1998 International Computer Symposium
Workshop on Algorithms
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

DNA SOLUTION TO THE TRAVELING SALESMAN OPTIMIZATION PROBLEM

Jing-Shang Hwu and Rong-Jaye Chen

Department of Computer Science and Information Engineering,
National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
Email: {jshwu,rjchen}@csie.nctu.edu.tw

ABSTRACT

In 1994, Leonard M. Adleman firstly used biological experiments with DNA strands to solve the directed Hamiltonian path problem, which is considered to be intractable because of its NP-completeness. In recent work, the use of DNA molecules to solve hard computational problems has been demonstrated. However, those DNA algorithms involved are meant to solve decision problems. In this paper, we propose new molecular solution to a main NP-hard optimization problem: the traveling salesman optimization problem.

1. INTRODUCTION

DNA computation is a new type of computation that employs molecule manipulation to solve computational problems. Its roots dates back to 1959, when Richard Feynmam first introduced the visionary idea of computing at a molecular level. In 1994, Leonard M. Adleman [1] used biological experiments with DNA strands to solve the directed Hamiltonian path problem, which is considered to be intractable because of its NP-completeness. The main idea of DNA computation is that data can be encoded in DNA strands, and molecular biology techniques can be used to execute computational operations. Besides the novelty of the approach, DNA computer has the potential to supply massive parallel computations that outperform electronic computers. There are 6×10^{23} molecules in one mole solution and each bio-operation is performed not on one single DNA strand, but on every strand in the test tube simultaneously. The real power of DNA computers lies in their inherent massive parallelism that electronic computers cannot possibly reach.

Adleman's article caused a quantity of further research work in various angles and developments of molecular computing, such as algorithm design [5,6,8], computation theory [9] and error control [2,7] etc. Here we are interested in designing DNA algorithms. Several algorithms were presented in previous research [5,6,8]. However they were usually designed for solving decision problems. In this paper, we present a DNA algorithm for solving a well-known NP-hard optimization problem: the traveling-salesman problem (TSP).

The rest of this paper is organized as follows. In Section 2, the basic concept of biochemistry is introduced. In Section 3, we introduce the Gupta's encoding method [4] in arithmetic operations. In Section 4, we propose our encoding scheme for solving the optimization problem: TSP. And the detailed steps of the DNA algorithm are also described. Finally, the conclusion is given in Section 5.

2. BIOLOGICAL BACKGROUND

DNA (deoxyribonucleic acid) is the most important molecule in living cells and contains all of the information that specifies cellular properties. In 1953, James Watson and Francis Crick suggested that two helical strands are present in DNA and showed that the two strands are coiled about one another to form a double-stranded *helix*. According to Watson-Crick model, each DNA strand can be viewed as a chain of *nucleotides* attached to a sugar-phosphate backbone. The four kinds of nucleotides that compose a DNA strand are: *Adenine* (A), *Guanine* (G), *Cytosine* (C) and *Thymine* (T); they are often called *bases*. The structure of DNA is organized in particular bond of two DNA strands to form a double-stranded helix. This bond follows a complementary property: A bonds with T and G bonds with C. This is known as Watson-Crick complementarity and denoted as: $\overline{A} = T$, $\overline{T} = A$, $\overline{G} = C$, $\overline{C} = G$.

Each DNA strand has two different ends: the 3' end and the 5' end, which determine its direction. A strand will only bond to its complement if they have opposite polarities. That is, if a strand is denoted AGTC from the 5' end to the 3' end, the other strand must be denoted TCAG from the 3' end to the 5' end. Figure 1 shows the detailed structure of double-stranded DNA.

DNA computation applies a sequence of biological operations to solve computational problems. We now describe the operations that are available on DNA strands.

1. **Synthesis** — A desired strand of DNA can be synthesized in laboratory. They also can be easily produced in large quantities.
2. **Melting and annealing** — We can separate two complementary base pairs of DNA strand by heating the

solution or applying special enzyme; such procedure we call melting. Annealing is the reverse procedure of melting allowing complementary strands to bond together.

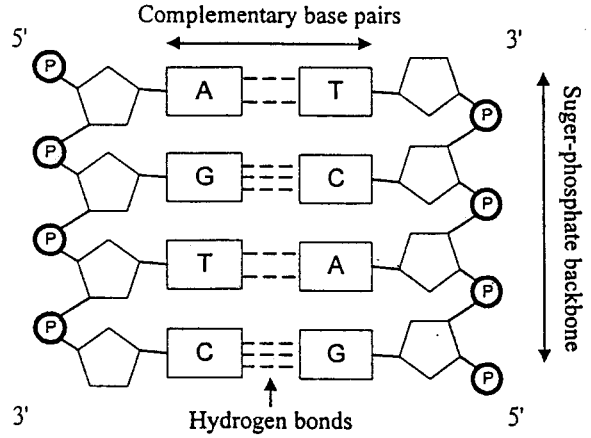


Figure 1: The structure of double-stranded DNA

3. **Ligation** — When several nucleotides are combined to form the double-stranded DNA according to the Watson-Crick complementarity, there are breaks in the sugar-phosphate backbone of each single-strand in duplex DNA. A specific enzyme called *ligase* catalyzes the ligation that links two adjacent bases separated by a break in one strand of a double-stranded DNA.

4. **Primer extension and PCR** — Given a short *primer* oligonucleotide p and a longer *template* oligonucleotide t in the solution, the polymerase enzyme extends p if and only if p is bound to t. The extension must follow the 3' to 5' direction. Polymerase Chain Reaction (PCR) is a process that quickly amplifies the amount of DNA in a given solution.

5. **Affinity purification** — Affinity purification performs the extraction of any single strand containing a specific short sequence from a test tube.

6. **Restriction enzymes** — Restriction enzymes recognize a specific sequence of DNA strand, known as restriction site. Any double-stranded DNA that contains the restriction site within its sequence will be cut by corresponding restriction enzyme at that location.

7. **Gel electrophoresis** — Gel electrophoresis is an important technique for sorting DNA strands by weight, or size.

3. ARITHMETIC OPERATIONS WITH DNA

In order to design algorithms of DNA molecules for optimization problems, the implementation of arithmetic operations in molecular level must be considered. Gupta et al. in [4] proposed a fixed bit-encoding scheme, and showed how a sequence of Boolean and arithmetic operations could be executed in a single test tube producing a unique result. Their approach is to encode truth tables for various binary operations in DNA using a three-level scheme. In each binary operation, they encode the first operand in Level 1, the second operand in Level 2 and the output in Level 3.

Given a binary representation, bit 0 and bit 1 are expressed with different *dinucleotides* unit (an oligonucleotide with two nucleotides). They use the natural DNA bases Adenine (A), Thymine (T), *Uracil* (U) and a non-natural base *7-deaza-adenine* (P) for constructing the dinucleotides. The non-natural base 7-deaza-adenine (P) is complementary with U just like A and yet is chemically distinct. By handling the given material, the "First Operand" DNA strand in Level 1 is constructed with dinucleotides 5'-AU representing bit 1, and 5'-UA representing bit 0. In Level 2, the "Second Operand" DNA strand is constructed with dinucleotides 3'-AT and 3'-TP representing bit 1, and 3'-TA and 3'-PT representing bit 0. Table 1 shows the encoding of truth table for various operations.

To realize the execution of operations in DNA, we give an example of 1001 ADD 0101. The first operand 1001 is encoded as a DNA strand 5'-AUUAUAAU (see Table 2). All 16 possible DNA sequences are used to represent the second operand 0101, since there are two alternative dinucleotides for each bit in Level 2 encoding. However, only one of the second-operand strands is complementary to the first-operand strand. 3'-TAATPTTP is the only one that can hybridize with 5'-AUUAUAAU to yield a unique output duplex representing the correct answer 1110.

This framework can be extended to handle a series of operations. First, a DNA sequence tag is used to ensure that the operations take place only in the desired order. Furthermore, we attach the corresponding output value to the second-operand strands. The corresponding output is constructed using Level 1 encoding so that they can play the role of the first-operand strand in the next operation. A detailed description of encoding for 1001 ADD 0101 is

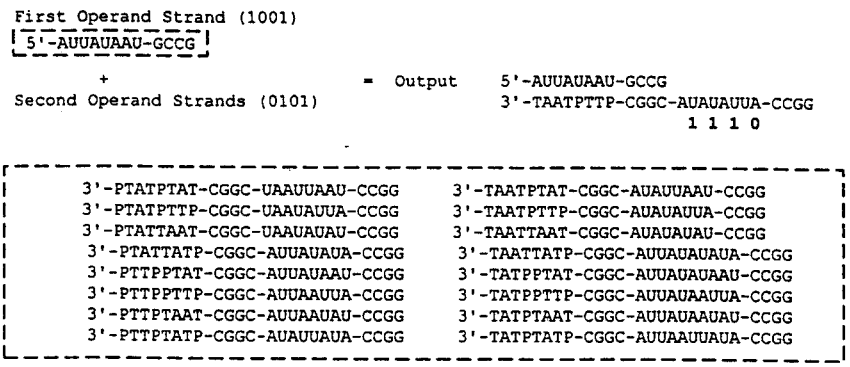
Level 1	Level 2	Level 3 Output			
First Operand	Second Operand	NAND	AND	XOR	ADD
UA = 0	PT = 0	1	0	0	0
	AT = 1	1	0	1	1
AU = 1	TA = 0	1	0	1	1
	TP = 1	0	1	0	10

Table 1: The truth table encoding

First Operand Strand	Second Operand Strand		Unique Output
1001	0101		1110
5′-AUUAUAAU	3′-PTATPTAT	3′-TAATPTAT	5′-AUUAUAAU
	3′-PTATPTTP	**3′-TAATPTTP**	**3′-TAATPTTP**
	3′-PTATTAAT	3′-TAATTAAT	
	3′-PTATTATP	3′-TAATTATP	
	3′-PTTPPTAT	3′-TATPPTAT	
	3′-PTTPPTTP	3′-TATPPTTP	
	3′-PTTPTAAT	3′-TATPTAAT	
	3′-PTTPTATP	3′-TATPTATP	

Table 2: Example of an ADD operation

(a) 1001 + 0101 = 1110



(b) ((1001 + 0101) + 0001) + 0001 = 10000

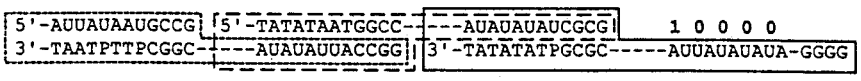


Figure 2: Example of ADD operation

shown in Figure 2(a). And Figure 2(b) illustrates the architecture of performing a series of operations with DNA strands.

In the following section, we apply the encoding method proposed by Gupta et al. to design a DNA algorithm for solving the traveling-salesman problem.

4. THE TRAVELING SALESMAN PROBLEM

Traveling Salesman Problem (TSP): Given a complete graph G = (V, E) with weights on the edges in E, and a specific vertex V_{start}. A traveling-salesman tour is a Hamiltonian cycle that begins with V_{start}, visits each vertex in V exactly once and returns to V_{start}. The problem is to determine the minimum weight of the traveling-salesman tours and is NP-hard.

4.1 Encoding scheme

Following the basic idea of Adleman's experiment [1], we introduce a new encoding method which not only generates all possible paths in the graph, but also executes automatically the addition for weights of edges within the path. A brute-forced algorithm is used to produce possible answers (in DNA strand) with sum attached to the end of DNA strand.

Consider a weighted complete graph G with five vertices shown in Figure 3. Because the graph G is complete, there must exist Hamiltonian cycles. We can choose any vertex in V as the specific vertex V_{start}; here we assume that vertex A is chosen as V_{start}. Each undirected edge in the graph G indicates two directed edges, and we have to encode them respectively. For example, an undirected edge $\overline{AB}$ denotes directed edges $\overrightarrow{AB}$ and $\overrightarrow{BA}$. Given a

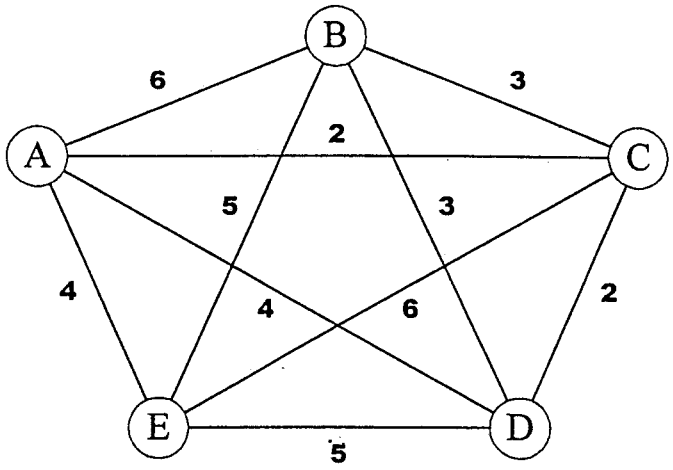


Figure 3: A weighted complete graph with 5 vertices

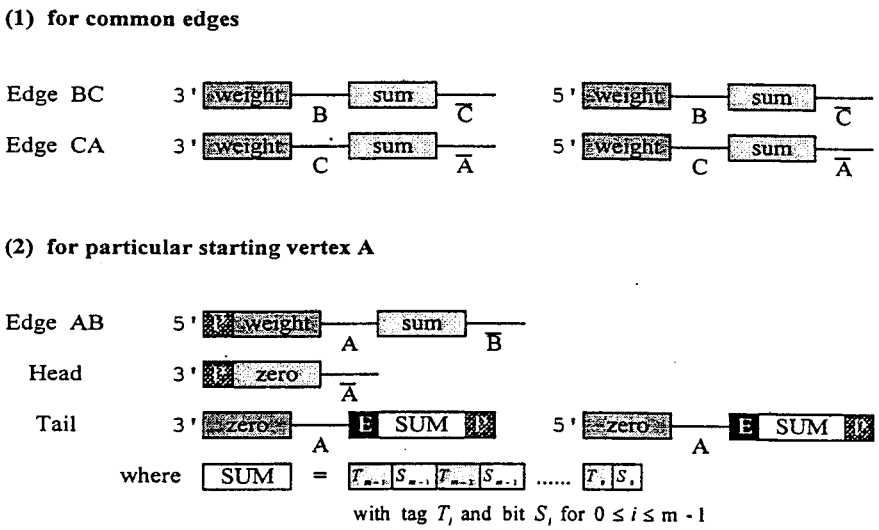


Figure 4: Encoding for edges and particular starting vertex

directed edge $\overline{AB}$, the vertex A is called the *initial* vertex, and the vertex B is called the *terminal* vertex.

The edges in G will be encoded into two categories (as shown in Figure 4):

(1) for common edges

Each vertex in the graph G is associated with an oligonucleotide (a short sequence of DNA) of fixed length *l*. The oligonucleotide that represents vertex A is denoted by *A*. Each directed edge is encoded by a DNA strand with four blocks of oligonucleotides in turn:

(i) ***Weight block***: represents the weight of the edge in binary and is constructed using Level 2 encoding in Section 3

(ii) ***Initial vertex block***: made by the oligonucleotide that represents initial vertex

(iii) ***Sum block***: represents the corresponding output under the ADD operation and is constructed using Level 1 encoding in Section 3

(iv) ***Terminal vertex block***: made by the complement of oligonucleotide that represents terminal vertex

Each DNA strand constructed using the above encoding method must be synthesized in two different directions, from 5' to 3' and from 3' to 5'. The reason why two directions are needed will be shown in Section 4.2.

(2) for particular vertex V_{start}

Here we introduce a new oligonucleotide named *P*, which will be considered as a primer in the reaction of PCR. Three kinds of DNA strand will be produced for particular vertex V_{start} in the stage. First, for the edges starting in V_{start}, we create a DNA strand from 5'

to 3' beginning with $\overline{P}$ (the complement of P), followed by the oligonucleotide of the edge which is synthesized similar to (1). Another two special DNA strands constructed in this stage are *Head* and *Tail*. They play the roles as they are named in the generated paths. The *Head* is synthesized in the 3'-to-5' direction, starting with tag P, followed by a zero weight (using Level 1 encoding) and the complement of V_{start} in turn. And the *Tail* is constructed in both the 3'-to-5' and 5'-to-3' directions with DNA sequence as follows:

$$\boxed{\text{Zero}} - V_{start} - E - \boxed{\text{SUM}} - P, \text{ where}$$

Zero is constructed using Level 2 encoding with value zero

E is the restriction site for some restriction enzyme

SUM represents the weight of the path with binary code $S_{m-1}S_{m-2}\ldots S_0$ and formed as $T_{m-1}S_{m-1}T_{m-2}S_{m-2}\ldots\ldots T_0S_0$,
T_i is a special tag which indicates the significance of the followed S_i

After encoding the edges into different DNA strands and producing two specific ones called *Head* and *Tail*, we can then perform our algorithm to solve the TSP.

4.2 Algorithm

Traveling-Salesman-Problem DNA Algorithm

1. Generate all possible paths through G and compute the weight of paths.
2. Keep only those paths that begin with V_{start} and end with V_{start}.
3. Keep only those paths that enter exactly n vertices.
4. Keep only those paths that enter all of the vertices of G at least once.
5. Compare the weight of the satisfied paths and find out the minimum.

The flowchart of DNA algorithm for TSP is shown in Figure 5.

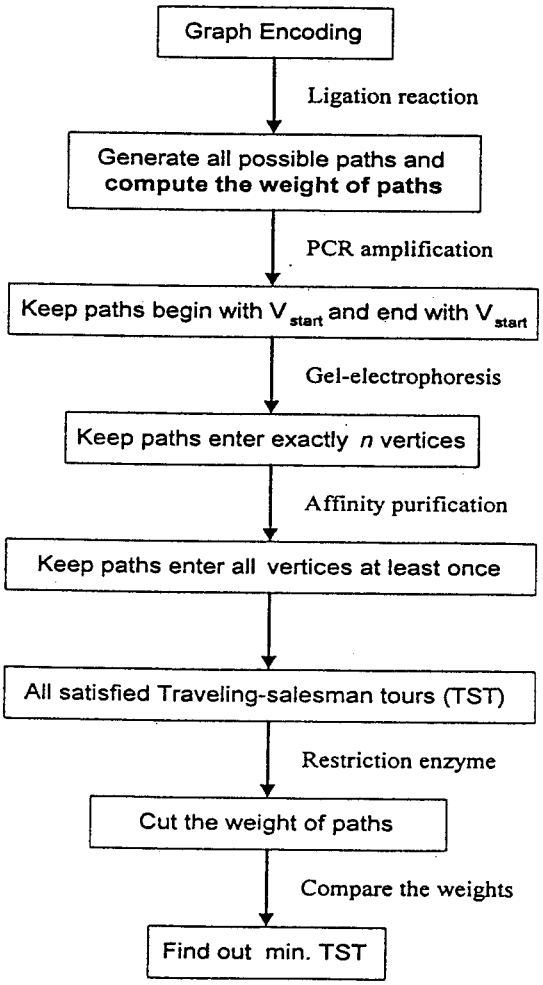


Figure 5: Flowchart of DNA algorithm for Traveling Salesman Problem

Step 1: *Path generation*

Mix the DNA strands synthesized by the encoding scheme mentioned in Section 4.1 into a single test tube. These DNA strands will hybridize with their suitable complementary. After applying a ligation reaction by using *ligase* (an enzyme performs ligation), the double-stranded DNA is formed to represent the corresponding path. And a result of the summation will be attached to the duplex DNA, which we call the sticky-end. The sticky-end may appear in different strands of the double-stranded DNA according to that the number of edge within the path is even or odd (as shown in Figure 6). That is the reason why we must encode some DNA sequences in both two directions, 3'-to-5' and 5'-to-3'.

Step 2: *Amplify the paths begin and end with* V_{start}

In order to increase the concentrations of the wanted DNA strands, PCR is applied to massively amplify the DNA strands. First we heat the solution to break the double-stranded DNA into single strands (melting). Then primers $\overline{P}$ are added into the test tube, annealing with its complement and amplifying the expected strands. Figure 7 illustrates the amplified strands in Figure 6 after applying PCR. In this reaction of PCR, an enzyme called *polymerase* is needed.

Step 3: *Extract the particular length of DNA strands*

Given a graph G with n vertices, the number of edges in the Hamiltonian cycle is n. Under our encoding scheme, the expected length of the DNA strand represents a traveling-salesman tour should be $(n+1)\,d/2 + p + q$,

Where d is the length of encoded DNA for common edge (Figure 4(1))

p is the length of primer P

q is the length of DNA with the form $E - \boxed{\text{SUM}} - P$

With a technique called gel electrophoresis, the DNA strands of such length can be selected.

Step 4: *Extract desired paths by a series of affinity purification*

Affinity purification is applied to extract the strands with specific subsequence. To keep the paths that visit each vertex at least once, the strands with all the vertex's oligo-

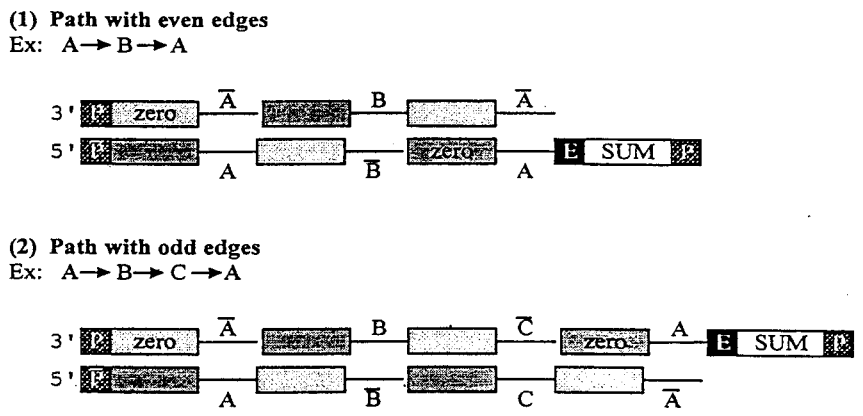


Figure 6: Path generation

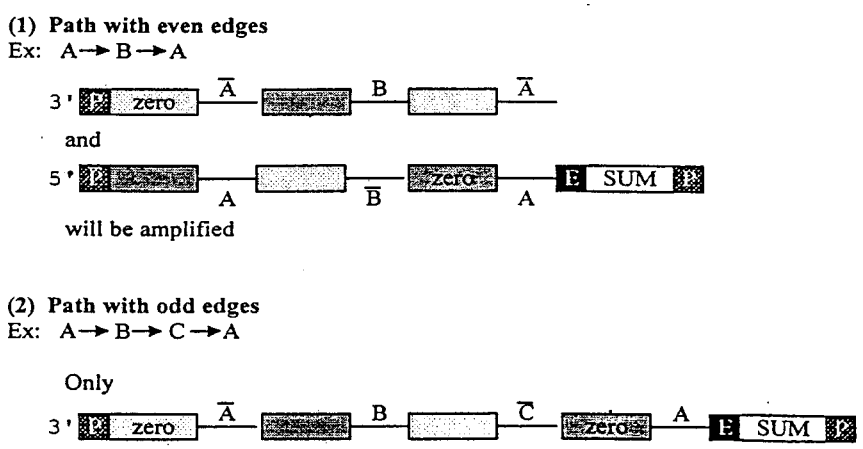


Figure 7: The result of adding primer $\overline{P}$

nucleotides as its subsequence should be left in the test tube. Since the oligonucleotide of vertex V may appear in the form of V or $\overline{V}$ within the DNA strands, the strands containing V or $\overline{V}$ must be both selected. The purpose can be achieved by applying both V and $\overline{V}$ in the affinity purification. Therefore, given a graph with n vertices, there are n extractions needed.

Step 5: *Result comparison*

After step 4, the test tube consists of all the DNA strands that denote satisfied tours with a sticky-end summation. To compare the results, we cut the sticky-end that represents the weight of the path by adding a specific restriction enzyme. The enzyme recognizes the restriction site E and cuts the location, and then the results are extracted and collected in a test tube T. Following a simple method in comparing two integers with binary representation, we run a procedure Min(T) in this stage to find out the minimum weight of all traveling-salesman tours.

Definition:
Given a DNA strand formed as $T_{m-1}S_{m-1}...T_0S_0$, where $S_i \in \{S^1, S^0\}$, S^1 and S^0 denote the binary bit 1 and 0 respectively. $T_i, S^1, S^0 \in \{A, G, T, C\}^*$. Define T_i^1 (or T_i^0) as a set containing all DNA strands with subsequence T_iS^1 (or T_iS^0),

Procedure Min(T)

begin

  for k := m-1 **to** 0 **do**

  begin

    if T≠ϕ **then** separate T into T_k^1 and T_k^0

    else exit and reject;

    if $T_k^0 \neq \phi$ **then** $B_k := 0$; T := T_k^0;

    else $B_k := 1$; T := T_k^1;

  end;

  return $B_{m-1}B_{m-2}...B_0$ in binary;

end

4.3 Discussion

Since our encoding scheme is applied for directed edges, it is easily extended for TSP with incomplete digraph, which is a hard problem with two phases of NP-complete in conventional computer. First phase is the Hamiltonian cycle problem, and the second is TSP. Although the encoding stage needs an exponential time, we omit it since the strands constructed once can be stored in the gene bank and reused for the next time. However, our approach is limited by the quantity of encoded DNA strands. Under our scheme, the edge with its weight represented by a length m binary number MUST be encoded in 2×2^m different DNA strands (twice since two directions are regarded). Each DNA strand also needs munificent copies according to the given size n. If the amount of strands is too large to be handled in laboratory, the problem is still unsolvable in our algorithm. Thus, the number of n and m bound the utilization of our algorithm.

5. CONCLUSION

In this paper, we proposed a linear-time DNA algorithm solving the traveling-salesman problem. The kernel of our algorithms is the encoding scheme, which combines the arithmetic operations and solution generation together. Although our algorithms have never been implemented in the practical laboratory, they work in theory. However, as we discuss in Section 4.3, an improved algorithm is needed to relax the boundary. The improvement can be done in two ways. The first is to reform the encoding scheme. And the second is to discover new bio-technique suitable for our algorithm.

6. REFERENCES

[1] L. M. Adleman, "Molecular Computation of Solutions to Combinatorial Problems," Science 266:1021-1024, December 1994.

[2] M. Amos, A. Gibbons, and D. Hodgson, "Error-resistant Implementation of DNA Computations," In Proceedings of the Second Annual Meeting on DNA based Computers, 1996.

[3] David Freifelder, Molecular Biology, Jones and Bartlett, 1987.

[4] V. Gupta, S. Parthasarathy, and M. J. Zaki, "Arithmetic and Logic Operations with DNA," In Proceedings of the Third Annual Meeting on DNA based Computers, 1997.

[5] L. Kari, G. Gloor, and S. Yu, "Using DNA to Solve the Bounded Post Correspondence Problem," Technical Report, University of Western Ontario, 1997.

[6] L. Kari, G. Gloor, and S. Yu, "Towards a Solution for the Shortest Common Superstring Problem," Technical Report, University of Western Ontario, 1997.

[7] R. M. Karp, C. Kenyon, and Orli Waarts, "Error-resilient DNA Computation," In 7th ACM-SIAM Symposium on Discrete Algorithms, pp.458-467, SIAM, 1996.

[8] R. J. Lipton, "DNA Solution of Hard Computational Problems," Science 268:542-545, 1995.

[9] Diana Rooβ and K. W. Wagner, "On the Power of DNA Computing," In Information and Computation Vol. 131, pp. 95-109, 1996.