B(N, s_1)$: Separate all the paths with the departure vertex v'_1, or separate all the DNA molecules with $5'$ end being s_1.

(4) $E(N, s_1)$: Separate all the paths with the arrival vertex v'_1, or separate all the DNA molecules with $3'$ end being s_1.

(5) $+(N, s_i)$: For each vertex v'_i $(2 \leq i \leq m)$, separate all the paths including v'_i.

(6) Separate the shortest path through gel electrophoresis.

(7) Determinate the nucleotides sequence of the shortest path, which corresponds to the optimal solution.

For more details, please see the reference [10, 11].

13.4.6 RLM: Relative Length Method of Encoding Weights

Han [12] presented a method of encoding weights in DNA strands for the problems related to graph with weights, which is referred to the relative length method (RLM), and applied it to the traveling salesman problem. The RLM method can directly deal with weights of either real numbers or integers, even very small and very big positive weights, and the lengths of DNA strands used in the RLM method are not proportional to the values of weights.

Definitions Involved in the RLM Method

Definition 13.4.6.1 For a weighted graph $G = (V, E)$, $v_i \in V$, $1 \leq i \leq n$, $e_j \in E$, $1 \leq j \leq m$, all the weights are sorted in a nondecreasing order, and the equal weights are at the same position. Thus, all the weights are divided into p groups $(p \leq m)$ according to their ranking. The p groups are numbered from 1 to p, respectively. The group number is called the order number of the weight.

Definition 13.4.6.2 For a weighted graph $G = (V,E)$, $v_i \in V$, $e_{ij} \in E$, $1 \le i, j \le n$, where the weight on edge e_{ij} is w_{ij}, all the weights w_{ij} are remarked as $w_{ij,k}$, where k is the order number of w_{ij}. For each remarked weight $w_{ij,k}$, we add $k-1$ nodes on edge e_{ij}. The obtained graph G' is called the relative length graph of G.

Obviously, if the weight w_{ij} is remarked as $w_{ij,k}$, the edge e_{ij} will be divided into k segments. The bigger the order number, the more the segments of the edge. That is, the segment number of an edge represents the relative length of the edge. Note that the segment number of an edge is not directly proportional to the weight on the edge. For example, the segment numbers of edges with weights 2, 1000 and 1002 are 1, 2 and 3, respectively.

RLM Method of Encoding Weights

With the help of the relative length graph, Han [12] devised a method of encoding weights in DNA strands for the traveling salesman problem. For a weighted graph $G = (V,E)$ with n vertices and m edges, $v_i \in V$, $e_{ij} \in E$, $1 \le i, j \le n$, where the weight on edge e_{ij} is w_{ij}, the RLM method [12] is as follows.

(1) All the weights are divided into p groups ($p \le m$) according their order numbers, and each weight w_{ij} is remarked as $w_{ij,k}$ if it belongs to the kth group ($1 \le k \le p$).

(2) For each remarked weight $w_{ij,k}$, we add $k-1$ nodes on edge e_{ij}. The added nodes are marked as $v_{eij,1}, v_{eij,2}, \ldots, v_{eij,k-1}$, respectively. The obtained graph G' is the relative length graph of G.

(3) For each vertex and each added node, we use DNA strand s_i of length $2c$ ($c \in Z$, $c \ge 5$) to encode it. The DNA strand s_i is divided into two sub-strands with equal length, or $s_i = s_i' s_i''$. See the DNA encoding method in section 13.4.5.

(4) For each edge e_{ij} (including the edges that are connecting the nodes newly added in step 2), we use DNA strand $s_{ij} = -h(s_i'' s_j')$ to encode it, where $-h(s)$ denotes the reverse complement of s. Thus, when the vertex-node codes and the edge codes are mixed together, they can combine with each other to form dsDNAs since any DNA strand s can combine with its reverse complement $-h(s)$ to form dsDNA.

Take the graph G shown in Fig. 13.11(a) as an example. All the weights are sorted in a nondecreasing order, or 1.2, 2, 2.5, 3. Thus, they are divided into 4 groups, and the order numbers of weights 1.2, 2, 2.5, 3 are 1, 2, 3, 4, respectively. Therefore, the weights w_{12}, w_{13}, w_{14}, w_{15}, w_{23}, w_{34}, w_{35}, w_{45} are remarked as $w_{12,1}$, $w_{13,3}$, $w_{14,4}$, $w_{15,2}$, $w_{23,4}$, $w_{34,2}$, $w_{35,1}$, $w_{45,3}$, respectively. For each remarked weight $w_{ij,k}$, we add $k-1$ nodes on edge e_{ij}. For example, we add two nodes on e_{13} since w_{13} is remarked as $w_{13,3}$, we add three nodes on e_{14} since w_{14} is remarked as $w_{14,4}$, and so on. The obtained graph G' is the relative length graph of G, as shown in Fig. 13.11(b).

For each vertex v_i in Fig. 13.11(b), we use DNA strand s_i of length 10 ($c = 5$) to encode it. Here, the added nodes are also viewed as vertices. Consider the vertex v_1 and the node $v_{e13,1}$. The DNA strands $s_1 = TTAGCGCATG$, $s_{e13,1} = GTTACGTGAG$ are selected to encode them, respectively.

For each edge e_{ij}, we use DNA strands $s_{ij} = -h(s_i'' s_j')$ to encode it. The edge linking the vertex v_1 and the node $v_{e13,1}$ are encoded by the following DNA strand.

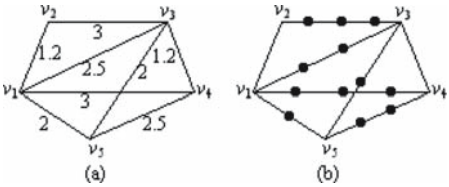

Fig. 13.11. A weighted graph its relative length graph. (a) A weighted graph G and (b) The relative length graph G' of G

$se_{1,e13,1} = -h(s''_1 s'_{e13,1}) = -h(GCATGGTTAC) = GTAACCATGC$

Thus, the DNA strands s_1, $s_{e13,1}$ and $se_{1,e13,1}$ can combine with each other. Based on the property of reverse complementation between the vertex-node codes and the edge codes, the dsDNAs may extend continually to form various random paths including the optimal solution. For more details, please see the reference [12].

DNA Algorithm

Given a weighted graph $G = (V,E)$ with n vertices and m edges, $v_i \in V$, $e_{ij} \in E$, $1 \leq i, j \leq n$, where the weight on edge e_{ij} is w_{ij}, $w_{ij} \geq 0$. Suppose that v_1 is the start vertex just as well. The DNA algorithm [12] for the traveling salesman problem using the RLM encoding method is as follows.

(1) Construct the relative length graph G' of the given graph $G = (V,E)$. For each vertex or node in G', use DNA strand s_i of length $2c$ to encode it. For each edge e_{ij} in G', use the DNA strand $s_{ij} = -h(s''_i s'_j)$ to encode it.

(2) *Merge*: All the DNA strands s_i and s_{ij} are mixed together in a single ligation reaction. Based on the Watson-Crick's law, randomly form various dsDNAs corresponding to the random paths.

(3) $B(N,s_1)$: Separate all the paths beginning with the start vertex v_1, or separate all the DNA molecules with $5'$ end being s_1.

(4) $E(N,s_1)$: Separate all the paths ending with the destination vertex v_1, or separate all the DNA molecules with $3'$ end being s_1.

(5) $+(N,s_i)$: For each vertex v_i ($2 \leq i \leq n$), separate all the paths including v_i.

(6) Separate the shortest path by means of gel electrophoresis.

(7) Determinate the nucleotide sequence of the shortest path. Suppose that the nucleotide sequence corresponds to v_1, v_{ei}, v_{ei+1}, ..., v_2, v_{ej}, v_{ej+1}, ..., v_1. Delete the nodes v_{ex} from the vertex sequence, the obtained vertex sequence v_1, v_2, ..., v_1 corresponds to the optimal solution.

The RLM method [12] is an improvement on the previous work [6]. The main improvements are as follows. (1) The lengths of DNA strands used in the RLM method are not proportional to the values of weights, which makes the RLM method can easily encode weights of positive real numbers or integers, even very small or very large number. That is, the weights that can be encoded by the RLM method may be in a very broad range since weights are encoded in DNA strands only according to their order numbers. For example, if the weights are 30, 1.8, 400, their order

numbers are 2, 1, 3, respectively. Thus, the edges are respectively divided into 2, 1, 3 segments, and the DNA strands to encode them are with the lengths of 20, 10, 30, respectively. (2) The RLM method can distinguish the paths with almost same weights, such as 1000 and 1001, because with the help of the relative length graph, the difference between the lengths of DNA strands used to encode paths is always above or equal to $2c$ ($c \geq 5$). For more details, please see the reference [12].

13.4.7 Method of Encoding Nonlinear Solutions

Han *et al* [16] presented a DNA encoding method for the minimum spanning tree problem, an instance of optimization problems on weighted graphs. The minimum spanning tree problem cannot be directly solved based on the molecular biology techniques because the degrees of some vertices in a minimum spanning tree may be above to 2, which cannot be directly represented by linear DNA strands.

The Minimum Spanning Tree Problem

For a connected, weighted, undirected graph $G = (V, E)$, a spanning tree is a tree that contains all vertices of G, the weight of the spanning tree is the sum of the weights on edges in it, and the minimum spanning tree (MST) is a spanning tree with minimum weight. The MST problem is to find a MST for a connected, weighted, undirected graph.

The MST problem is very important because there are many situations in which MST must be found. Whenever one wants to find the cheapest way to connect a set of terminals, such as cities, electrical terminals, computers, or factories, by using roads, wires, or telephone lines, the solution is a MST for the graph with an edge for each possible connection weighted by the cost of that connection. The MST problem has been studied since the fifties, there are many exact algorithms for it. Han *et al* [16] presented a DNA solution to the MST problem.

DNA Encoding Method

In order to clearly describe the DNA encoding method [16] for the MST problem, we first give the definition of recognition code.

Definition 13.4.7.1 For a connected, weighted graph $G = (V, E)$ with n vertices, a DNA strand used to distinguish a vertex from others is called the recognition code of the vertex.

The length l of recognition code should satisfy: $4^{l-1} < n \leq 4^l$, or $l = \lceil log_4 n \rceil$, where 4 stands for the number of letters in the alphabet $\{A, T, G, C\}$.

For a connected, weighted graph $G = (V, E)$, $v_i \in V$, $e_{ij} \in E$, where each weight w_{ij} on edge e_{ij} is an integer, the DNA encoding method proposed by Han *et al* [16] for the MST problem is as follows.

(1) Let $l = max\{\lceil log_4 n \rceil, 6\}$. For each vertex v_i, use DNA strand r_i of length l to encode it. Here, l denotes the length of recognition code, and 6 is an empirical value which indicates the minimum length of recognition code.

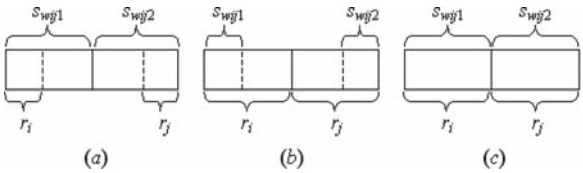

Fig. 13.12. The DNA strand s_{ij} of encoding edge e_{ij}. (a) In the case of w_{ij} being larger than l, (b) In the case of w_{ij} being less than l and (c) In the case of w_{ij} being equal to l

If $n > 4^6$, the recognition codes of length $\lceil log_4 n \rceil$ are needed to distinguish each vertex from others; otherwise, each vertex can be distinguished from others using the recognition codes of length 6. Here, select $l = 6$ instead of $l < 6$ because too short recognition codes would result in high error rate. For example, if the number of vertices in G is 4 and $l = 1$, each vertex can be distinguished from others but in the DNA algorithm, when the recognition codes combine with one part of the DNA strands corresponding to edges, they may combine with another part of the DNA strands because they are too short and easy to be successfully matched to several parts of the DNA strands based on the Watson-Crick's law.

(2) For each edge e_{ij}, the DNA strand s_{ij} of length $2p = 2 \times max\{w_{ij}, l\}$ are used to encode it. Here, the first l letters of s_{ij} are the same as r_i, and the last l letters of s_{ij} are the same as r_j. In addition, the first w_{ij} letters of s_{ij} is marked as s_{wij1}, and the last w_{ij} letters of s_{ij} is marked as s_{wij2}. Note that, when w_{ij} is larger than l, the DNA strand s_{ij} is with a length of $2p = 2 \times w_{ij}$, as shown in Fig. 13.12(a). Here, r_i or r_j should not be the substring of the center part of s_{ij}. When w_{ij} is less than l, the DNA strand s_{ij} is with a length of $2p = 2 \times l$, as shown in Fig. 13.12(b). And when w_{ij} is equal to l, the DNA strand s_{ij} used to encode edge e_{ij} is shown in Fig. 13.12(c).

(3) For any two adjacent edges e_{ij}, e_{jk}, add one DNA strand s_{aijk} as an additional code, which is the reverse complement of s_{wij2} and s_{wjk1}, or $s_{aijk} = -h(s_{wij2}s_{wjk1})$. Obviously, the additional code s_{aijk} is with a length of $w_{ij} + w_{jk}$. Thus, the DNA strands s_{ij} and s_{jk} can combine with the additional code s_{aijk} to form a fragment of dsDNA. Note that, for the edges e_{ij}, e_{ji}, add one DNA strand $s_{aiji} = -h(s_{wij2}s_{wji1})$ as an additional code, which is with a length of $2w_{ij}$.

For more details, please see the reference [16].

DNA Algorithm

The MST problem cannot be directly solved based on molecular biology techniques because some degrees of vertices in a MST may be above to 2, which cannot be directly represented by linear DNA strands. Take the graph G shown in Fig. 13.13(a) as an example, a MST of G is given in Fig. 13.13(b). The degrees of vertices v_2 and v_5 in the MST are 3, which cannot be directly represented by linear DNA strands. In order to generate DNA strands of encoding a MST, each edge in the MST is copied to form an Euler graph G', as shown in Fig. 13.13(c). The Euler cycle in G' can be found out by means of the molecular biology techniques, and the MST can be obtained from the Euler cycle.

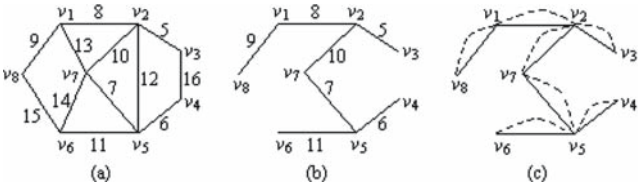

Fig. 13.13. A weighted graph and its minimum spanning tree and Euler graph. (a) A weighted graph G, (b) A minimum spanning tree T of G and (c) The Euler graph G' of T

For a connected, weighted graph $G = (V, E)$, $v_i \in V$, $1 \leq i \leq n$, $e_{ij} \in E$, where each weight w_{ij} on edge e_{ij} is an integer, the DNA algorithm proposed by Han *et al* [16] for the MST problem is as follows.

(1) Let $l = max\{\lceil log_4 n \rceil, 6\}$. For each vertex v_i, use DNA strand r_i of length l to encode it. For each edge e_{ij}, let $p = max\{w_{ij}, l\}$, use DNA strands s_{ij} of length $2p$ to encode it, in which the first l letters are the same as r_i, the last l letters are the same as r_j, and the center part does not include the substrings r_i or r_j. In addition, the first w_{ij} letters of s_{ij} is marked as s_{wij1}, and the last w_{ij} letters of s_{ij} is marked as s_{wij2}.

(2) For any two adjacent edges e_{ij}, e_{jk}, add one DNA strand s_{aijk} as an additional code, which is the reverse complement of s_{wij2} and s_{wjk1}, or $s_{aijk} = -h(s_{wij2}s_{wjk1})$. Note that, for any edge e_{ij}, add one DNA strand $s_{aiji} = -h(s_{wij2}s_{wji1})$ as an additional code.

(3) *Merge*: All the DNA strands s_{ij} and s_{aijk} $(1 \leq i, j \leq n)$ are mixed together. Based on the Watson-Crick's law, generate randomly various part dsDNAs.

(4) *Denature*: All the part dsDNAs are converted into the DNA strands by means of heating.

(6) $-(N, s_i)$: All the DNA strands with any additional code are discarded. Let m denote the number of edges in G, and s_i denote one additional code. All the DNA strands without any additional code can be obtained through the following DNA program: For i:=1 to m do $\{-(N, s_i)\}$. Note that in the DNA encoding method, all the upper DNA strands do not include any additional code.

(7) $+(N, r_i)$: Separate the DNA strands in which the number of recognition codes being at $5'$ end is equal to n, and the number of recognition codes being at $3'$ end is also equal to n. Let cr_i denote the reverse complement of the recognition code r_i, or $cr_i = -h(r_i)$, $1 \leq i \leq n$. All the DNA strands generated in step 6 are mixed with cr_i so as to make the DNA strands combine with cr_i to form part dsDNAs. Thus, all the DNA strands including r_i can be obtained by the operation $+(N, r_i)$. The number of the recognition codes in the DNA algorithm is n, so all the DNA strands with n recognition codes can be obtained through the following DNA program: For i:=1 to n do $\{+(N, r_i)\}$.

(8) Separate the DNA strands with the minimum weight through gel electrophoresis.

(9) Determinate the nucleotide sequence of the DNA strand with minimum weight, which corresponds to the Euler cycle. The minimum spanning tree can be obtained from the Euler cycle.

For more details, please see the reference [16].

13.4.8 DNA Encoding Method for the Maximal Weight Clique Problem

Based on Ouyang's DNA algorithm [3] for the the maximal clique problem, Han *et al* [13] proposed an DNA encoding method for the maximal weight clique problem. For an undirected, weighted graph $G = (V, E)$, $v_i \in V$, $e_{ij} \in E$, where the weight on vertex v_i is w_i, $w_i \geq 0$, the maximal weight clique problem (MWCP) is to find a subset of mutually adjacent vertices, i.e. a clique, which has the largest total weight. Suppose that each weight w_i on vertex v_i is an integer. If one of the weights is a real number, all the weights are multiplied by a certain integer (i.e. 10) and then they are rounded into integers.

DNA Encoding Method

For an undirected, weighted graph $G = (V, E)$ without parallel edges, that is, there is maximum one edge between any two vertices in G, the DNA encoding method proposed by Han *et al* [13] for the maximal weight clique problem is as follows.

(1) For each vertex $v_i \in V$, use two DNA strands s_{i1} and s_{i2} to encode it. The DNA strand s_{i1} consists of three parts: s'_{i1}, s_{wi} and s''_{i1}, where s_{wi} is with a length of w_i, and s'_{i1} or s''_{i1} is with a length of 10, that is, the DNA strand $s_{i1} = s'_i s_{wi}\, s''_i$ is with a length of $20 + w_i$. The strand s_{i2} is the reverse complement of s_{wi}, or $s_{i2} = -h(s_{wi})$. Obviously, s_{i2} can combine with the center part of s_{i1} to form a fragment of dsDNA. After encoding each vertex, the restriction sequences are embedded at both sides of s_{wi} and s_{i2}. The codes of vertex v_i are shown in Fig. 13.14(a).

(2) For each edge $e_{ij} \in E$, use the DNA strand $se_{ij} = -h(s''_{i1} s'_{j1})$ to encode it, which is the reverse complement of the last part of s_{i1} and the first part of s_{j1}. Obviously, the DNA strand se_{ij} is with a length of 20. Thus, the DNA strands corresponding to vertices v_i and v_j can combine with the DNA strand corresponding to edge e_{ij} to form a stable dsDNA, as shown in Fig. 13.14(b).

For more details, please see the reference [13].

DNA Algorithm

For an undirected, weighted graph $G = (V, E)$, $v_i \in V$, $e_{ij} \in E$, where the weight on vertex v_i is w_i, $w_i \geq 0$, let n_c denote the number of edges in the complementary graph G' of G. The DNA algorithm proposed by Han *et al* [13] for MWCP is as follows.

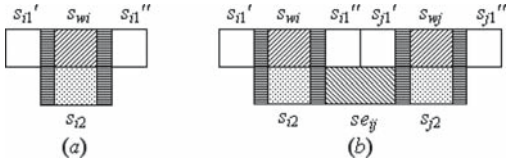

Fig. 13.14. Joint of the vertex codes and the edge code. (a) DNA strands s_{i1} and s_{i2} of encoding vertex v_i and (b) Joint of DNA strands s_{i1}, s_{i2}, s_{j1}, s_{j2} and se_{ij} of encoding vertices v_i, v_j and edge e_{ij}

(1) For each vertex $v_i \in V$, use two DNA strands s_{i1} and s_{i2} to encode it. The DNA strand $s_{i1} = s'_{i1}s_{wi}s''_{i1}$ is with a length of $20 + w_i$, and the strand $s_{i2} = -h(s_{wi})$ is with a length of w_i. For each edge $e_{ij} \in E$, use the DNA strand $se_{ij} = -h(s''_{i1}s'_{j1})$ to encode it.

(2) *Merge*: All the DNA strands s_{i1}, s_{i2} and se_{ij} are mixed together in a single ligation reaction. Based on the Watson-Crick's law, generate randomly the various dsDNAs.

(3) *Amplify*: The dsDNAs starting with v_1 and ending with v_n are amplified through polymerase chain reaction (PCR). Let $sumw = \Sigma_{i=1\to n} w_i$. Only those dsDNAs whose length is equal to or less than *sumw* are saved. The set of the saved dsDNAs is called the complete data pool.

(4) *Digest*: All the dsDNAs in the complete data pool are digested with restriction enzymes. The enzymes break DNA at specific restriction sites, which were embedded within the sequences for v_i ($1 \le i \le n$). See step 2 in the DNA algorithm for MCP in section 13.3.3.

(5) $(N, \le sumw)$: Separate all the dsDNA whose length is less than or equal to *sumw*. By n_c sequential restriction operations with different enzymes, all the DNA fragments connected by the edges in the complementary graph G' are digested. Each time of digesting the DNA fragments, let $sumw = sumw - min\{w_i, w_j\}$.

(7) Separate the longest dsDNAs in the remaining data pool through gel electrophoresis.

(8) Determinate nucleotides sequence of the longest dsDNAs, which corresponds to the optimal solution.

The DNA algorithm for MWCP [13] is an improvement on Ouyang's algorithm for MCP [3]. The main improvements are as follows. (1) On the basis of Ouyang's DNA computing model, Han *et al* [13] add weight representation in DNA strands. (2) In Ouyang's algorithm for MCP, the space complexity is $O(n^n)$, where n denotes the number of vertices in the given graph. In Han's algorithm for MWCP, the space complexity is $O(d_{max}^n)$, where d_{max} denotes the maximum of vertex degrees in the given graph. (3) In Ouyang's algorithm, all the combinations of vertices are in the complete data pool, whereas in Han's algorithm, those vertices disconnected by edge in G are not in the complete data pool. For more details, please see the reference [13].

13.4.9 DNA Encoding Method for the 0/1 Knapsack Problem

Han *et al* [15] presented a DNA encoding method for the 0/1 knapsack problem. Given a set of n items and a knapsack of capacity c, where each item i has a profit p_i and a weight w_i, the 0/1 knapsack problem is to select a subset of the items which satisfies: the sum of weight does not exceed the knapsack capacity c and the sum of profit is maximal.

DNA Encoding Method

For an instance of the 0/1 knapsack problem, let $\mathbf{I} = \{1, 2, \ldots, n\}$, $\mathbf{P} = \{p_1, p_2, \ldots, p_n\}$, $\mathbf{W} = \{w_1, w_2, \ldots, w_n\}$, and the knapsack capacity $c = c_0$. Suppose that each

profit p_i and each weight w_i are integers. If one of the profits is not an integer, all the profits are multiplied by a certain integer (i.e.10) and then they are rounded into integers. If one of the weights is not an integer, all the weights and the knapsack capacity are multiplied by a certain integer and then they are rounded into integers. Also suppose that $w_i \geq p_i$ for all i. If there is any $w_i < p_i$, all of the weights and the capacity are multiplied by a certain integer. The DNA encoding method proposed by Han [15] for 0/1 knapsack problem is as follows.

(1) For each item i ($1 \leq i \leq n$), use two DNA strands s_{i1} and s_{i2} of different length to encode it. The DNA strand $s_{i1} = s'_{i1}s_{pi}s''_{i1}$ is with a length of w_i, where the center part s_{pi} is with a length of p_i, the first part s'_{i1} is with a length of $\lfloor (w_i - p_i)/2 \rfloor$, and the last part s''_{i1} is with a length of $w_i - p_i - \lfloor (w_i - p_i)/2 \rfloor$. The DNA strand s_{i2} is the reverse complement of s_{pi}, or $s_{i2} = -h(s_{pi})$. Thus, s_{i2} can combine with the center part of s_{i1} to form a fragment of dsDNA.

(2) For any two items i and j ($1 \leq i, j \leq n$), add one DNA strand s_{aij} as an additional code, which is the reverse complement of the last part of s_{i1} and the first part of s_{j1}, or $s_{aij} = -h(s''_{i1}s'_{j1})$. Thus, the DNA strands of encoding items i and j can combine with the additional code s_{aij} to form dsDNA.

For more details, please see the reference [15].

DNA Algorithm

Based on the DNA encoding method, the DNA algorithm proposed by Han [15] for the 0/1 knapsack problem is as follows.

(1) For each item i ($1 \leq i \leq n$), use the DNA strands $s_{i1} = s'_{i1}s_{pi}s''_{i1}$ and $s_{i2} = -h(s_{pi})$ to encode it. For any two items i and j ($1 \leq i, j \leq n$), add one DNA strand $s_{aij} = -h(s''_{i1}s'_{j1})$ as an additional code.

(2) *Merge*: All the DNA strands s_{i1}, s_{i2} and s_{aij} ($1 \leq i, j \leq n$) are mixed together. Based on the Watson-Crick's law, generate randomly various dsDNAs.

(3) $(N, \leq c)$: All the dsDNAs whose length is above to the knapsack capacity c are discarded.

(4) *Denature*: The remaining dsDNAs are converted into the DNA strands by means of heating.

(5) $+(N, h(s_{aij}))$: All the DNA strands without any additional code are discarded. Let w denote the reverse complement of an additional code, or $w = -h(s_{aij})$, $w \in \{A, C, G, T\}^*$, $1 \leq i, j \leq n$. All the DNA strands generated in step 4 are mixed with w so as to make the DNA strands combine with w to form part dsDNAs. Thus, the DNA strands without any additional code can be separated away. That is, by means of the operation $+(N, w)$, all the DNA strands without any additional code can be separated away. Note that in the encoding method, all the upper DNA strands do not include any additional code.

(6) Delete the additional codes from the remaining DNA strands. All the DNA strands with additional code are put in a tube. Let $s = uyv$ denote one of the DNA strands with additional codes, where u, y, v represent a fragment of one DNA strand, respectively. Put the DNA strands $-h(u)$ and $-h(v)$ into the tube. After annealing,

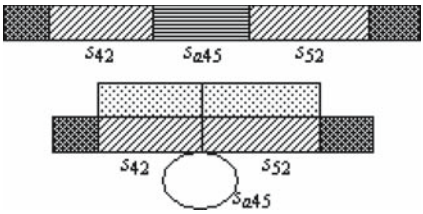

Fig. 13.15. Deletion of the Additional Codes

the strand u combine with $-h(u)$, the strand v combine with $-h(v)$, and fold y, as shown in Fig. 13.15. And then the restriction enzymes are put into the tube to delete y.

(7) Separate the DNA strands with the maximum profit by means of gel electrophoresis.

(8) Determinate the nucleotides sequence of the DNA strand with the maximum profit, which corresponds to the optimal solution.

The DNA algorithm [15] for the 0/1 knapsack problem has the following characteristics: (1) The length of DNA strand s_{i1} which is used to encode item i are equal to the weight w_i, and its center part s_{pi} is with a length of the profit p_i. Thus, the length of the dsDNAs generated in the DNA algorithm is equal to the sum of the weights. By means of the operation $(N, \leq c)$, all the dsDNAs whose length is above to the knapsack capacity c can be discarded. (2) It uses one additional code to link the DNA strands of encoding two items, and the DNA strands s_{i2} and s_{j2} are still linked to the additional code after the dsDNAs are denatured. Since the additional codes can be deleted from the strand s_{i2} $(1 \leq i \leq n)$ by means of the deletion operation and the length of the remaining fragment of s_{i2} is equal to the sum of the profits, so the fragment with the maximum profit can be separated by means of gel electrophoresis which corresponds to the optimal solution. For more details, please see the reference [15].

13.5 Conlusion

Bioinformatics studies the biological information by means of mathematics, computer science and biological techniques. The results of these researches provide a probability of computing with DNA molecules. As an applied branch of the thriving multidisciplinary research area of Bioinformatics, DNA computing has characteristics of higher parallelism and lower costs. Based on the massive parallelism of DNA computing, many researchers tried to solve a large number of difficult problems. These researches demonstrate how DNA can be used for representing information and solving the computational problems and enrich the theories related to DNA computing, in which the methods of representing weights in DNA strands are one of the most important but also challenging issues in DNA computing. Some methods of encoding weights in DNA strands are given in this chapter, which will benefit the further researches on DNA computing, and the rapid development of Bioinformatics will certainly improve the capabilities of DNA computing.

References

1. Adleman L M (1994) Molecular Computation of Solutions to Combinatorial problems. Science 266:1021–1024
2. Lipton R J (1995) DNA solution of hard computational problems. Science 268:542–545
3. Ouyang Q, Kaplan P D, Liu S, et al (1997) DNA solution of the maximal clique problem. Science 278:446–449
4. Head T, Rozenberg G, Bladergroen R S, et al (2000) Computing with DNA by operating on plasmids. Biosystems 57:87–93
5. Sakamoto K, Gouzu H, Komiya K, et al (2000) Molecular computation by DNA hairpin formation. Science 288:1223–1226
6. Narayanan A, Zorbalas S, et al (1998) DNA algorithms for computing shortest paths. In: Proceedings of the Genetic Programming, Morgan Kaufmann 718–723
7. Shin S Y, Zhang B T, Jun S S, et al (1999) Solving traveling salesman problems using molecular programming. In: Proceedings of the Congress on Evolutionary Computation. IEEE Press 994–1000
8. Yamamoto M, Matsuura N, Shiba T, et al (2002) Solutions of shortest path problems by concentration control. Lecture Notes in Computer Science 2340:203–212
9. Lee J Y, Shin S Y, Park T H, et al (2004) Solving traveling salesman problems with DNA molecules encoding numerical values. BioSystems 78:39–47
10. Han A, Zhu D (2006) DNA Encoding Method of Weight for Chinese Postman Problem. In: Proceedings of 2006 IEEE Congress on Evolutionary Computation. IEEE Press 2696–2701
11. Han A, Zhu D (2007) DNA Computing Model Based on a New Scheme of Encoding Weight for Chinese Postman Problem. Computer Research and Development 44:1053–1062
12. Han A (2006) RLM: A New Method of Encoding Weights in DNA Strands. In: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems. IEEE Press 118–121
13. Han A, Zhu D (2006) A New DNA-Based Approach to Solve the Maximum Weight Clique Problem. Lecture Notes in Computer Science 4115:320–327
14. Han A, Zhu D (2006) A New DNA Encoding Method for Traveling Salesman Problem. Lecture Notes in Computer Science 4115:328–335
15. Han A, Zhu D (2006) DNA Computing Model for the Minimum Spanning Tree Problem. In: Proceedings of the 8th International Symposium of Symbolic and Numeric Algorithms for Scientific Computing. IEEE Press 372–377
16. Han A (2006) DNA Computing Model for the 0/1 Knapsack Problem. In: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems. IEEE Press 122–125
17. Paun G, Rozenberg G, Salomaa A (1998) DNA Computing: New Computing Paradigms. Springer, Berlin. Translated by Xu Jin, Wang Shudong, Pan Linqiang (2004) Tsinghua University Press, Beijing
18. Setubal J, Meidanis J (1997) Introduction to Computational Molecular Biology. Cole Publishing Company, Thomson. translated by Zhu H, et al (2003) Science Press, Beijing
19. Zhang B T, Shin S Y (1998) Molecular algorithms for efficient and reliable DNA computing. In: Genetic Programming, Morgan Kaufmann 735–742
20. Xu J, Zhang L (2003) DNA Computer Principle, Advances and Difficulties (I): Biological Computing System and Its Applications to Graph Theory. Journal of Computer Science and Technology 26: 1–10
21. Yin Z (2004) DNA Computing in Graph and Combination Optimization. Science Press, Beijing

22. Wang L, Lin Y, Li Z (2005) DNA Computation for a Category of Special Integer Planning Problem. Computer Research and Development 42:1431–1437
23. Chen Z, Li X, Wang L, et al (2005) A Surface-Based DNA Algorithm for the Perfect Matching Problem. Computer Research and Development 42:1241–1246
24. Braich R S, Chelyapov N, Johnson C, et al (2002) Solution of a 20-variable 3-SAT problem on a DNA computer. Science 296:499–502
25. Lancia G (2004) Integer Programming Models for Computional Biology Problems. Journal of Computer Science and Technology 19:60–77
26. Ibrahim Z, Tsuboi Y, Muhammad M S, et al (2005) DNA implementation of k-shortest paths computation. In: Proceedings of IEEE Congress on Evolutionary Computation. IEEE press 707–713
27. Jonoska N, Kari S A, Saito M (1998) Graph structures in DNA computing. In: Computing with Bio-Molecules–Theory and Experiments. Penn State 93–110