Research Journal of Biological Sciences 2 (6): 658-662, 2007
ISSN: 1815-8846

A Parallel Algorithm for the Degree-Constrained Minimum Spanning Tree Problem by Using DNA Computing

Majid Darehmiraki and Hasan Mishmast Nehi
Department of Mathematics, University of Sistan and Baluchestan, Zahedan, Iran

Abstract: DNA computing is new research areas in biology science and information science separately. The essential characteristic of it is the massive parallel of obtaining and managing information. It has been evidenced that DNA computing can solve those problems which are currently intractable on event the fastest electronic computers. The degree-constrained minimum spanning tree is an important problem in graph theory and it is an NP-complete problem. In this study, we present a algorithm for solving degree-constrained minimum spanning tree problem based on sticker model in DNA computing. The study finds all spanning trees of given graph and minimum spanning tree of given graph.

Key words: DNA computing, NP-complete, spanning tree, molecular biology

INTRODUCTION

There are 2 reason for using molecular biology to solve computational problems:

- The information density of DNA is much greater than that of silicon: 1 bit can be stored in approximately one cubic nanometer. Others storage media, such as video tapes, can store 1 bit in 10^{12} cubic nanometer.
- Operation on DNA are massively parallel: a test tube of DNA can contain trillions of strands. Each operation on a test tube of DNA is carried out on all strands in the tube in parallel.

Research in this area was started by Adleman (1994) when he surprised the scientific community by using the tools of molecular biology to solve a hard computational problem. Adleman's experiment, solving an instance of the directed Hamiltonian path problem solely by manipulating DNA strand, marked the first instance of a mathematical problem being solved by biological means.

One of the major achievements of computer science in the last two decades is to understand that many important computational search problems are NP-complete, taking minimum spanning tree problem into consideration, thus are unlikely to have efficient algorithm implemented on silicon-based computer science. Adleman solved one 7-vertex instance of the Hamiltonian path problem, a well-known representative of NP-complete problems, the major goal of subsequent research in DNA computing area is to develop new techniques to solve NP-complete problems that cannot be solved by current electronic computers in a reasonable amount of time. NP-complete problems are those problems for which no polynomial-time algorithm whose worst-case run time is $o(n^k)$ for some constant k, where n is the size of the problem.

WHAT IS DNA?

Before delving into the principles of DNA computing, we must have a basic understanding of what DNA actually is.

DNA (Deoxyribonucleic Acid) is a double stranded sequence of 4 nucleotides; the 4 nucleotides That compose a strand of DNA are as follows: Adenine (A), Guanine (G), cytosine (c) and Thymine (T), they are often called bases . The chemical structure of DNA (the famous double-helix) was discovered by James Watson and Francis Crick in 1953. It consists of a particular bond of 2 linear sequences of bases. This bond follows a property of complementary: Adenine bonds with Thymine (A-T) and vice Versa (T-A), Cytosine bonds with Guanine (C-G) and vice versa (C-G). This is known as Watson-Crick complementary. Each DNA strand has 2 different ends that determine its polarity: The 3' end and the 5' end. The double helix is an anti-parallel (2 strands of opposite polarity) bonding of 2 complementary strands. In recent years, many techniques have been developed in order to study and manipulate DNA in a lab, for various biological applications (Amos *et al*., 2002; Yeh *et al*., 2005; Guo *et al*., 2004).

Corresponding Author: Majid Darehmiraki, Department of Mathematics, University of Sistan and Baluchestan, Zahedan, Iran

PROBLEM DESCRIPTION

A spanning tree in a graph G is a minimal subgraph connecting all the vertices of G. If graph G is a weighted graph (i.e., if there is a real number associated with each edge of G), then the weight of a spanning tree T of G is defined as the sum of the weights of all the branches in T. In general, different spanning trees of G will have different weights. Among all the spanning trees of G, one with smallest weight is of practical significance (Fig. 1). There may be several spanning trees with the smallest weight; for instance, in a graph of n vertices in which every edge has unit weight, all spanning trees have a weight of n-1 units. A spanning tree with the smallest weight in a weighted graph is called a Minimal Spanning Tree (MST) (Deo, 2005).

Degree-constrained MST (dc-MST) problem is to find a minimal spanning tree which contain no edges of degree greater than d (d is a constant). In general, the problem may be stated as follow: Given a weighted connected graph G, find a minimal spanning tree T in G such that:

$$d(v_i) \leq d \text{ for every edge } e_{ij} \text{ in } T$$

This problem is NP-complete, because the Hamilton path problem which is NP-complete, is a special case of dc-MST with d = 2 and all edge weights identical (Papadimitrion and Steightz, 2003).

FINDING ALL SPANNING TREES

The goal in this study, is to determine all spanning tree of graph G. To achieve this goal, initially we introduce study by the operations and procedures which has been used in this section cited from (Adleman, 1998):

Polymerases: Polymerases copy information from one molecule into another. For example, DNA polymerase will make a Watson-Crick complementary DNA strand from a DNA template. In fact, DNA polymerase needs a "start signal" to tell it where to begin making the complementary copy. This signal is provided by a primer -a (possibly short) piece of DNA that is annealed to the template by Watson-Crick complementarities. Wherever such a primer-template pair is found, DNA polymerase will begin adding bases to the primer to create a complementary copy of the template.

Ligases: Ligases bind molecules together. For example, DNA ligase will take to strands of DNA in proximity and covalently bond them into a single strand. DNA ligase is used by the cell to repair breaks in DNA strands that occur, for instance, after skin cells are exposed to ultraviolet light.

DNA synthesis: It is now possible to write a DNA sequence on a piece of paper, send it to a commercial synthesis facility and in a few days receive a test tube containing approximately 10^{18} molecules of DNA, all (or the least most) of which have the described sequence. Currently sequences of length approximately 100 can be reliably handled in this manner.

Gel electrophoresis: This is not stolen from the cell. A solution of heterogeneous DNA molecules is placed in one end of a slab a of gel and a current is applied. The negatively charged DNA molecules move toward the anode, with shorter strands moving more quickly than longer ones. Hence, this process separates DNA by length. With special chemicals and ultraviolet light, it is possible to see bands in the gel where the DNA molecules of various lengths have come to rest.

Now, the following algorithm must be used to find all spanning trees of graph G:

1. Generate a set of random paths through the graph G.
2. For each path in the set:
   a. Check if that path passes through exactly n vertices. If not, remove that path from the set.
   b. For each vertex ,check if that path passes through that vertex. If not, remove that path from the set.
3. If the set is empty, then report that there is not a spanning tree. Otherwise, introduce spanning trees.

This is not a perfect algorithm; nevertheless, if the generation of paths is random enough and the resulting set large enough, then there is a high probability that it will give the correct answer.

To simply the discussion here, consider the graph on Fig. 1 ,which contain just 7 vertex linked by 12 edges.

We must begin by assigning a random DNA sequence to any vertex which made of 8 nucleotide

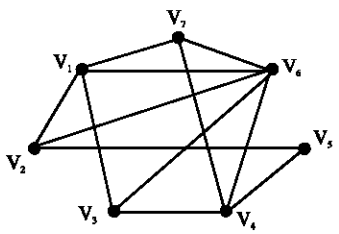

Fig. 1: Seven vertex linked by 12 edges

Table 1: A random DNA sequence to any vertex

Vertices	DNA name	Complement
V_1	ACTTGCAT	TGAACGTA
V_2	TCTAGGCC	AGATCCGG
V_3	GCTATGAC	CGATACTG
V_4	TCGGACTG	AGCCTGAC
V_5	GCAGTACC	CGTCATGG
V_6	GGCTATTC	CCGATAAG
V_7	TCAGTTCG	AGTCAAGC

(Table 1). For show edge's of graph G, we use 2 sequence of nucleotides, for example suppose two end vertices of edge V_1V_2 are vertices 1,2, two sequence defined following: First sequence is constructed from concatenating first 4 nucleotide of vertex 1 and second 4 nucleotide of vertex 2 and 2nd sequence is constructed from concatenating second 4 nucleotide of vertex 1 and first 4 nucleotide of vertex 2.

Recall that each strand of DNA has its Watson-Crick complement. Thus, each vertex has its complementary DNA name. V_1 complementary name becomes, for instance, TGAACGTA.

After working out these encoding, we must synthesize the complementary DNA vertex names and the DNA edge names (As is turned out, the DNA city names themselves were largely unnecessary) we must take a pinch (about 10^{14} molecules) of each of the different sequence and put them into common test tube. To begin the computation, we simply added water-plus ligase, salt and a few other ingredients to approximate the condition inside a cell. Altogether only about one fiftieth of a teaspoon of solution was used. We will achieve the answer in about one second.

To see how, consider what transpires in the tube. For example, the edge V_4V_5 (GCAGTCGG) and the complementary name of V_4 (AGCCTGAC) might meet by chance. By design, the former sequence ends with TCGG and the latter starts with AGCC. Because these sequences are complementary, they will stick together. If the resulting complex now encounters the edge V_4V_6 (ACTGGGCT), it ,too ,will join the complex because the end of the former (TGAC) is complementary to the beginning of the latter (ACTG). In this manner, complexes will grow in length, with DNA name of edges splinted together by complementary DNA vertex names. The ligase in the mixture will then permanently concatenate the chains of DNA name of edges. Hence, the test tube contains molecules that encode random paths through the different vertices.

Notice also that all the paths were created at once by the simultaneous interactions of literally hundreds of trillions of molecules. This biochemical reaction represents enormous parallel processing.

Unfortunately, although we hold the solution in my hand, we also hold about 100 trillion molecules that encoded paths that were not tree. These have to be eliminated.

We must use gel electrophoresis to identify those molecules that have the right length (in the our example, a length of 48). All other molecules must be discarded. This complete step 2a of the algorithm.

To check the remaining sequences for whether their paths passed through all the vertices, we take advantage of Watson-Crick annealing in a procedure called affinity separation. This process uses multiple copies of a DNA "probe" molecule that encodes the complementary name of a particular vertex. These probes are attached to microscopic iron balls, each approximately one micron in diameter.

We suspend the balls in the tube containing the remaining molecules under conditions that encouraged Watson-Crick pairing. Only those molecules that contain the desired vertex's name would anneal to the probes. Then we place a magnet against the wall of the test tube to attract and hold the metal balls to the side while we poured out the liquid phase containing molecules that do not have the desired vertex's name.

Then we must add new solvent and remove the magnet in order to resuspend the balls. Raising the temperature of the mixture cause the molecules to break free from the probes and redissolve in the liquid. Next, we must reapply the magnet to attract the balls again to the side of the test tube, but this time without any molecules attach. The liquid, which now contain the desired DNA strands, could then be pour into a new tube for further screening. The process must be repeated for the remaining vertices. This iterative procedure is the most tedious part of the experiment.

At the conclusion of the affinity separations, step 2b of the algorithm was over and we know that DNA molecules left in the tube should be precisely those encoding spanning trees of given graph. We must use an additional PCR step for identify spanning trees of given graph.

FINDING MINIMUM SPANNING TREES

The goal of this study is to find a spanning tree with the smallest weight among all trees and which contain no edges of degree greater than d (Yeh *et al*., 2005; Guo *et al*., 2004).

Initially we introduce operations which must be used in this study:

Extract (P,s,P^+,P^-): Given a tube P and a short single strand of DNA called s, we can produce 2 tubes P^+ and P^-, where P^+ is all molecules of DNA of P which consist of the short strand s and P^- is all of the molecules of DNA in P which do not contain the short strand s.

Merge (P_0 ,P_1 , . . .,P_n): Given n tubes P_1,P_2,. . .,P_n ,yield P_0, where P_0=P_1 ∪ P_2∪...∪P_n. This operation is to pour n tubes into one, with no change in the individual strands.

Detect (P): Given a tube P , if it includes at least one DNA molecule we can say ' yes' and if it contains no DNA we can say 'no'.

Append (P,s): Given a tube P and a short strand of DNA called s, the operation will append the short strand s onto the end of every strand in tube P.

Discard (P): Given a tube P , the operation will discard the tube P.

Amplify (P_0 ,P_1 , . . . ,P_n): The amplify operation is used to yield n new identical tubes P_1, . . . ,P_n of P_0 and then to empty the tube P_0 .

Set (P,t): Given a tube P. Consider a particular bit position "t". The operation logically turns bit position t "on" (value 1) on every strand in tube P.

Clear (P,t): Given a tube P. Consider a particular bit position "t". The operation logically turns bit position t "off" (value 0) on every strand in tube P.

Number (P): Given a tube P ,count many DNA strands in it.

The algorithm in this study, is in the following manner:

- Step 1: Define strand.
- Step 2: Eliminate infeasible solution.
- Step 3: Compare feasible solution for find minimum spanning tree.

Each step is explained in the following:

Step 1: We designed the data structure in the form of single strand DNA. Each strand is subdivided into nine non-overlapping regions. The first six region encode degree sequence of tree. The 7'th region encode value of b. The 8'th region encodes the summation of degree sequence of tree. Finally, the 9'th region applied for flag bit for addition. The all regions include n bit.

Notice which 2 distinct " value sequences" of 15 nucleotides are assigned to represent values "1" and "0" for each region.

In order to reduce errors in computation ,sequence were designed to discourage intra-and inter-library strand hybridization and unintended probe-library strand hybridization. So, to achieve these goals, sequence were computer generated must satisfy reported constrains in Guo *et al.* (2005).

Step 2: Initially all strands which encode degree sequence of trees is poured into tube T_0. Then by using the below procedure add to them value of d.

The following is defined: If the ith bit in strand h is set to 1, then the corresponding variable X_i in h is denoted X^1_i ;if ith bit in strand h is set to 0, then the corresponding variable X_i in h is denoted X^0_i .

Procedure Init (T_0 , n)

- For i = 1-n
- Append (T_0 ,X^1_{6n+i}) or Append (T_0 , X^0_{6n+i})
- Next i.

In upper procedure depending to value of d, is used whether Append (T_0 , X^1_{6n+i}) or Append (T_0 , X^0_{6n+i}).

The below procedure remove trees which don't satisfy in degree constraint.

Procedure checking (T_0 , n)

- For i = 1-n
- Extract (T_4 , X^1_{k+i} , T_1 , T_3)
- Extract (T_1 , X^0_{6n+i} , T_{drop} , T_4)
- Extract (T_3 , X^1_{6n+i} , T_{ok} , T_4)
- Next i
- Merge (T_0 , T_{ok} , T_4)

In the upper procedure k = 0,n,2n,3n,4n,5n.

Step 3: In this step firstly we initializes bit values by appending DNA strand with a encode value of "0" to the ends of the strands in tube P_0:

Procedure Init-value (P_0 , n)

- For i = 1 – 2n
- Append (P_0 , X^0_{i+7n})
- Next i

Then we compute summation of degree sequence of trees by parallel-add process which passes parameters (k, a, β, f) with (n, 7n, 8n, 9n):

Procedure Parallel-add (P_0 ,n)
For i = 1 - n
For j = 0 – (k-1)

- Extract (T_0 ,$X^1_{(a+k*i)-j}$,T_p ,T_q)
- Extract (T_p ,$X^1_{\beta-j}$,T_a ,T_b)
- Set (T_b , $X_{\beta-j}$)
- Clear (T_a , $X_{\beta-j}$)
- Set (T_a ,$X_{f-(j+1)}$)
- Merge (T_0 ,T_a ,T_b ,T_q)

  For j = 0 – (n+k)
  - Extract (T_0 ,X^1_{f-j} ,T_{f1} ,T_{f0})
  - Extract (T_{f1} , $X^1_{\beta-j}$,T_{A1} ,T_{A0})
  - Set (T_{A0} , $X_{\beta-j}$)
  - Clear (T_{A0} ,X_{f-j})
  - Clear (T_{A1} , $X_{\beta-j}$)
  - Set (T_{A1} ,X_{f-j-1})
  - Clear (T_{A1} ,X_{f-j})
  - Merge (T_0 , T_{A1} ,T_{A0} ,T_{f0})

  Next j

Next i

The final procedure is Parallel-comparator which applied to determine minimum spanning tree. The remained strand (or strands) in tube P_{ans} at the end of the procedure has minimum spanning tree:

Procedure Parallel-comparator (P_0 ,n)

- s= 0
- Do
- s++
- Extract (P_0 ,X^0_{n+1+s} ,P_{ans} ,P_0)
- If number (T_{ans}) = 1 then exit

If number (T_{ans}) = 0 then Amplify (T_0 , T_{ans})

- IF number (P_{ans}) >1 then

  For i = s+1 - n
  - Extract (P_{ans} ,X^0_{n+1+i} ,T_i ,T_{ans})

  If number (T_i) = 1 Then
  - Wash (T_{ans})
  - Amplify (T_i ,T_{ans})
  - Exit

  If number (T_i) > 1 Then
  - Wash (T_{ans})
  - Amplify (T_i ,T_{ans})

  Next i
- Until number (T_{ans}) > 0

CONCLUSION

Because computers have obvious limits in storage , speed, intelligence and miniaturization, the methods of DNA computation have arisen, especially for their efficient parallelism.

In this study, a completely different approach for solving Degree-Constrained MST problem based on DNA computing is proposed which improve upon conventionally adopted exhaustive search. The time complexity of the proposed algorithm is O (n).

REFERENCES

Adleman, L., 1994. Molecular computation of solutions to combinatorial problems. Science, 266: 1021-1024.

Adleman, L., 1998. Computing with DNA. Scientific Am., pp: 54-61.

Amos, M., G. Pãun, G. Rozenberg and A. Saloma, 2002. Topic in the theory of DNA computing. Theoretical Comput. Sci., 287: 3-38.

Deo, N., 2005. Graph theory with applications to engineering and computer science. Prentice-Hall of India private limited new Delhi-110001, pp: 61-63.

Guo, M., W.L. Chang, M. Ho, J. Lu and J. Cao, 2005. Is optimal solution of every NP-complete or NP-hard problem determined from its characteristic for DNA-based computing. Biosystems, 80: 71-82.

Guo, M., W.L. Chang and J. Cao, 2004. Using sticker to solve the 3-dimensional matching problem in molecular super computers. Int. J. High Performance Computing and Networking, 1: 128-139.

Papadimitriou, C.H. and K. Steiglitz, 2003. Combinatorial Optimization Algorithm and Complexity. Prentice-Hall of India private Limited New Delhi-110001, pp: 370-371.

Yeh, C.W., C.P. Chu and K.R. Wu, 2005. Molecular solution to the binary integer programming problem based DNA computing. Biosystems, 83: 56-66.