

# Molecular solutions for the maximum K-Facility dispersion problem on DNA-based supercomputing

Nozar Safaei <sup>2</sup>, Babak Dalvand <sup>2</sup>, Saeed Safaei <sup>1</sup>, Vahid Safaei

<sup>1</sup>-Department of Mathematics, Arak University, Arak, Markazi, Iran

<sup>2</sup>-Department of Mathematics, Islamic Azad University of Khoramabad, Lorestan, Iran

<sup>3</sup>-Department of Mechanics, Islamic Azad University of Khomini shahr, Esfahan, Iran

**Abstract** - In recent works for high-performance computing, computation with DNA molecules, i.e. DNA computing, has considerable attention as one of non-silicon-based computing. Watson–Crick complementarity and massive parallelism are two important features of DNA. Using the features, 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. In this paper, we consider a procedure for maximum K-Facility dispersion problem in the Adleman–Lipton model. The procedure works in  $O(n^2)$  steps for maximum K-Facility dispersion problem of a directed graph with  $n$  vertices.

**Keywords:** maximum K-Facility dispersion problem; Adleman–Lipton model, NP complete;

## 1 Introduction

In recent works for high-performance computing, computation with DNA molecules, i.e. DNA computing, has considerable attention as one of non-silicon-based computing. Watson–Crick complementarity and massive parallelism are two important features of DNA. Using the features, 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. As the first work for DNA computing, Adleman (1994) presented an idea of solving the Hamiltonian path problem of size  $n$  in  $O(n)$  steps using DNA molecules. Lipton (1995) demonstrated that Adleman’s experiment could be used to determine the NP-complete satisfiability (SAT) problem (the first NP-complete problem). Ouyang et al. (1997) presented a molecule biology-based experimental solution to the maximal clique NP-complete problem. In recent years, lots of papers have occurred for designing DNA procedures and algorithms to solve various NP-complete problems. Moreover, procedures for primitive operations, such as logic or

arithmetic operations, have been also proposed so as to apply DNA computing on a wide range of problems (Frisco, 2002; Fujiwara et al., 2004; Guarnieri et al., 1996; Gupta et al., 1997; Hug and Schuler, 2001; and Kamio et al., 2003).

In this paper, the DNA operations proposed by Adleman (1994) and Lipton (1995) are used for figuring out solutions of maximum K-Facility dispersion problem.

Given a complete directed graph  $G=(V,E)$  with costs on edge satisfying the triangle inequality and an integer  $k$  find a set  $F \subseteq V, |F|=k$  so as to minimize  $\min_{f_1, f_2 \in F} \{d(f_1, f_2)\}$

The rest of this paper is organized as follows. In Section 2, the Adleman–Lipton model is introduced in detail. Section 3 we present a DNA algorithm for solving the maximum K-Facility dispersion problem and the complexity of the proposed algorithm is described. We give conclusions in Section 4.

## 2 The Adleman–Lipton model

Bio-molecular computers work at the molecular level. Because biological and mathematical operations have some similarities, DNA, the genetic material that encodes for 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 (Păun et al., 1998). Distinct nucleotides are detected only with their bases. Those bases are, respectively, 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’-

ACCGGATGTCA-3' and 3'-TGGCCTACAGT-5' 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

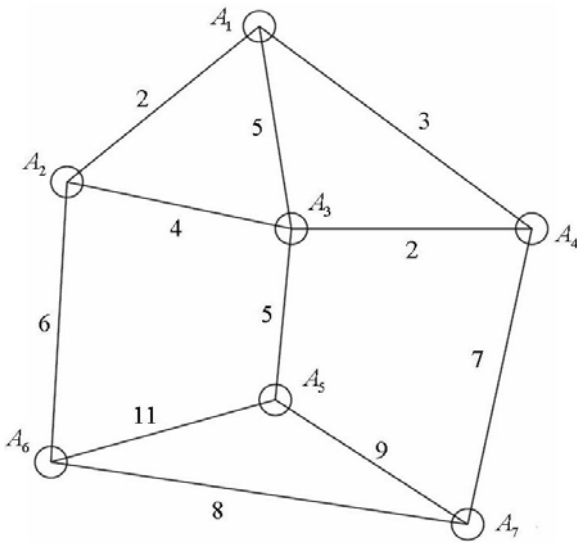


Fig. 1. Graph G.

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.

The Adleman–Lipton model: 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:

- (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  $T_1$ , 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  $\left[ \begin{smallmatrix} \sigma_0\sigma_1 \\ \sigma_0\sigma_1 \end{smallmatrix} \right]$  in  $T$  into two double strands as follows:

$$\left[ \begin{smallmatrix} \alpha_0\sigma_0\sigma_1\beta_0 \\ \alpha_1\sigma_0\sigma_1\beta_1 \end{smallmatrix} \right] \Rightarrow \left[ \begin{smallmatrix} \alpha_0\sigma_0 \\ \alpha_1\sigma_0 \end{smallmatrix} \right], \left[ \begin{smallmatrix} \sigma_1\beta_0 \\ \sigma_1\beta_1 \end{smallmatrix} \right]$$

(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  $T$ ;

(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 (Păun et al., 1998), we assume that the complexity of each manipulation is  $O(1)$  steps.

### 3 DNA algorithm for the maximum $k$ -Facility dispersion problem

Let  $G=(V,E)$  be a directed graph with the set of vertices being  $V=\{A_k \mid k=1,2,\dots,n\}$  and the set of edges being  $E=\{e_{ij} \mid \text{for some } 1 \leq i, j \leq n\}$ . Let  $|E|=d$ . Then  $d \leq n(n-1)$ . Note that  $e_{ij}$  is in  $E$  if the vertices  $A_i$  and  $A_j$  are connected by an edge.

In the following, the symbols  $0,1,\#,X,A_k,B_k$  ( $k=1,2,\dots,n$ ) denote distinct DNA singled strands with same length, say 10-mer. And  $\|\cdot\|$  denotes the length of the DNA singled strand. 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 (Li et al., 2003). The DNA singled strand  $Y_{ij}$  is used to denote the weights on the edges  $e_{ij} \in E$  with  $\|Y_{ij}\| = w_{ij}$  if the corresponding weight equals  $w_{ij}$ . 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.

$Y_{ij}$  single strings represents the distance between vertices  $A_i$  and vertices  $A_j$ . Besides that, the single string  $B_i1A_i$  shows that the  $A_i$  vertices exist in the set whereas  $B_i0A_i$  illustrates that  $A_i$  vertices does not exist in the set.

If  $i = j$  and there is no edge between  $A_i$  and  $A_j$  then the length of the strings representing this edge is considered to be 0 mer. Let

$$P = \{0, 1, X, A_1\#, \#B_n, A_k B_{k-1} \mid k = 2, 3, \dots, n\}$$

$$Q = \{\#, \overline{B_k 1 A_k}, \overline{B_k 0 A_k} \mid k = 2, 3, \dots, n\}$$

$$H = \{Y_{ij} \mid 1 \leq i, j \leq n, i \neq j\}$$

We design the following algorithm to solve the maximum K-Facility dispersion problem and give the corresponding DNA operations as follows:

### 3.1 Produce each possible subset of the set V

For a graph with  $n$  vertices, each possible subset of the set  $V$  of vertices is represented by an  $n$ -digit binary number. A bit set to 1 represents a vertex in the subset, and a bit set to 0 represents a vertex out of the subset. For example, the subset  $(A_6, A_5)$  in Fig. 1 is represented by the binary number 0110000. In this way, we transform all possible subsets of  $V$  in an  $n$ -vertex graph into an ensemble of all  $n$ -digit binary numbers. We call this the data pool.

(1-1) Merge (P, Q);

(1-2) Annealing (P);

(1-3) Denaturation (P);

(1-4) Separation (P,  $\{A_1\#$ },  $T_{tmp}$ );

(1-5) Discard (P);

(1-6) Separation ( $T_{tmp}$ ,  $\{\#B_n\}$ , P);

After above six steps of manipulation, singled strands in tube P will encode all  $2^n$  partitions of  $V$  in the form of  $n$ -digit

ternary numbers. For example, for the graph in Fig. 1 with  $n=7$  we have, e.g. the singled strand  $\#B_7 0 A_7 B_6 1 A_6 B_5 0 A_5 B_4 0 A_4 B_3 1 A_3 B_2 1 A_2 B_1 0 A_1 \#$

which denotes the subset  $\{A_6, A_3, A_2\}$  corresponding to the binary number 0100110. This operation can be finished in  $O(1)$  steps since each manipulation above works in  $O(1)$  steps.

Suppose that all edges cost of Fig.1 satisfying the triangle inequality.

### 3.2 Counting vertices of each subset

At first we want to counting vertices of each subset.

If a vertex exists in the set, the following algorithm will add a string  $X$  to its corresponding string.

For  $d = 1$  to  $d = n$

2-1 separation (P,  $\{B_d 1 A_d\}$ ,  $T_1$ )

2-2 Append ( $T_1$ , X)

2-3 Merge (P,  $T_1$ )

2-4 Discard ( $T_1$ )

End for

Time analysis of the above algorithm

Each of the above actions will conclude at  $O(1)$ . Therefore the algorithm will terminate at  $O(n)$ .

### 3.3 Finding sets of k vertices

Separation of all the strings representing subsets that contain  $k$  vertices. Therefore to find sets of  $k$  vertices, strings that contain  $\underbrace{\{XX \dots XX\}}_{k \text{ times}}$  must be found.

3-1 Separation (P,  $\underbrace{\{XX \dots XX\}}_{k \text{ times}}$ ,  $T_1$ )

3-2 Discard (P)

3-3 Separation ( $T_1$ ,  $\underbrace{\{XX \dots XX\}}_{k+1 \text{ times}}$ ,  $T_2$ )

3-4 Merge (P,  $T_1$ )

Instruction (3-1) will separate all the strings containing  $\underbrace{\{XX \dots XX\}}_{k \text{ times}}$ ,  $\underbrace{\{XX \dots XX\}}_{k+1 \text{ times}}$ ,  $\underbrace{\{XX \dots XX\}}_{k+2 \text{ times}}$ , ... from tube P and will place them in tube  $T_1$ .

Instruction (3-3) will separate all the strings containing  $\{\underbrace{XX\dots XX}_{k+1 \text{ times}}, \underbrace{XX\dots XX}_{k+2 \text{ times}}, \underbrace{XX\dots XX}_{k+3 \text{ times}}, \dots$  from tube  $T_1$  and will place them in tube  $T_2$ .

Hence all the strings containing  $\underbrace{XX\dots XX}_{k \text{ times}}$  will remain in  $T_1$  tube.

Time analysis of the above algorithm:

Each of the above actions will conclude at  $O(1)$ . Therefore the algorithm will terminate at  $O(n)$ .

### 3.4 Step 4

Let the subset  $F$  correspond the  $n$ -digit binary number.  $a_n \dots a_i \dots a_j \dots a_1$  For each pair  $(a_i, a_j)$  with  $a_i = 1, a_j = 0$  or  $a_i = 0, a_j = 1$  we append the singled strand  $y_{i,j}$  or  $y_{j,i}$  to the end of the singled strand which encode the  $n$ -digit binary number  $a_n \dots a_i \dots a_j \dots a_1$ . For example, the singled strands

# $B_7 0A_7 B_6 1A_6 B_5 1A_5 B_4 0A_4 B_3 1A_3 B_2 1A_2 B_1 0A_1$ #  
(representing the binary number 0110110 for the

graph in Fig. 1) is transformed into

# $B_7 0A_7 B_6 1A_6 B_5 1A_5 B_4 0A_4 B_3 1A_3 B_2 1A_2 B_1 0A_1$ #  $y_{6,5} y_{6,2}$   
 $y_{3,2} y_{5,3}$

where the singled strands  $y_{6,3}, y_{5,2}$  do not appear since there are

not corresponding edges in the graph shown in Fig. 1

For  $k = n$  to  $k = 1$

(4 - 1) Separation  $(P, \{B_k 1A_k\}, T_1)$

For  $i = n$  to  $i = 1$

(4 - 2) Separation  $(T_1, \{B_i 1A_i\}, T_2)$

(4 - 3) Append  $(T_2, y_{k,i})$

(4 - 4) Merge  $(T_1, T_2)$

End For

(4 - 5) Merge  $(P, T_1)$

End For

Time analysis of the above algorithm

Each of the above actions will conclude at  $O(1)$ . This algorithm consists of 2 for blocks, therefore the algorithm will terminate at  $O(n^2)$ .

### 3.5 Step 5

In the last stage, the set with the minimum distance must be recognized.

To do this  $Y_{i,j}$  strings must be sorted based on their length.

These strings will be indexed from 1 to  $n^2$ , in a way that the string with the minimum length will be indexed 1 and the string with the shortest length will be indexed  $n^2$ .

For  $d = 1$  to  $n^2$

(5 - 1) Separation  $(P, Y_d, T_1)$

(5 - 2) if detect(T) is yes,

then end for else continue the circulation.

Time analysis of the above algorithm

Each of the above actions will conclude at  $O(1)$ . This algorithm consists of 2 for blocks, therefore the algorithm will terminate at  $O(n^2)$ .

### 3.6 Giving the exact solutions

Finally the Read operation is applied to giving the exact solutions to the maximum K-Facility dispersion.

(6 - 1) Read (T).

## 4 Conclusion

As the first work for DNA computing, (Adleman, 1994) presented an idea to demonstrate that deoxyribonucleic acid (DNA) strands can be applied

to solving the Hamiltonian path NP-complete problem of size  $n$  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 (1995) demonstrated that Adleman's experiment could be used to determine the NP-complete satisfiability (SAT) problem (the first NP-complete problem). Ouyang et al. (1997) 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. (2005) 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.

In this paper, we propose a procedure for maximum K-Facility dispersion NP-complete problems in the Adleman–Lipton model. The procedure works in  $O(n^2)$  steps for maximum K-Facility dispersion problem of a directed graph with  $n$  vertices. All our results in this paper are based on a theoretical model. However, the proposed procedures can be implemented practically since every DNA manipulation used in this model has been already realized in lab level.

## 5 References:

- [1] Adleman, L.M., Molecular computation of solution to combinatorial problems. *Science* 266, 1021–1024, 1994.
- [2] Frisco, P., Parallel arithmetic with splicing. *Romanian J. Inf.Sci. Technol.* 2, 113–128, 2002.
- [3] Fujiwara, A., Matsumoto, K., Chen, Wei. Procedures for logic and arithmetic operations with DNA molecules. *Int. J. Found. Comput. Sci.* 15, 461–474, 2004.
- [4] Guarnieri, F., Fliss, M., Bancroft, C. Making DNA add. *Science* 273, 220–223, 1996..
- [5] Guo, M.Y., Chang, W.L., Ho, M., Lu, J., Cao, J.N. Is optimal solution of every NP-complete or NP-hard problem determined from its characteristic for DNA-Based computing. *BioSystem* 80, 71–82, 2005.
- [6] Gupta, V., Parthasarathy, S., Zaki, M.J. Arithmetic and logic operations with DNA. In: *Proceedings of Third DIMACS Workshop on DNA-Based Computers*, 212–220, 1997.
- [7] Hug, H., Schuler, R. DNA-based parallel computation of simple arithmetic. In: *Proceedings of the Seventh International Meeting on DNA-based Computers*, 159–166, 2001.
- [8] Kamio, S., Takehara, A., Fujiwara, A. Procedures for computing the maximum with DNA strands. In: *Arabnia, Humid, R., Mun, Youngsong (Eds.), In: Proceedings of the International Conference on DNA-Based Computers*. 2003.
- [9] Li, D., Huang, H., Li, X., Li, X. Hairpin formation in DNA computation presents limits for large NP-complete problems. *BioSystem* 72, 203–207, 2003.
- [10] Lipton, R.J. DNA solution of HARD computational problems. *Science* 268, 542–545, 1995.
- [11] Ouyang, Q., Kaplan, Peter, D., Liu, S., Libchaber, A. DNA solution of the maximal clique problem. *Science* 278, 446–449, 1997.
- [12] Păun, G., Rozeberg, G., Salomaa, A. *DNA Computing*. Springer-Verlag. 1998.