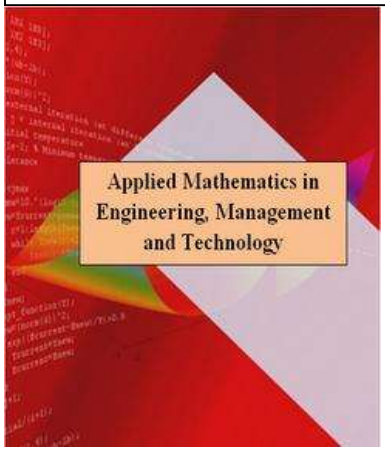
Applied mathematics in Engineering, Management and Technology 3(1) 2015: 527-533
www.amiemt-journal.com

Solving several kinds of constrained shortest path problems via DNA computations

Mahdi Sohrabi-Haghighat[1,*], Ardeshir Dolati[2], Saeed Safaee[1]
[1] Department of Mathematics, Arak University, Arak 38156-8-8349, Iran.
[2] Department of Mathematics, Shahed University,Tehran, Iran.



Abstract

In this paper, we consider some procedures for solving several kinds of constrained shortest path problems in the Adleman-Lipton model. These procedures work (in spite of the NP-completeness of the problems)in polynomial time for an edge-weighted graph in Adleman-Lipton model.

Keywords: constrained shortest path problem, complexity, Adleman-Lipton model, DNA computing.

1 Introduction

Recently, DNA computing has considerable attention as one of non-silicon based computing. Watson-Crick complementarity and massive parallelism are two important features of DNA. By using these features, one can solve an NP-complete problem, which usually needs exponential time on a silicon based computer, in a polynomial time with DNA molecules, e.g. Adleman [1] for Hamiltonian path problem - the first work for DNA computing, Lipton [11] for satisfiability (SAT) problem (the first NP-complete problem), Ouyang et al. [13] for the maximal clique problem, etc. Meanwhile, procedures for primitive operations, such as logic or arithmetic operations, have also been proposed so as to apply DNA computing in a wide range of problems [2-4, 6-8, 19, 20]. However, most of the previous works in DNA computing do not require the consideration of the representation of numerical data in DNA strands. In fact, many practical applications in the real world involve edge-weighted graph problems such as the shortest path problem, the traveling-salesman problem, etc. Therefore, representation of numerical data in DNA strands is an important issue toward expanding the capability of DNA computing to solve the numerical optimization problems. There have been some previous works to represent the numerical data with DNA. Narayanan et al. [12] presented a conceptual encoding method that represents costs with the lengths of DNA strands. Shin et al. [16] proposed a method for representing the real numbers in fixed-length DNA strands by varying the number of hydrogen bonds. Yamamura et al. [20] proposed a concentration control method which encoded the numerical data by means of the concentrations of DNA strands. Lee et al. [9] introduced a novel encoding method that utilizes a temperature gradient to design the sequences so that the DNA strands for higher-cost values have higher melting temperatures than those for lower-cost values [11]. Some other NP-hard problems that have been solved by the model are minimum spanning tree problem [15] and traveling salesman problem [17].
In this paper, we propose some DNA procedures to solve the constrained shortest path problem which imposes k vertices on target path: for an edge-weighted graph find a path starting and ending at the specified vertices such that the total weights on the path is smallest and it passes through k specified or arbitrary vertices.
The rest of this paper is organized as follows. In Section 2, the Adleman-Lipton model is introduced in detail. We introduce some DNA algorithms to solve the several kinds of constrained shortest path problems and their complexities in Section 3. We give conclusions in Section 4.

2 The Adleman-Lipton model

Bio-molecular computers work at the molecular level. Since biological and mathematical operations have some similarities, DNA, the genetic material that encodes the living organisms, is stable and predictable in its reactions and can be used to encode information for mathematical systems.
A DNA (deoxyribonucleic acid) is a polymer which is strung together from monomers called deoxyribo-nucleotides [14]. Distinct nucleotides are detected only with their bases. Those bases are abbreviated as A (adenine), G (guanine), C (cytosine) and T (thymine). Two strands of DNA can form (under appropriate

conditions) a double strand, if the respective bases are the Watson-Crick complements of each other - *A* matches *T* and *C* matches *G*; also 3' end matches 5' end, e.g. the singled strands 5'ACCGGATGTCA3' and 3'TGGCCTACAGT5' can form a double strand. We also call the strand 3'*TGGCCTACAGT*5' as the complementary strand of 5'*ACCGGATGTCA*3' and simply denote 3'*TGGCCTACAGT*5' by $\overline{\text{ACCGGATGTCA}}$. The length of a single stranded DNA is the number of nucleotides comprising the single strand. Thus, if a single stranded DNA includes 20 nucleotides, it is called a 20 mer. The length of a double stranded DNA (where each nucleotide is base paired) is counted in the number of base pairs. Thus, if we make a double stranded DNA from a single stranded 20 mer, then the length of the double stranded DNA is 20 base pairs, also written as 20 bp [11].
The DNA operations proposed by Aldeman and Lipton are described below. These operations will be used to solve the shortest path problem in this paper. A (test) tube is a set of molecules of DNA (i.e., a multi -set of finite strings over the alphabet *A*,*C*,*G*,*T*). Given a tube, one can perform the following operations [11]:
(1)*Merge*(T_1,T_2): for two given test tubes T_1, T_2 it stores the union $T_1 \cup T_2$ in T_1 and leaves T_2 empty;
(2)*Copy*(T_1,T_2): for a given test tube T_1 it produces a test tube T_2 with the same contents as T_1;
(3)*Detect*(*T*): given a test tube *T* it outputs "yes" if *T* contains at least one strand, otherwise, outputs "no"
(4)*Separation*(T_1,*X*,T_2): for a given test tube T_1 and a given set of strings *X* it removes all single strands containing a string in *X* from T_1, and produces a test tube T_2 with the removed strands;
(5)*Selection*(T_1,*L*,T_2): for a given test tube T_1 and a given integer *L* it removes all strands with length *L* from , and produces a test tube T_2 with the removed strands;
(6)*Cleavage*($T,\sigma_0\sigma_1$): for a given test tube *T* and a string of two (specified) symbols $\sigma_0\sigma_1$ it cuts each double strand containing in *T* into two double strands as follows:

$$\left[\frac{\alpha_0\sigma_0\sigma_1\alpha_1}{\alpha_0\sigma_0\sigma_1\alpha_1}\right]\Rightarrow\left[\frac{\alpha_0\sigma_0}{\alpha_0\sigma_0}\right],\left[\frac{\sigma_1\alpha_1}{\sigma_1\alpha_1}\right]$$

(7)*Annealing*(*T*): for a given test tube *T* it produces all feasible double strands in *T*. The produced double strands are still stored in *T* after annealing;
(8)*Denaturation*(*T*): for a given test tube *T* it dissociates each double strand in *T* into two single strands;
(9)*Discard*(*T*): for a given test tube *T* it discards the tube *T*;
(10)*Append*(*T*,*Z*): for a given test tube *T* and a given short DNA singled strand *Z* it appends *Z* onto the end of every strand in the tube *T*.
Since these ten manipulations are implemented with a constant number of biological steps for DNA strands [13], we assume that the complexity of each manipulation is O(1) steps.
(11)*Read*(*T*): for a given tube *T*, the operation is used to describe a single molecule, which is contained in the tube *T*. Even if *T* contains many different molecules each encoding a different set of bases, the operation can give an explicit description of exactly one of them. Since these eleven manipulations are implemented with a constant number of biological steps for DNA strands [14], we assume that the complexity of each manipulation is O(1) steps.

3 DNA algorithms for the shortest path problem passes through *k* vertices

Let *G*=(*V*,*E*) be an edge-weighted graph with the set of vertices $V=\{v_i : i=1,2,...,n\}$ and the set of edges $E=\{e_{i,j} : for\ some\ 1\le i,j\le n\}$. Note that $e_{i,j}$ is in E if and only if the vertices v_i and v_j are connected by an edge. Without loss of generality, we assume that v_1 and v_n are the starting and ending vertices, respectively. Let $|E|=s$. clearly, $s\le n(n-1)$.
In the following, we use the symbols *X*,*Y*,#,B_i ,A_i (*i*=1,2,…*n*), to denote distinct DNA singled strands for which $\|Y\|=\|\#\|=\|A_i\|=\|B_i\|=\frac{1}{2}\|X\|$, where |.| denotes the length of DNA singled strand. For simplicity, we take $\|\#\|=10$ mer.
Obviously the length of the DNA singled strands greatly depends on the size of the problem involved in order to distinguish all above symbols and to avoid hairpin formation [10]. Let $w_{i,j}$ be the weight of edge $e_{i,j}$. The DNA singled strand $Y_{i,j}$ is used to denote the weights on the edges $e_{i,j}$ in *E*, so $\|Y_{i,j}\|=w_{i,j}$. Let $L=\max_{e_{i,j}\in E}\{w_{i,j}\}$. Suppose that all weights in the given graph are commensurable, i.e., there exists a number *y* such that each weight is an integral multiple of *y* (here, take *y*=10) in the following discussion. Let

$$P=\{Y_{i,j},\#A_1B_1,A_nB_n\#,A_iB_i \mid e_{i,j}\in E, i=2,3,...,n-1\},$$
$$Q=\{\overline{A_1},\overline{B_n},\overline{\#},\overline{B_iY_{i,j}A_j} \mid e_{i,j}\in E\},\qquad H=\{X,Y\}.$$

Suppose that $\#A_1B_1$,$A_nB_n\#$ and A_iB_i, (i=2,3,…,n-1) denote the vertices v_i , v_n , and v_i (i=2,3,…,n-1) respectively and k is integer number where $k \leq n$.

3.1 DNA algorithm for the shortest path problem passes exactly through *k* arbitrary vertices

Now, we design the following algorithm to solve the shortest path problem which goes exactly through k arbitrary vertices and give the corresponding DNA operations as follows:

(1) We choose all possible walks which start at v_1 and end at v_n .
(1-1)*Merge*(P,Q)
(1-2)*Annealing*(P)
(1-3)*Denaturation*(P)
(1-4)*Separation* $(P,\#A_1B_1,T_{tmp})$
(1-5)*Discard*(P)
(1-6) *Separation* $(T_{tmp},A_nB_n\#,P)$

After the above six steps of manipulation, the singled strands in tube P will encode all walks which start at v_1 and end at v_n . For example, for the graph in Fig. 1 we have singled strands $\#A_1B_1Y_{1,2}A_2B_2Y_{2,4}A_4B_4Y_{4,6}A_6B_6Y_{6,7}A_7B_7\#$ and $\#A_1B_1Y_{1,2}A_2B_2Y_{2,4}A_4B_4Y_{4,2}A_2B_2Y_{2,5}A_5B_5Y_{5,7}A_7B_7\#$ in P, which denote the walks 1→2→4→6→7and 1→2→4→2→5→7, respectively. Note that some cycles may occur in walks of P, e.g. the latter walk contains a cycle 2→4→2. This operation can be done in O(1) steps.

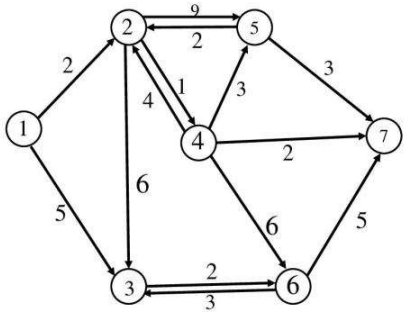


Fig. 1: An edge-weighted graph G with 7 vertices

(2) Each singled strand in P denotes an admissible walk in which the occurrence of the sub-strand A_iB_i with length 20 mer means that the walk passes through vertex v_i . Note that some admissible walk in P may not pass through all vertices of the graph G. For each given admissible walk in P, we have to determine the number of vertices not appearing in the walk to be utilized in later stages. This is done by the following manipulations.

For d=1 *to n*
(2-1) *Separation*(P,A_dB_d,T)
(2-2) *Append*(P,X)
(2-3) *Merge*(P,T)
EndFor

For example, for the graph in Fig.1 the admissible walk 1→2→4→6→7 does not contain vertices 3 and 5. After the above manipulations, its corresponding DNA singled strand becomes $\#A_1B_1Y_{1,2}A_2B_2Y_{2,4}A_4B_4Y_{4,6}A_6B_6Y_{6,7}A_7B_7\#XX$. The two appended single strands XX with total length 40 mer remedy the lack of vertices 3 and 5. This operation can be done in O(n) steps .

(3) Now we choose singled strands in P whose corresponding walks contain exactly k vertices. Such walks therefore do not pass exactly through $n-k$ vertices. So we look for singled strands which contain exactly $n-k$ substrand X as follows.
(3-1) *Separation*(P,$XX...X$,T_1)

(*X* has been repeated *n*-*k*+1 times)
(3-2) *Discard*(T_1)
(3-3) *Separation*(*P*,*XX...X*,T_1)
(*X* has been repeated *n*-*k* times)
(3-4) *Discard*(*P*)
(3-5) *Copy*(T_1,*P*)

Manipulation (3-1) and (3-2) removes all singled strands whose corresponding walks pass through at most *k*-1 vertices and the next steps remains the singled strands whose corresponding walks pass exactly through *k* vertices . This operation can be finished in O(1) steps since each manipulation above works in O(1) steps.
(4)Now, the tube *P* contains DNA strands whose corresponding walks may not be necessary a path. Here we separate the DNA singled strands (if exist) whose corresponding walks are really path. At first, we remember that the number of edges of paths is at most $n-1$ and a walk with *k* vertices is path if the number of its edges is $k-1$ (even if they have been repeated). The following manipulation separates the paths of tube *P*.

For i=1 *to n*
For j=1 *to n*
(4-1) *Separation*(*P*,$Y_{i,j}$,*T*)
(4-2) *Append*(*T*,*Y*)
(4-3) *Merge*(*P*,*T*)
(4-4) *Discard*(*T*)
EndFor
EndFor
(4-5) *Append*(*P*,#)
(4-6) *Separation*(*P*,*XYY...Y#*,*T*)
(*Y* has been repeated *k*-1 times)
(4-7) *Discard*(*P*)
(4-8) *Copy*(*T*,*P*)

Manipulation (4-2) appends a singled strand *Y* to DNA strands of *P* that contain edge $e_{i,j}$ and Manipulation (4-5) appends a singled strand # to all DNA strands of *P*. The next manipulations separate DNA strands with *k*-1 edges (i.e. all paths with *k* vertices). This operation can be done in O(n^2) steps.
(5) We choose the strands of *P* whose corresponding paths are shortest paths. Let $L'=L/10$.

For d=1 *to L'**(*n*-1)
(5-1) *Selection*(*P*,20**n*+30+(*k*-1)*10+10**d*,*T*)
If Detect(*T*) "yes", then break the loop
Else continue the loop
EndFor
Copy(*T*,*P*)

Note that for each vertex of single strand, we use a substrand of length 20 mer. Also each singled strand contains three single strands # and *k*-1 strands *Y*, therefore the length of single strands is at least 20**n*+30+(*k*-1)*10. In the above operation we use a "For" clause. Thus if L' is independent of *n*, then this operation can be done in O(*n*) steps.

(6) Finally the Read operation is applied to give the exact edges in the shortest path problem.
(6−1) *Read*(*P*).

If *P* is empty then the problem of shortest path has no solution. Otherwise, if *P* contains a path it is really the solution of the shortest path problem.
Note that in the above algorithm, if steps (3) and (4) is removed, the algorithm finds out the solutions of the unconstrained shortest path problem. Also prevention of stopping in step (5), determines the *j*-th shortest path of graph.

3.2 DNA algorithm for the shortest path problem passes exactly through *k* specified vertices

We modify the above algorithms to solve the shortest path problem that goes exactly through *k* specified vertices. Assume $v_{i1}(=v_1)$, v_{i2} , v_{i3} ,…, $v_{ik}(=v_n)$ are *k* specified vertices. Step (3) of the algorithm presented in section 3.1

chooses singled strands in P whose corresponding walks contain exactly k vertices. The following manipulations after step (3), chooses strands in P whose corresponding walks contain exactly k specified vertices.

For d=1 *to* k
(3'-1) *Separation*(P,{$A_{id}B_{id}$},T)
(3'-2) *copy*(T,P)
(3'-3) *Discard*(T)
EndFor

This operation can be done in O(n) steps. By continuing the next steps of the algorithm, have been presented in section 3.1, we find the solution of the shortest path problem that goes exactly through k specif ied vertices.

3.3 DNA algorithm for the shortest path problem passes exactly through *k-m* arbitrary vertices and *m* specified vertices where $0 \le m \le k$.

We change the above algorithm to solve the shortest path problem that goes exactly through k-m arbitrary vertices and m specified vertices, where $0 \le m \le k$. Assume that $v_{i1}(=v_1)$, v_{i2} , v_{i3} ,…, $v_{im}(=v_n)$ are m specified vertices. Step (3) of the algorithm presented in section 3.1 chooses singled strands in P whose corresponding walks contain exactly k vertices. By running the following manipulations after step (3), we choose strands in P whose corresponding walks contain exactly m specified vertices (and therefore $k-m$ arbitrary vertices).

For d=1 *to* m
(3"-1) *Separation*(P,{$A_{id}B_{id}$},T)
(3"-2) *copy*(T,P)
(3"-3) *Discard*(T)
EndFor

This operation can be done in O(m) steps. By continuing the next steps of the algorithm, have been presented in section 3.1, we find out the solution of the shortest path problem that goes exactly through m specified and k-m arbitrary vertices.

3.4 DNA algorithm for the shortest path problem passes through *k* specified vertices

We modify the above steps to solve the shortest path problem passes through k specified vertices. Here, there is no assumption about the exact number of vertices of the target path, and we look for a path that its total weight is the smallest and contains k specified vertices. To this end, after steps (1) and (2) of the algorithm presented in section 3.1, we run the step (3) with only manipulations (3-1) and (3-2). Then, step (3') is running. These steps generate all walks which start at v_1 and end at v_n and pass through k specified vertices. Now, the following manipulation is run.

For i=1 *to* n
For j=1 *to* n
(4'-1) *Separation*($P,Y_{i,j},T$)
(4'-2) *Append*(T,Y)
(4'-3) *Merge*(P,T)
(4'-4) *Discard*(T)
EndFor
EndFor
(4'-5) *Append*(P,#)
For d=k *to* n
(4'-6) *Separation*($P,XYY…Y\#,T_1$)
(Y has been repeated d-1 times)
(4'-7) *Merge*(T_2,T_1)
EndFor
(4'-8) *Copy*(T_2,P)

Manipulation (4') separates all paths contain at least k specified vertices. This operations can be finished in O(n^2) steps because each of them can be done in O(1) steps and there is at most $n(n$-1) edges and k is equal to 2 in the worst case. By continuing the next steps (5) and (6) of algorithm, have been presented in section 3.1, we find out the shortest path that goes through k vertices.

In summary, operations costs of each steps (1), (3) and (6) is O(1), and operations costs of each steps (2), (3'), (3") and (5) is O(n), while operations costs of step (4) and (4') is O(n^2). Thus, the constrained shortest path problem for

an edge-weighted graph with n vertices can be figured out in $O(n^2)$ time using DNA molecules. So we have the following theorem.

Theorem 3.1 Let $G=(V,E)$ be an edge-weighted graph with the set of vertices $V=\{v_i : i=1,2,...,n\}$ and the set of edges $E=\{e_{i,j} : for\ some\ 1\le i,j\le n\}$. Suppose that the maximum weight in the graph is independent of n. The algorithms proposed above can give solutions of the constrained shortest path problems in $O(n^2)$ time using DNA molecules.

4 Conclusions

As the first work for DNA computing, Adleman [1] presented an idea to demonstrate that deoxyribonucleic acid (DNA) strands can be applied to solve the Hamiltonian path problem of size n (in spite of its NP-completeness)in $O(n)$ steps using DNA molecules. Adleman's work shows that one can solve an NP-complete problem, which usually needs exponential time on a silicon based computer, in a polynomial number of steps with DNA molecules. From then on, Lipton [11] demonstrated that Adleman's experiment could be used to determine the satisfiability (SAT) problem (the first NP-complete problem). Ouyang et al. [13] showed that restriction enzymes could be used to solve the NP-complete clique problem. In recent years, lots of papers have occurred for designing DNA procedures and algorithms to solve various NP-complete problems. As Guo et al. [5] pointed out, it is still important to design DNA procedures and algorithms for solving various NP-complete problems since it is very difficult to use biological operations for replacing mathematical operations.

Since the constrained shortest path passed trough all vertices of the graph is the shortest Hamiltonian path, therefore the constrained shortest path is an NP-complete problem. In this paper, we propose some procedures to solve the constrained shortest path problem in the Adleman-Lipton model. The procedures work in $O(n^2)$ time for edge-weighted graph with n vertices.

References

[1] L. M. Adleman, Molecular computation of solution to combinatorial problems, Science, 266 (1994) 1021-1024.
[2] P. Frisco, Parallel arithmetic with splicing, Romanian Journal of Information Science and Technology, 2 (2002) 113-128.
[3] A. Fujiwara, K. Matsumoto, Wei Chen, Procedures for logic and arithmetic operations with DNA molecules, International Journal of Foundations of Computer Science, 15 (2004) 461-474.
[4] F. Guarnieri, M. Fliss, C. Bancroft, Making DNA add, Science, 273 (1996) 220-223.
[5] V. Gupta, S. Parthasarathy, M.J. Zaki, Arithmetic and logic operations with DNA, in: Proceedings of 3rd DIMACS Workshop on DNA Based Computers, (1997) 212-220.
[6] H. Hug, R. Schuler, DNA-based parallel computation of simple arithmetic, in: Proceedings of the 7th International Meeting on DNA Based Computers, (2001) 159-166.
[7] S. Kamio, A. Takehara, A. Fujiwara, Procedures for computing the maximum with DNA strands, in: Humid R. Arabnia, Youngsong Mun (Eds.), Proceedings of the International Conference on DNA Based Computers, (2003).
[8] J.-Y. Lee, S.-Y. Shin, T.-H. Park, B.-T. Zhang, Solving traveling salesman problems with DNA molecules encoding numerical values, BioSystems, 78 (2004) 39-47.
[9] D. Li, H. Huang, X. Li, Hairpin formation in DNA computation presents limits for large NP-complete problems, BioSystems, 72 (2003) 203-207.
[10] R. J. Lipton, DNA solution of HARD computational problems, Science, 268 (1995) 542-545.
[11] A. Narayanan, S. Zorbalas, et al., DNA algorithms for computing shortest paths, in: J.R. Koza (Ed.), Proceedings of the Genetic Programming 1998, Morgan Kaufmann (1998) 718-723.
[12] Q. Ouyang, P. D. Kaplan, S. Liu, A. Libchaber, DNA solution of the maximal clique problem, Science, 278 (1997) 446-449.
[13] G. Paun, G. Rozeberg, A. Salomaa, DNA Computing, Springer-Verlag, (1998).
[14] S.-Y. Shin, B.-T. Zhang, S.-S. Jun, et al., Solving traveling salesman problems using molecular programming, in: P.J. Angeline (Ed.), Proceedings of the Congress on Evolutionary Computation 1999, IEEE Press, (1999) 994-1000.
[15] Z. Wang, D. Huang, H. Meng, C. Tang, A new fast algorithm for solving the minimum spanning tree problem based on DNA molecules computation, Biosystems, 114 (2013) 1-7.
[16] Z. Wang , D. Xiao, W. Li, L. He, A DNA procedure for solving the shortest path problem, Applied Mathematics and Computation, 183 (2006) 79-84.
[17] Z. Wang, Y. Zhang, W. Zhou, H. Liu, Solving traveling salesman problem in the Adleman-Lipton model, Applied Mathematics and Computation, 219 (2012) 2267-2270
[18] D. M. Xiao, W. X. Li, Z. Z. Zhang, L. He, Solving maximum cut problem in the Adleman-Lipton model, BioSystems, 82 (2005) 203-207.

[19] D. M. Xiao, W. X. Li, J. Yu, X. D. Zhang, Z. Z. Zhang, L. He, Procedures for a dynamical system on $\{0,1\}^n$ with DNA molecules, BioSystems, 84 (2006) 207-216.
[20] M. Yamamura, Y. Hiroto, T. Matoba, Solutions of shortest path problems by concentration control, Lecture Notes Computer Science, 2340 (2002) 231-240.