Computing and Informatics, Vol. 28, 2009, 599–618

MOLECULAR SOLUTIONS FOR DOUBLE AND PARTIAL DIGEST PROBLEMS IN POLYNOMIAL TIME

Mohammad Ganjtabesh

Laboratoir d'Informatique
Ecole Ploytechnique
Palaiseau CEDEX, France
&
Center of Excellence in Biomathematics
School of Mathematics, Statistics, and Computer Science
University of Tehran, Tehran, Iran
e-mail: mgtabesh@ut.ac.ir

Hayedeh Ahrabian, Abbas Nowzari-Dalini

Center of Excellence in Biomathematics
School of Mathematics, Statistics, and Computer Science
University of Tehran, Tehran, Iran
e-mail: {ahrabian, nowzari}@ut.ac.ir

Manuscript received 3 October 2008; revised 7 January 2009
Communicated by Ivan Plander

Abstract. A fundamental problem in computational biology is the construction of physical maps of chromosomes from the hybridization experiments between unique probes and clones of chromosome fragments. Double and partial digest problems are two intractable problems used to construct physical maps of DNA molecules in bioinformatics. Several approaches, including exponential algorithms and heuristic algorithms, have been proposed to tackle these problems. In this paper we present two polynomial time molecular algorithms for both problems. For this reason, a molecular model similar to Adleman and Lipton model is presented. The presented operations are simple and performed in polynomial time. Our algorithms are computationally simulated.

Keywords: DNA computing, bioinformatics, digest problem.

Mathematics Subject Classification 2000: 68Q15, 68W10, 68W25

1 INTRODUCTION

The field of DNA computing was pioneered by Adleman [1] who showed the potential of using biomolecules for solving computational problems. He solved an instance of the NP-complete Hamiltonian path problem in linear time. Later, Lipton demonstrated that Adleman's experiment could be used to determine the NP-complete satisfiability problem [17]. The major goal of subsequent research in DNA computing area is to develop new techniques to solve NP-complete problems that can not be solved by current electronic computers in a reasonable amount of time. In this manner, a large number of DNA algorithms have been proposed for a large class of NP-complete problems [9, 17, 18, 20, 35].

There are a number of feasible operations in DNA computing. Based on the employed operations, several DNA based models for computing have been introduced. The first model is used by Adleman and Lipton, called Adleman-Lipton model which contains the simple DNA operations [1, 2, 7, 17]. The other popular models are: sticker model [15, 23, 35], surface-based [30], self-assembly [32] and splicing system [11, 19, 24]. These models are different in molecular operations and creating the state space of problem.

A human chromosome which is a DNA molecule of about 10^8 base pairs is too long to be studied entirely and must be broken into fragments or clones. Depending on the cloning technology used, the size of the clones may be as small as 3 000 base pairs or as large as 2 000 000 base pairs. Information is gathered from the individual clones, and then the DNA is constructed by mathematically determining the position of the clones. The process of reconstructing the DNA sequence by the broken fragments is called Digest Problem. Digest experiment plays an important role in molecular biology. In such experiments, enzymes are used to cleave DNA molecules at specific sequence patterns, the restriction sites. The resulting fragments are used in many different ways to study the structure of DNA molecules. Double Digest Problem (DDP) and Partial Digest Problem (PDP) are two famous combinatorial problems that no polynomial time algorithm with electronic computer is given for them until now [31].

In the DDP, we are given the lengths of DNA fragments arising from digestion experiments with two enzymes, and we want to find a physical map of the DNA, i.e. the positions of the restriction sites of the enzymes along the DNA sequence. DDP is known to be NP-Complete [13, 31]. In the PDP, we are given DNA fragment lengths arising from digestion experiment with only one enzyme, and we again ask for a physical map of the DNA. Neither a proof of NP-completeness nor a polynomial time algorithm is known for PDP [6, 21, 22].

In this paper, we present two molecular algorithms for solving DDP and PDP problems in polynomial time complexity. The molecular model employed in these algorithms is similar to Adleman-Lipton model to construct solution space of DNA strands for these two combinatorial problems.

The paper is organized as follows: In Section 2, two digestion problems (DDP and PDP) are defined. Our DNA computational model is given in Section 3. The DNA encoding and molecular algorithm for DDP problem are presented in Section 4. In Section 5, DNA encoding and molecular algorithm for PDP problem are presented. In Section 6, we present a genetic algorithm for constructing error resistance DNA sequences. The computational simulation of two algorithms are given in Section 7. Finally, the conclusion is given in Section 8.

2 DIGESTING DNA

A DNA molecule is a large molecule that is composed of smaller molecules, the nucleotides. There are four nucleotides, namely Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). A nuclease is an enzyme that can cleave DNA molecules at specific restriction sites. This process is called digestion. If the enzyme is applied for long enough time, then it cuts all restriction sites in each clone, yielding fragments between any two adjacent restriction sites. This process is called full or complete digestion, in contrast to partial digestion, where only very small amount of enzymes are used, we obtain all fragments between any two restriction sites (that do not need to be adjacent). Digest experiments can be used to construct physical maps of DNA molecules. A physical map describes the location of markers along the DNA molecules. For constructing the physical map, we can either use a single digestion by applying one enzyme or double digestion by applying two different enzymes.

The fragment length resulting from a single full digestion experiment can not yield any information about the ordering of the fragments or the positions of the restriction sites. For this reason, double digestion experiments are performed where two different enzymes are used as follows. First a set of clones of the DNA molecules are digested by an enzyme A. Then a second set of clones are digested by an enzyme B. Finally, the third set of clones are digested by a mix of both enzymes A and B, which we refer to as C. All digestions are full digestion. This results in three multisets of DNA fragments, and in three multisets of distance between all adjacent restriction sites. The objective is to reconstruct the original ordering of the fragments in the DNA molecules. This is referred to as Double Digest Problem (DDP). In the following definition of DDP, $\text{sum}(S)$ denotes the sum of the elements in a multiset S, and $\text{dist}(P)$ is the multiset of all distances between two neighboring points in a set P of points on a line.

Definition 1 (Double Digest)**.** Given three multisets A, B and C of positive integers with $\text{sum}(A) = \text{sum}(B) = \text{sum}(C)$, there are three sets P^A, P^B and P^C of points on a line, such that $\text{dist}(P^A) = A$, $\text{dist}(P^B) = B$, $\text{dist}(P^C) = C$ and $P^A \cup P^B = P^C$ (0 is the minimal point in each set).

For example, given multisets $A = \{1, 2, 3, 5\}$, $B = \{2, 2, 3, 4\}$ and $C = \{1, 1, 1, 2, 2, 2, 2\}$ as an instance of Double Digest. Then $P^A = \{0, 2, 7, 10, 11\}$, $P^B = \{0, 3, 5, 9, 11\}$ and $P^C = \{0, 2, 3, 5, 7, 9, 10, 11\}$ is a feasible solution which is shown in Figure 1.

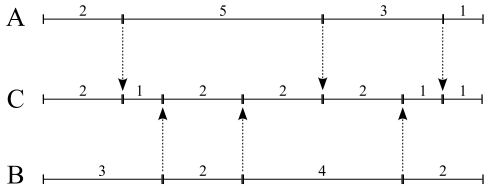


Fig. 1. An instance of *DDP*

DDP is an NP-complete problem [13, 31] and several approaches including exponential algorithms, heuristic algorithms, and computer assistant interactive strategies have been proposed in order to tackle this problem [3, 5, 14, 33]. In Section 4, we give a molecular polynomial time algorithm for this problem. The other approach for finding physical maps of DNA molecules is by partial digestion experiment.

Definition 2 (Partial Digest)**.** Given an integer m and a multiset $D = \{d_1, d_2, \ldots, d_k\}$ of $k = \binom{m}{2}$ positive integers, there is a set $P = \{p_1, p_2, \ldots, p_m\}$ of m points on a line such that $\{|p_i - p_j| : i \leq j \leq m\} = D$.

For example, for the distance multiset $D = \{2, 3, 5, 7, 8, 10\}$, the point set $P = \{0, 2, 7, 10\}$ is a feasible solution which is shown in Figure 2.

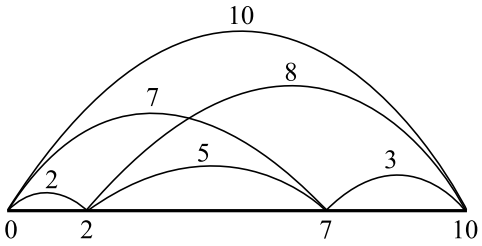


Fig. 2. An instance of *PDP*

The exact computational complexity of *PDP* is a long standing open problem. For this problem, neither a polynomial time algorithm nor a proof of NP-completeness is known [6, 21, 22], but several backtracking algorithm with exponential time complexity are presented in [28, 34]. In Section 5 we also give a polynomial time molecular algorithm for this problem.

3 DNA MODEL OF COMPUTATION

Prior to presentation of the molecular algorithms for our problems, we define the DNA operations that we apply in our algorithms [25]. Our DNA operations are similar to operations proposed by Adleman and Lipton [1, 17]. A test tube is a set of DNA molecules (i.e. a multiset of finite strings over the alphabet $\{A, C, G, T\}$). Given a test tube, we can perform the following operations:

i) Separate(P, P_1, P_2, s): This operation produces two tubes P_1 and P_2 where P_1 is all of the DNA molecules separated from P which contain the strand s as a sub-strand and P_2 is all of the DNA molecules from P which do not contain the short strand s as a sub-strand. To implement this operation, the content of tube P is affinity purified with a biotin-avidin magnetic beads system. This is accomplished by incubating the single-stranded DNA in tube P with $\overline{s}$ conjugated to magnetic beads. Only those single-stranded DNA molecules that contain the short DNA strand s can be annealed to the bound $\overline{s}$. Now these strands are separated and poured to P_1 and the remaining strands are poured to P_2.

ii) Extract(P, P_1, P_2, i, s): As the above operation, this operation produces two tubes P_1 and P_2, where P_1 is all of the DNA strands separated from the tube P which contain the strand s as a sub-strand in a specific position i. An arbitrary length ℓ can be used for scaling the position, depending on the algorithm. After each separation operation, the strands that have the strand s in the i^{th} position will be stored in P_1, while all the strands that don't have the strand s in the i^{th} position will be stored in P_2. This is accomplished by incubating the single-stranded DNA in the test tube P with $\overline{s}$ conjugated to magnetic beads. Only those single-stranded DNA molecules that contain the sequence s are annealed to the bound $\overline{s}$. In the suitable condition, primer extension occurs in the test tube. By melting the double-stranded DNA sequences, the strands which contain $\overline{s}$ are separated in the test tube P_2 and the rest remains in P. Employing gel-electrophoresis, the strands with length $i \times \ell$ are detected. Later, these strands are added to tube P_1 in order to distinguish the strands which contain s in position i.

iii) Append(P, s): Molecularly, this operation appends the strand s into the end of every strand in the tube P. This operation is implemented by pouring the strands containing the end complementary strands in the tube P ($3'$-termination of the strands in P) and the complement of sequence s ($\overline{s}$) to the tube P. With adding the strand s and the ligase enzyme to the tube P, s is appended to all the strands in P.

iv) Merge(P, P_1, P_2): This operation combines the strands in P_1 and P_2 into one test tube P, without any change in the individual strands. This operation is implemented easily by pouring the content of two test tubes P_1 and P_2 in the test tube P.

v) Primer-Extension(P_1, P_2): By pouring the contents of the test tube P_2 in the tube P_1, in the suitable condition, and employing the strands in P_2 as probes, primer extension occurs for each single strand in P_1 in the direction $5' \rightarrow 3'$, i.e. all the strands in P_2 are hybridized to their corresponding complement strands in P_1 and by adding free nucleotides to the solution in the test tube, primer extension occurs and double strands are constructed.

vi) Amplify(P, P_1, P_2): This operation produces two new tubes P_1 and P_2 such that P_1 and P_2 are two copies of tube P (which are now identical) and tube P becomes empty tube. To implement this operation, the content of P is amplified by polymerase chain reaction (PCR) using primers $5'$-terminate and marked $3'$-terminate (by magnetic beads) of the strands in P. The unmarked and marked sequences are distinguished and assigned to the test tubes P_1 and P_2, respectively. The marked sequences in P_2 become unmarked later.

vii) Length(P, P_1, ℓ): This operation separates all DNA strands of length ℓ from P, and pours them into P_1. Molecularly, the content of P is run in gel-electrophoresis, and the strands of length ℓ are separated and assigned to the test tube P_1.

viii) Test(P): This operation produces "true" if P includes at least one DNA strand. This operation can also be done by amplifying the content of P by polymerase chain reaction and run on a gel-electrophoresis.

ix) Readout(P): This operation describes all the stands in the test tube P in a recognizable form.

x) Discard($P_1, P_2, \ldots, P_n$): This operation will discard all the tubes P_1, P_2, $\ldots$, P_n.

Note that in all of the above operations, all the DNA strands can be assumed to be single strands and each operation is performed in $O(1)$ time complexity.

4 THE MOLECULAR ALGORITHM FOR DDP

The molecular algorithm given in this section for DDP problem can be simulated by DNA operations in a polynomial time. Our simulation has two stages: DNA encoding and performing DNA operations. As mentioned, in the definition of the DDP problem we have given three multisets A, B and C of positive integers with $t = \text{sum}(A) = \text{sum}(B) = \text{sum}(C)$. Now, before the construction of strands corresponding to the encoding of the problem, these three sets are encoded as 0-1 sequences and the solution space is constructed with regard to these sequences and sets.

Each set, such as $S = \{s_1, s_2, \ldots, s_n\}$ of length n ($|S| = n$), can be encoded as t-bit binary number where $t = \sum_{i=1}^{n} s_i$, with exactly n zeros and $k = \sum_{i=1}^{n} (s_i - 1)$ ones, such that each element of S, say s_i, is encoded by $s_i - 1$ ones followed by one zero. The sets A and B in the DDP problem in our example can be converted to the following binary numbers:

$$A = [2, 5, 3, 1] \Longrightarrow [10, 11110, 110, 0] \Longrightarrow A' = [10111101100],$$
$$B = [3, 2, 4, 2] \Longrightarrow [110, 10, 1110, 10] \Longrightarrow B' = [11010111010].$$

Consequently, the set C' can be obtained by performing the binary AND operation on A' and B':

$$C' = A' \text{ And } B' = [10111101100] \text{ And } [11010111010] = [10010101000].$$

Therefore, the solution to the DDP problem is to construct all the permutations of binary representation of the sets A and B. The correct mapping is any permutation of A and B that the result of AND operation on them is equal to the binary representation of one of the permutations on C.

For encoding DDP problem the following sets of DNA sequences are constructed:

- Two different DNA strands of a fixed length ℓ are constructed for representing the bit values 0 and 1 which are denoted by $\mathbf{0}$ and $\mathbf{1}$, respectively.
- A set containing of $|C|$ different DNA strands of fixed length ℓ are constructed for representing the bit value 0 which are used as delimiter in the multisets A, B, and C and denoted by $\mathbf{0}_i$ $(1 \leq i \leq |C|)$. These strands help us to easily verify the occurrence of each element of A, B, and C exactly once later in the algorithm.
- For each element a_i in A , we construct a strand of length $\ell \times a_i$ denoted by α_i $(1 \leq i \leq |A|)$, such that α_i is constructed by $a_i - 1$ strands of $\mathbf{1}$ representing bit 1 followed by a strand $\mathbf{0}_i$ representing bit 0. Clearly, for the elements with equal values in A, different strands are constructed.
- Strands corresponding to the elements of B are constructed similarly to the above definition for the elements of A and denoted by β_j $(1 \leq j \leq |B|)$.
- For each element c_i in C, we construct a strand of length $\ell \times (c_i + 1)$ denoted by γ_i, such that γ_i is constructed by $c_i - 1$ strands of $\overline{\mathbf{1}}$ (the complement of $\mathbf{1}$) followed by a strand $\overline{\mathbf{0}}$ (the complement of $\mathbf{0}$) and also followed by a strand $\overline{\mathbf{0}}_i$ (thecomplement of $\mathbf{0}_i$).
- We construct a set of sequences that contain all permutations of the strands in $A = \{a_1, a_2, \ldots, a_n\}$. Similar to Adleman's scheme for encoding the Hamiltonian path problem [1], this can be done easily by considering a complete graph whose vertices are the elements of the set A. For each vertex, we consider the strand α_{a_i} of length $\ell \times a_i$ corresponding to the elements in A and for each edge of the graph the strand $\alpha_{e_{ij}}$ are constructed such that $\alpha_{e_{ij}} = \overline{\alpha}^R_{a_i} + \overline{\alpha}^L_{a_j}$ (the right half end of the complement of α_{a_i} corresponding to vertex i plus the left half end of the complement of α_{a_j} corresponding to vertex j). Pouring the edge strands and vertex strands in a test tube, in a suitable condition, all the paths of this graph are constructed. The paths with length equal to $\ell \times \sum_{i=1}^{n} a_i$ are selected such

that each zero strand $\mathbf{0}_i$ $(1 \leq i \leq n)$ occurs exactly once in the sequences. These selected paths represent the permutations of the strands corresponding to all the vertices. For example for $A = \{2, 4, 4\}$ we construct a graph which is shown in Figure 3. The strands corresponding to all the paths represent all the permutations of the set A. We call this set of strands Δ_A, where $|\Delta_A| = |A|!$. The ith sequence in each set is denoted by an index i, for example in the set Δ_A, the ith element is denoted by Δ_{A_i}. Considering that the bit values $0_i (1 \leq i \leq n)$ for each element a_i in A, are encoded with different strands, therefore elements with equal values in A are also encoded with different sequences. For this reason, the verification of the occurrence of exactly n zeros is very simple and the paths with cycle can not be chosen.

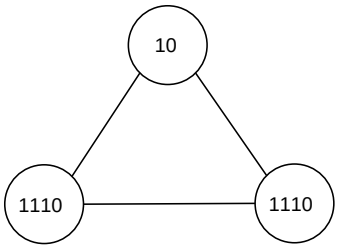


Fig. 3. Graph corresponding to the multiset $A = \{2, 4, 4\}$

- Similar to the above discussion, all the permutations of strands in B are constructed and we call this set Δ_B.
- Sequences corresponding to all the permutations of C are also constructed similar to A and are called Δ_C. As it is defined, each sequence γ_i $(1 \leq i \leq |C|)$ corresponding to the element c_i, contains of $c_i - 1$ strands of $\overline{\mathbf{1}}$ followed by the complementary of two zero strands $\overline{\mathbf{0}}$ and $\overline{\mathbf{0}}_i$. Clearly, in construction of the permutations of the set C, the subsequences $\overline{\mathbf{0}}_i$ are used as delimiter and this causes the elements with equal values in C are encoded with different sequences. After the construction of all permutations for C, we can delete the subsequences $\overline{\mathbf{0}}_i$ and in the algorithm we do not need these subsequences. By using polymerase and ligase, we obtain double stranded molecules, with one strand being the original one and the other strand being its complement except that the subsequence $\overline{\mathbf{0}}_i$ is removed. After denaturing, the strands which contain $\overline{\mathbf{0}}_i$ are separated and the remaining strands are the strands with $\overline{\mathbf{0}}_i$ removed in them. This process should be performed for removing each $\overline{\mathbf{0}}_i$ $(1 \leq i \leq |C|)$, separately.

It should be noted that, for the convenience in the separation operation, all the strands in Δ_A and Δ_B begin with a specific strand α_A and β_B (tag strands), and all the strands in Δ_C end with the strand γ_C, respectively, as a marker.

Now, assume that all strands of the sets Δ_A, Δ_B, and Δ_C are in the test tubes P_α, P_β, and P_γ, respectively. Note that each strand in Δ_A, Δ_B, and Δ_C is a DNA

sequence of length $\ell \times (t+1)$, where $t = \sum_{i=1}^{|A|} a_i = \sum_{i=1}^{|B|} b_i = \sum_{i=1}^{|C|} c_i$. The molecular algorithm for DDP is given in Algorithm 1.

Now, a brief discussion of the performance of the above algorithm is presented. After the construction of all the required sequences, the algorithm $DDP - MOL$ is performed as follows. In phase 1, a set of molecular operations are performed for the AND operation. This can be done by checking the x^{th} position in both sequences. Depending on the values, the result of AND operation is appended to both sequences, i.e. for each strand in Δ_A namely Δ_{A_i} $(1 \leq i \leq |A|!)$ and each strand in Δ_B namely Δ_{B_j} $(1 \leq j \leq |B|!)$, we obtain Δ_{A_i} AND Δ_{B_j} and append the result of AND operation to the end of Δ_{A_i} and Δ_{B_j}. Later, all the sequences corresponding to the elements of A and B are separated in two different test tubes, P_α and P_β, respectively.

In phase 2, by employing the operation Primer-Extension on the test tubes P_α and P_β separately using the content of test tube P_γ as primer, double strands are constructed. Finally, the sequences which hold all the strands Δ_C are extracted as a solution of our problem in each test tube.

Theorem 1. The molecular algorithm *DDP-MOL* is performed in $O(n)$ complexity for any DDP problem given three multisets A, B, and C of positive integers such that n is a constant proportional to the sum(A) (note that sum(A) = sum(B) = sum(C)).

Proof. The *DDP-MOL* has two phases. Phase 1 has a single loop and all the operations in the loop are performed in $O(1)$. Therefore phase 1 is performed in $O(n)$ complexity. In phase 2, we have no loop and this is performed in $O(1)$ complexity. Hence, the total complexity of the algorithm is $O(n)$. □

5 THE MOLECULAR ALGORITHM FOR PDP

In this section, we present a molecular algorithm for the PDP. The algorithm is simulated by DNA operations in a polynomial time. The main idea of our simulation is to first generate solution space of DNA sequences. Then, the biological operations are used to remove infeasible solutions and to find feasible solutions from the solution space.

As mentioned in the previous section, PDP problem can be briefed as follows: Given an integer m and a multiset $D = \{d_1, d_2, \ldots, d_k\}$ of $k = \binom{m}{2}$ positive integers, is there a set $P = \{p_1, p_2, \ldots, p_m\}$ of m points on a line such that $\{|p_i - p_j| : i \leq j \leq m\} = D$.

In order to encode PDP problem as DNA sequences, we construct three different sets of strands:

- For the multiset $D = \{d_0, d_1, d_2, \ldots, d_k\}$, $k+1$ different strands δ_{d_0}, δ_{d_1}, δ_{d_2}, …, δ_{d_k} of a constant length ℓ are constructed such that each δ_{d_i} corresponds to a d_i $(0 \leq i \leq k)$. It is noted that for elements with equal values in D, different strands are constructed. Here we assume that d_0 is an element with value 0

Algorithm 1 Double digest molecular algorithm

Algorithm *DDP-MOL*
begin
phase 1:
    for $x = 1$ to t do begin
        Extract(P_α, P_0, P_1, x, 1);
        Extract(P_β, P_2, P_3, x, 1);
        Amplify(P_0, P_4, P_5);
        Amplify(P_1, P_6, P_7);
        Amplify(P_2, P_8, P_9);
        Amplify(P_3, P_{10}, P_{11});
        if (Test(P_4) and Test(P_8)) then begin
            Merge(P_{12}, P_4, P_8);
            Append(P_{12}, **1**);
        end if;
        if (Test(P_5) and Test(P_{10})) then begin
            Merge(P_{13}, P_5, P_{10});
            Append(P_{13}, **0**);
        end if;
        if (Test(P_6) and Test(P_9)) then begin
            Merge(P_{14}, P_6, P_9);
            Append(P_{14}, **0**);
        end if;
        if (Test(P_7) and Test(P_{11})) then begin
            Merge(P_{15}, P_7, P_{11});
            Append(P_{15}, **0**);
        end if;
        Extract(P_{12}, P_α, P_β, 0, α_A);
        Extract(P_{13}, P_α, P_β, 0, α_A);
        Extract(P_{14}, P_α, P_β, 0, α_A);
        Extract(P_{15}, P_α, P_β, 0, α_A);
        Discard(P_0, P_1, ..., P_{15});
    end for;
phase 2:
    Primer-Extension(P_α, P_γ);
    Primer-Extension(P_β, P_γ);
    Extract(P_α, P_0, P_1, $2t + 2$, γ_C);
    Extract(P_β, P_2, P_3, $2t + 2$, γ_C);
    if Test(P_α) and Test(P_β) then begin
        Readout(P_α);
        Readout(P_β);
    end if;
end.

and without loss of generality, it is included in the set D. We call this set of strands Γ.

- Two different DNA strands of fixed length ℓ are constructed for representing the bit values 0 and 1. We show these strands by **0** and **1**, respectively.
- A set of strands of length $(k+1) \times \ell$ corresponding to all 0-1 combinations of binary numbers of length $k+1$ are provided by employing the **0** and **1** strands. This set is constructed similarly to Lipton's scheme for encoding the satisfiability problem. We call this set Δ. Indeed, each strand of length $(k+1) \times \ell$ in this set corresponds to a binary number of length $k+1$. Therefore, the set Δ contains the strands corresponding to all $(k+1)$-bits binary numbers.

After construction of the above strands, and assuming that the strands of the set Δ are in to the tube P_0, the molecular algorithm for PDP is presented in Algorithm 2.

A brief discussion of this algorithm is presented here. In phase 1, all the sequences corresponding to the values in the set D are appended to the sequences in the initial test tube P_0. This process is performed as follows. Consider one of these binary numbers be $x = x_1 x_2 \dots x_k$. If x_i and x_j both have value 1 and $|d_i - d_j|$ belongs to the multiset D, then the sequence $\delta_{|d_i - d_j|}$ from the set Γ is appended to the strands corresponding to this numbers and the value $|d_i - d_j|$ is removed from the multiset D. Because of this removal, different strands are appended for equal values in D.

In phase 2, all the strands in P_1 corresponding to the binary numbers with exactly m-bits value 1 that have appended values with length greater or less than $2k+1$ are discarded. Later, the sequences with the appended strands corresponding to all the values in D are selected. The selected strands are the solution for PDP problem.

It should be noted that this algorithm obtains all the feasible solutions, but at least we have two complement solutions because of changing the direction of the original sequence.

Theorem 2. The molecular algorithm $PDP\text{-}MOL$ is performed in $O(k^2)$ time complexity for any PDP problem given as integer m and a multiset $D = \{d_0, d_1, d_2, \dots, d_k\}$ of k distances.

Proof. As we can see, the algorithm has two phases. Phase 1 has two nested loops. Since all the molecular operations employed in this paper are performed in $O(1)$, therefore phase 1 is performed in $O(k^2)$. In phase 2 we have two separated loops. With regards to the complexity of the operations, phase 2 is also performed in $O(k)$. So the total complexity of the algorithm is $O(k^2)$. □

6 ERROR RESISTANCE DNA SEQUENCE GENERATION

At the time when DNA computing was introduced, a question was raised about how errors may affect the computing results. Although mature biological operations

Algorithm 2 Partial digest molecular algorithm.

Algorithm $PDP\text{-}MOL$
begin
phase 1:
  for $i = 0$ to $k - 1$ do begin
    for $j = i + 1$ to k do begin
      $Extract(P_0, P_1, P_2, i, 1)$;
      $Extract(P_1, P_3, P_4, j, 1)$;
      if $|d_i - d_j| \in D$ then begin
        $D = D \backslash \{|d_i - d_j|\}$
        $Append(P_3, \delta_{|d_i - d_j|})$
        $Merge(P_0, P_2, P_3)$;
        $Merge(P_0, P_0, P_4)$;
      end if;
    end for;
  end for;
phase 2:
  $Length(P_0, P_1, 2(k+1)\ell)$;
  for $i = 1$ to k do begin
    $Separate(P_1, P_0, P_2, \delta_{d_i})$;
    $P_1 \leftarrow P_0$;
  end for;
  for $i = 0$ to k do begin
    $Extract(P_1, P_2, P_3, i, 1)$;
    if $T(P_2)$ then begin
      print(i);
      $P_1 \leftarrow P_2$;
    else
      $Merge(P_1, P_2, P_3)$;
    end if;
  end for;
end.

have very low error rates, errors may still accumulate and thus generate incorrect answers. As hybridization reactions are essential components in the implementation protocol and as under certain circumstances such reactions are prone to errors that would cause false positive and negative results, care must be taken to avoid the errors. Errors arise commonly when sequences that generate or tolerate hairpins and internal loops, mismatch hybridization, shifted mismatches, and 3′-end hybridization are used. Application of appropriate sequence considerations, such as base composition and melting temperature (T_m) of hybrids have proven to be very effective in other experimental contexts where mis-hybridization was a critical con-

sideration [10]. Similarly, as ligation reactions are included in the implementation protocol, care must be taken to avoid mis-ligations. The use of an appropriate ligase under stringent conditions will prevent mis-ligations [10, 26]. In order to avoid these errors we should prevent mis-hybridization and undesired secondary structure and also keep uniform chemical characteristics [8, 29]. Such unintended interactions among DNA strands can be minimized by careful sequence design. Random sequence generation does not satisfy the above properties. Several heuristic methods for constructing DNA strands set that are robust with respect to the hybridization errors have been proposed [4, 10, 16, 26].

DNA sequence design can be considered as an optimization problem. Therefore, DNA sequences corresponding to any encoding can be generated by a genetic algorithm that minimizes the potential of errors in DNA sequences for reliable molecular operations and produces reliable sequences. Based on ideas given in Shin et al. [27], we discuss a genetic algorithm for constructing DNA sequences with regard to the above errors for our model. This algorithm is summarized below:

i) Generate sequences of four alphabets $\{A, C, G, T\}$ randomly with length $2d$.

ii) Set $counter = 1$.

iii) While ($counter <= max\_count$) do

   a) Evaluate the fitness of each sequence.
   b) Select the sequences with best fitness.
   c) Apply genetic operators (crossover and mutation) to produce a new population.
   d) $counter = counter + 1$.

iv) Let the best codes be the fittest encodings.

In this algorithm, we use the conventional genetic operations such as *roulette wheel selection*, *one-cut-point crossover*, and *single-point mutation* [12]. In roulette wheel selection, offsprings with a higher fitness value have a higher probability of contributing in the next population. Crossover and mutation are applied with probability p_c and p_m, respectively, in offsprings that are selected by roulette wheel selection process. In this algorithm, we have employed a multiobjective fitness function with similar criteria given in Tanaka et al. [29]. Three measures, *H-Measure*, *Similarity*, and *Completely complementary at* $3'$*-end*, are considered for avoiding mis-hybridization error. To prevent an undesired secondary structure two measures *Self-Complementary* and *Continuity* are considered. To keep a uniform chemical characteristics, two other measures, *GC-Content* and *Temperature* (T_m), are used. In our algorithm, a unique fitness function is designed for each of these measures, and the multiobjective fitness function is the summation of the above fitness functions.

7 COMPUTATIONAL SIMULATION

We developed a tool for simulating our two molecular algorithms for DDP and PDP. Our tool first provides the required DNA sequences for these problems using the genetic algorithm discussed in the previous section and later performs the simulated molecular operations on them. The first simulated example that we present here is a DDP for a given three multisets $A = \{2, 4, 4\}$, $B = \{2, 3, 5\}$, and $C = \{1, 2, 2, 2, 3\}$. The generated sequences for this example by our genetic algorithm are also of length $\ell = 6$. The strands corresponding to 0, 1, and $0_i (1 \leq i \leq 3)$ are constructed and shown in Table 1. The elements in the multisets A, B and C are encoded to 0-1 digits, with respect to the encoding which is defined in Section 5, and their corresponding DNA sequences are constructed by the strands corresponding to 0 and 1 as shown in Table 2.

Bit	Symbol	Corresponding DNA sequences	Complement DNA sequences
0	**0**	$GCCATT$	$CGGTAA$
1	**1**	$CATGAC$	$GTACTG$
0_1	$\mathbf{0}_1$	$AGTCAC$	$TCAGTG$
0_2	$\mathbf{0}_2$	$CGTACA$	$GCATGT$
0_3	$\mathbf{0}_3$	$ATCTCG$	$TAGAGC$
0_4	$\mathbf{0}_4$	$TAGAGG$	$ATCTCC$
0_5	$\mathbf{0}_5$	$TGAGTC$	$ACTCAG$

Table 1. 0-1 bits and their corresponding DNA sequences

Multiset	i	Encoded	Symbol	Corresponding DNA sequences
A	2	10_1	α_1	$CATGAC$-$AGTCAC$
	4	1110_2	α_2	$CATGAC$-$CATGAC$-$CATGAC$-$CGTACA$
	4	1110_3	α_3	$CATGAC$-$CATGAC$-$CATGAC$-$ATCTCG$
B	2	10_1	β_1	$CATGAC$-$AGTCAC$
	3	110_2	β_2	$CATGAC$-$CATGAC$-$CGTACA$
	5	11110_3	β_3	$CATGAC$-$CATGAC$-$CATGAC$-$CATGAC$-$ATCTCG$
C	1	00_1	γ_1	$GCCATT$-$TCAGTG$
	2	100_2	γ_2	$GTACTG$-$GCCATT$-$GCATGT$
	2	100_3	γ_3	$GTACTG$-$GCCATT$-$TAGAGC$
	2	100_4	γ_4	$GTACTG$-$GCCATT$-$ATCTCC$
	3	1100_5	γ_5	$GTACTG$-$GTACTG$-$GCCATT$-$ACTCAG$

Table 2. Elements of A, B, and C and their corresponding strands

The three sets Δ_A, Δ_B, and Δ_C are all sequences corresponding to the permutations of the elements in A, B, and C, which are generated and shown in Tables 3 and 4. It should be noted that, for summarizing, in Tables 3 and 4 the DNA sequences are presented by their corresponding symbols denoted in Table 2. As mentioned previously, for performing the Primer-Extention operation in the algorithm,

the set Δ_C is constructed by using the Watson-Crick complementary sequences. So, for constructing all permutation for the multiset C we use the complement strand of γ_i as shown in Table 4.

Set	Permutation	Corresponding DNA sequence
Δ_A	$10_1 - 1110_2 - 1110_3$	$\alpha_A - \alpha_1 - \alpha_2 - \alpha_3$
	$10_1 - 1110_3 - 1110_2$	$\alpha_A - \alpha_1 - \alpha_3 - \alpha_2$
	$1110_2 - 10_1 - 1110_3$	$\alpha_A - \alpha_2 - \alpha_1 - \alpha_3$
	$1110_2 - 1110_3 - 10_1$	$\alpha_A - \alpha_2 - \alpha_3 - \alpha_1$
	$1110_3 - 10_1 - 1110_2$	$\alpha_A - \alpha_3 - \alpha_1 - \alpha_2$
	$1110_3 - 1110_2 - 10_1$	$\alpha_A - \alpha_3 - \alpha_2 - \alpha_1$
Δ_B	$10_1 - 110_2 - 11110_3$	$\beta_B - \beta_1 - \beta_2 - \beta_3$
	$10_1 - 11110_3 - 110_2$	$\beta_B - \beta_1 - \beta_3 - \beta_2$
	$110_2 - 10_1 - 11110_3$	$\beta_B - \beta_2 - \beta_1 - \beta_3$
	$110_2 - 11110_3 - 10_1$	$\beta_B - \beta_2 - \beta_3 - \beta_1$
	$11110_3 - 10_1 - 110_2$	$\beta_B - \beta_3 - \beta_1 - \beta_2$
	$11110_3 - 110_2 - 10_1$	$\beta_B - \beta_3 - \beta_2 - \beta_1$

Table 3. The permutations of the elements in A and B appended by tag strand α_A and β_B, respectively

Set	Permutation	Corresponding DNA sequence
Δ_C	$0 - 10 - 10 - 10 - 110$	$\gamma_1 - \gamma_2 - \gamma_3 - \gamma_4 - \gamma_5 - \gamma_C$
	$0 - 10 - 10 - 110 - 10$	$\gamma_1 - \gamma_2 - \gamma_3 - \gamma_5 - \gamma_4 - \gamma_C$
	$0 - 10 - 110 - 10 - 10$	$\gamma_1 - \gamma_2 - \gamma_5 - \gamma_3 - \gamma_4 - \gamma_C$
	$0 - 110 - 10 - 10 - 10$	$\gamma_1 - \gamma_5 - \gamma_2 - \gamma_3 - \gamma_4 - \gamma_C$
	$10 - 0 - 10 - 10 - 110$	$\gamma_2 - \gamma_1 - \gamma_3 - \gamma_4 - \gamma_5 - \gamma_C$
	$10 - 0 - 10 - 110 - 10$	$\gamma_2 - \gamma_1 - \gamma_3 - \gamma_5 - \gamma_4 - \gamma_C$
	$10 - 0 - 110 - 10 - 10$	$\gamma_2 - \gamma_1 - \gamma_5 - \gamma_3 - \gamma_4 - \gamma_C$
	$10 - 10 - 0 - 10 - 110$	$\gamma_2 - \gamma_3 - \gamma_1 - \gamma_4 - \gamma_5 - \gamma_C$
	$10 - 10 - 0 - 110 - 10$	$\gamma_2 - \gamma_3 - \gamma_1 - \gamma_5 - \gamma_4 - \gamma_C$
	$10 - 10 - 10 - 0 - 110$	$\gamma_2 - \gamma_3 - \gamma_4 - \gamma_1 - \gamma_5 - \gamma_C$
	$10 - 10 - 10 - 110 - 0$	$\gamma_2 - \gamma_3 - \gamma_4 - \gamma_5 - \gamma_1 - \gamma_C$
	$10 - 10 - 110 - 10 - 0$	$\gamma_2 - \gamma_3 - \gamma_5 - \gamma_4 - \gamma_1 - \gamma_C$
	$10 - 10 - 110 - 0 - 10$	$\gamma_2 - \gamma_3 - \gamma_5 - \gamma_1 - \gamma_4 - \gamma_C$
	$10 - 110 - 10 - 10 - 0$	$\gamma_2 - \gamma_5 - \gamma_3 - \gamma_4 - \gamma_1 - \gamma_C$
	$10 - 110 - 10 - 0 - 10$	$\gamma_2 - \gamma_5 - \gamma_3 - \gamma_1 - \gamma_4 - \gamma_C$
	$10 - 110 - 0 - 10 - 10$	$\gamma_2 - \gamma_5 - \gamma_1 - \gamma_3 - \gamma_4 - \gamma_C$
	$110 - 10 - 10 - 10 - 0$	$\gamma_5 - \gamma_2 - \gamma_3 - \gamma_4 - \gamma_1 - \gamma_C$
	$110 - 10 - 10 - 0 - 10$	$\gamma_5 - \gamma_2 - \gamma_3 - \gamma_1 - \gamma_4 - \gamma_C$
	$110 - 10 - 0 - 10 - 10$	$\gamma_5 - \gamma_2 - \gamma_1 - \gamma_3 - \gamma_4 - \gamma_C$
	$110 - 0 - 10 - 10 - 10$	$\gamma_5 - \gamma_1 - \gamma_2 - \gamma_3 - \gamma_4 - \gamma_C$

Table 4. The distinct permutations of the elements in C appended by tag strand $\overline{\gamma}_C$

After the construction of all the required sequences, the algorithm $DDP\text{-}MOL$ is simulated as follows. With respect to the algorithm in phase 1, the molecular operations required for the AND operation are computationally simulated and performed on the sequences of the sets Δ_A and Δ_B in the test tubes P_α and P_β, respectively. In this phase, the result of AND operation is appended to the all sequences and separated in two different test tubes, P_α and P_β, with respect to the tag strands called α_A and β_B. In phase 2, by pouring the strands of the set Δ_C which are in the test tube P_γ into each tube and allowing the occurrence of primer extension, double strands are constructed. Finally, the sequences which hold all the strand γ_C are the solution in each test tube. Therefore, the solutions for this example are shown in Table 5. As we can see, both solutions in Table 5 are correct for this problem.

The second simulated example that we present here is an example of PDP for a given multiset $D = \{0, 2, 3, 5, 7, 8, 10\}$ where $k = 6$ and $m = 4$. Our genetic algorithm provides DNA sequences of length $\ell = 6$ for this example. We consider that the strands corresponding to 0 and 1 are similar to the **0** and **1** strands generated in the previous example. Similarly, the strands corresponding to the elements of the multiset D are also constructed and shown in Table 6. Also, the 2^7 binary numbers of length 7 and corresponding sequences of length 7×6 are constructed which are shown in Table 7.

Solution Number	Container set	Corresponding DNA sequences
1	A	$\alpha_A - \alpha_1 - \alpha_2 - \alpha_3 - \gamma_2 - \gamma_1 - \gamma_5 - \gamma_3 - \gamma_4 - \gamma_C$
	B	$\beta_B - \beta_2 - \beta_3 - \beta_1 - \gamma_2 - \gamma_1 - \gamma_5 - \gamma_3 - \gamma_4 - \gamma_C$
2	A	$\alpha_A - \alpha_2 - \alpha_3 - \alpha_1 - \gamma_2 - \gamma_3 - \gamma_5 - \gamma_1 - \gamma_4 - \gamma_C$
	B	$\beta_B - \beta_1 - \beta_3 - \beta_2 - \gamma_2 - \gamma_3 - \gamma_5 - \gamma_1 - \gamma_4 - \gamma_C$

Table 5. The solutions for the given example

d_i	δ_{d_i}
0	TAGCGA
2	TGTACC
3	GCTGAA
5	TCCATC
7	CAATCC
8	TGACGA
10	CGTGTT

Table 6. Strands corresponding to the elements of D

By performing the algorithm and employing simulated molecular operations, in phase 1, for any position i and j with value 1, the sequences corresponding to the value $|d_i - d_j|$ that belong to the multiset D, $\delta_{|d_i - d_j|}$ are appended to these sequences. In phase 2, the sequences with length $2(k+1)\ell$ which in this example is equal to 82 are separated and illustrated in Table 8. Among these sequences, the sequences whose appended subsequences are corresponding to the values in the multiset D are

selected as a solution. The bolded sequences in Table 8 show the solution of this problem. One of the solutions is $1-1-0-0-1-0-1-\delta_2-\delta_7-\delta_{10}-\delta_5-\delta_8-\delta_3$, which represents the points 0, 2, 7 and 10, therefore the solution is $\{0, 2, 7, 10\}$.

0-1 combinations	corresponding strand
0000000	*TAGAGG-TAGAGG-TAGAGG-TAGAGG-TAGAGG-TAGAGG-TAGAGG*
0000001	*TAGAGG-TAGAGG-TAGAGG-TAGAGG-TAGAGG-TAGAGG-CATGAC*
0000010	*TAGAGG-TAGAGG-TAGAGG-TAGAGG-TAGAGG-CATGAC-TAGAGG*
0000011	*TAGAGG-TAGAGG-TAGAGG-TAGAGG-TAGAGG-CATGAC-CATGAC*
0000100	*TAGAGG-TAGAGG-TAGAGG-TAGAGG-CATGAC-TAGAGG-TAGAGG*
⋮	⋮
1111011	*CATGAC-CATGAC-CATGAC-CATGAC-TAGAGG-CATGAC-CATGAC*
1111100	*CATGAC-CATGAC-CATGAC-CATGAC-CATGAC-TAGAGG-TAGAGG*
1111101	*CATGAC-CATGAC-CATGAC-CATGAC-CATGAC-TAGAGG-CATGAC*
1111110	*CATGAC-CATGAC-CATGAC-CATGAC-CATGAC-CATGAC-TAGAGG*
1111111	*CATGAC-CATGAC-CATGAC-CATGAC-CATGAC-CATGAC-CATGAC*

Table 7. Strands corresponding to the binary numbers of length 7

Constructed DNA strands
$0-2-3-5-7-8-10$
$1-0-0-1-0-1-1-\delta_5-\delta_8-\delta_{10}-\delta_3-\delta_5-\delta_2$
$1-0-0-1-1-0-1-\delta_5-\delta_5-\delta_2-\delta_7-\delta_3-\delta_5$
$1-0-1-0-0-1-1-\delta_3-\delta_8-\delta_{10}-\delta_5-\delta_7-\delta_2 \leftarrow solution1$
$1-0-1-1-0-0-1-\delta_3-\delta_5-\delta_{10}-\delta_2-\delta_7-\delta_5$
$1-1-0-0-1-0-1-\delta_2-\delta_7-\delta_{10}-\delta_5-\delta_8-\delta_3 \leftarrow solution2$
$1-1-0-1-0-0-1-\delta_2-\delta_5-\delta_{10}-\delta_3-\delta_8-\delta_5$

Table 8. Produced strands by performing the algorithm

8 CONCLUSION

Two DNA-based algorithms are presented for the solution of PDP and DDP. Our molecular computational model is similar to Adleman-Lipton model. The algorithms

are proved to have polynomial time complexity. Both algorithms are computationally simulated for two different examples. The DNA sequences corresponding to the encoding of these problems are generated by a genetic algorithm that minimizes the potential of errors in DNA sequences for reliable molecular operations and produces reliable sequences.

Acknowlegment

This research was partially supported by University of Tehran.

REFERENCES

[1] Adleman, L.M.: Molecular Computation of Solutions to Combinatorial Problems. Science 266, 1994, pp. 1021–1029.

[2] Adleman, L. M.: Computing with DNA. Sci. Am. 279, 1998, pp. 54–61.

[3] Allison, L.—Yee, C. N.: Restriction Site Mapping Is in Separation Theory. Comput. Appl. Biosci. 4, 1988, pp. 97–101.

[4] Amos, M.—Gibbons, A.—Hodgson, D.: Error-Resistant Implementation of DNA Computations. In DNA Based Computers II, L. Landweber and E. Baum (Eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 44 (American Mathematical Society, Providence, 1999), pp. 151–162.

[5] Bellon, B.: Construction of Restriction Maps. Comput. Appl. Biosci. 4, 1998, pp. 111–115.

[6] Blażewicz, J.—Formanowicz, P.—Kasprzak, M.—Jaroszewski, M.—Markiewicz, W. T.: Construction of DNA Restriction Maps Based on a Simplified Experiment. Bioinformatics 17, 2001, pp. 396–404.

[7] Boneh, D.—Dunworth, C.—Lipton, R. J.—Sgall, J.: On the Computational Power of DNA. Discrete Appl. Math. 71, 1996, pp. 79–94.

[8] Braich, R. S.—Chelyapov, N.—Johnson, C.—Rothemund, P. W. K.—Adleman, L. M.: Solution of a 20-Variable 3-Sat Problem on a DNA Computer. Science 296, 2002, pp. 499–502.

[9] Chang, W. L.—Ho, M.—Guo, M.: Molecular Solutions for the Subset-Sum Problem on DNA-Based Supercomputing. BioSystems 73, 2004, pp. 117–130.

[10] Deaton, R.—Garzon, M.—Murphy, R. C.—Rose, J. A.—Franceschetti, D. R.—Stevens Jr, S. A.: Reliability and Efficiency of a DNA Based Computation. Phys. Rev. Lett. 80, 1998, pp. 417–420.

[11] Freund, R.—Kari, L.—Păun, G.: DNA Computation Based on Splicing: The Existence of Universal Computers. Theory Comput. Syst. 32, 1999, pp. 69–112.

[12] Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Boston 1989.

[13] Goldstein, L.—Waterman, M. S.: Mapping DNA by Stochastic Relaxation. Adv. Appl. Math. 8, 1987, pp. 194–207.

[14] Inglehart, J.—Nelson, P. C.: On the Limitations of Automated Restriction Mapping. Comput. Appl. Biosci. 10, 1994, pp. 249–261.

[15] Kari, L.—Păun, G.—Rozenberg, G.—Salomaa, A.—Yu, S.: DNA Computing, Sticker Systems, and Universality. Acta Inform. 35, 1998, pp. 401–420.

[16] Kim, D.—Shin, S.—Lee, I.—-Zhang, B.: NACST/Seq: A Sequence Design System With Multiobjective Optimization. In Proceedings of 8th International Workshop on DNA Based Computers, M. Hagiya and A. Ohuchi (Eds.), Lecture Notes in Computer Science, Vol. 2568, Springer-Verlag, London 2003, pp. 242–251.

[17] Lipton, R. J.: DNA Solution of Hard Computational Problem. Science 268, 1995, pp. 542–545.

[18] Liu, J.—Shimohara, K.: Signaling-Pathway-Based Molecular Computing for Efficient 3-Sat Problem Solving. Inf. Sci. 161, 2004, pp. 121–137.

[19] Mateescu, A.—Păun, G.—Rozenberg, G.—Salomaa, A.: Simple Splicing Systems. Discrete Appl. Math. 84, 1998, pp. 145–163.

[20] Ouyang, Q.—Kaplan, P. D.—Liu, S.–Libchaber, A.: DNA Solution of the Maximal Clique Problem. Science 278, 1997, pp. 446–449.

[21] Pandurangan, G.—Ramesh, H.: The Restriction Mapping Problem Revisited. J. Comput. System Sci. 65, 2002, pp. 526–544.

[22] Pevzner, P. A.—Waterman, M. S.: Open Combinatorial Problems in Computational Molecular Biology. In Proc. of the 3rd Israel Symposium on Theory of Computing and Systems, IEEE Computer Society Press, Piscataway, 1995, pp. 158–173.

[23] Păun, G.—Rozenberg, G.: Sticker Systems. Theoret. Comput. Sci. 204, 1998, pp. 183–203.

[24] Păun, G.—Rozenberg, G.—Salomaa, A.: Computing by Splicing. Theoret. Comput. Sci. 168, 1996, pp. 321–336.

[25] Păun, G.—Rozenberg, G.—Salomaa, A.: DNA Computing: New Computing Paradigms. Springer-Verlag, Heidelberg 1998.

[26] Roweis, S.—Winfree, E.: On the Reduction of Errors in DNA Computation. J. Comput. Biol. 6, 1999, pp. 65–75.

[27] Shin, S. Y.—Kim, D. M.—Lee, I. H.—Zhang, B. T.: Evolutionary Sequence Generation for Reliable DNA Computing. In Proc. of the 2002 Congress on Evolutionary Computation, D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow and M. Shackleton (Eds.), IEEE Computer Society Press, New Jersey 2002, pp. 79–84.

[28] Skiena, S. S.—Sundaram, G.: A Partial Digest Approach to Restriction Site Mapping. Bull. Math. Biology 56, 1994, pp. 275–294.

[29] Tanaka, F.—Nakatsugawa, M.—Yamamoto, M.—Shiba, T.—Ohuchi, A.: Developing Support System for Sequence Design in DNA Computing. In Proceedings of 7th International Workshop on DNA-Based Computers, N. Jonoska and N. C. Seeman (Eds.), Lecture Notes in Computer Science, Vol. 2340, Springer-Verlag, Berlin 2001, pp. 129–137.

[30] Wang, L.—Liu, Q.—Frutos, A.—Gillmor, S.—Thiel, A.—Strother, T.—Condon, A.—Corn, R.—Lagally, M.—Smith, L.: Surface-Based DNA Computing Operations: Destroy and Readout. Biosystems 52, 1999, pp. 189–191.

[31] Waterman, M. S.: Introduction to Computational Biology. CRC Press, New York 1995.

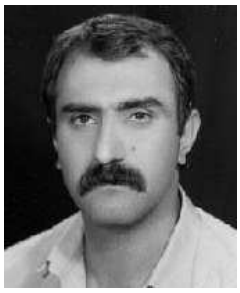
[32] Winfree, E.: DNA Computing by Self-Assembly. The Bridge 33, 2003, pp. 31–38.

[33] Wright, L. W.—Lichter, J. B.—Reinitz, J.—Shifman, J. A.—Kidd, K. K.—Miller, P. L.: Computer-Assisted Restriction Mapping: An Integrated Approach to Handling Experimental Uncertainty. Comput. Appl. Biosci. 10, 1994, pp. 435–442.

[34] Zhang, Z.: An Exponential Example for a Partial Digest Mapping Algorithm. J. Comput. Biol. 1, 1994, pp. 235–239.

[35] Zimmermann, K.: Efficient DNA Sticker Algorithms for Graph Theoretic Problems. Comput. Phys. Comm. 144, 2002, pp. 297–309.

Hayadeh Ahrabian is an Associate Professor of School of Mathematics, Statistics and Computer Science, University of Tehran, and is currently the Director of graduate studies in this school. Her research interest includes combinatorial algorithms, parallel algorithms, DNA computing, bioinformatics, and genetic algorithms.



Abbas Nowzari-Dalini is an Associate Professor of School of Mathematics, Statistics and Computer Science, University of Tehran. He is currently the Head of Department of Computer Science in this school. His research interest includes combinatorial algorithms, parallel algorithms, DNA computing, bioinformatics, neural networks, and computer networks.

Mohammad Ganjtabesh is a Ph. D. student in the Department of Computer Science, School of Mathematics, Statistics and Computer Science, University of Tehran, under supervision of Dr. H. Ahrabian. His research interest includes combinatorial algorithms, DNA computing, bioinformatics, and genetic algorithms.