International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 3 (2011) pp. 306 -313
© MIR Labs, www.mirlabs.net/ijcisim/index.html

A Molecular Algorithm for Longest Path Problem

H.Ahrabian[1], M.Ganjtabesh[2] and A. Nowzari-Dalini[3]

[1] Center of Excellence in Biomathematics, School of Mathematics, Statistics, and Computer Science, University of Tehran, Tehran, Iran
ahrabian@ut.ac.ir

[2] Center of Excellence in Biomathematics, School of Mathematics, Statistics, and Computer Science, University of Tehran, Tehran, Iran
mgtabesh@ut.ac.ir

[3] Center of Excellence in Biomathematics, School of Mathematics, Statistics, and Computer Science, University of Tehran, Tehran, Iran
nowzari@ut.ac.ir

Abstract: The field of DNA computing has recently attracted considerable attention. Because of its great capacity to conduct parallel relations, many NP-complete problems are solved by this approach. In this paper we present a molecular algorithm to solve the Longest Path Problem. Till now, no molecular algorithm is presented for this problem in the literatures on the weighted graph *G*=(*V*, *E*). The proposed molecular algorithm can be performed in O($|V|^2$) molecular operations. Special effort is spent on designing an scaling method for weight values in order to obtain an appropriate encoding for the problem. The effectiveness of this algorithm is verified by the computational simulation.

***Keywords*: DNA Computing, Genetic Algorithm, Graph Theory, NP-Completeness.**

I. Introduction

Since the publication of Adleman's original paper [1], several authors have described how various models of computation may be simulated using bio-molecular methods [14], [15]. The vast parallelism, exceptional energy efficiency, and extra ordinary information density inherent in molecular computation have raised the possibility that molecular computation might some day prove capable of attacking problem that have resisted conventional methods [2]-[4], [7], [12], [15]. With regard to these advantages, a major goal of subsequent research is how to use DNA manipulations to solve NP-Complete problems.
The Traveling Salesman Problem (TSP) is an NP-Complete problem which is finding a simple path of length $|V|-1$ with minimum cost between two specified vertices in the given weighted graph *G* = (*V*, *E*). Different DNA computing solutions are given for this problem in the literatures [1], [14], [16]. Manipulating the paths in this problem is rather simple since all the paths should have the same length (each vertex is seen only once in each path). The given molecular algorithm for TSP can be also employed to solve the shortest path problem, even though this problem has polynomial time algorithm with electronic computers [16].
Another interesting problem in graph theory is the Longest Path Problem (LPP). This problem is finding a simple path from a source vertex v_s to a target vertex v_t with maximum weight in a given weighted graph *G* = (*V*, *E*). We need to include the requirement of simplicity for paths; otherwise by repeatedly traversing the cycles, paths with arbitrary large weight can be created. We know that finding the longest simple path between two vertices is NP-Complete (even for un-weighted graphs). The only molecular algorithm for this problem is presented in [12] which is designed for un-weighted graphs with the time complexity O($|V|^3$). Till now, no other molecular algorithm is given for longest path problem in general form. In this paper, we design a molecular algorithm to solve the longest path problem for a weighted graph in polynomial-time complexity. We present an encoding scheme for representing a graph (vertices and edges) similar to the encoding given by Lee et al. [13]. Based on this encoding, we present a new molecular algorithm for solving the LPP for a given weighted graph *G* = (*V*, *E*) with O($|V|^2$) molecular operations, including the DNA sequences construction.
This paper is organized as follows: In Section II, we present the encoding scheme for solving the LPP. Section III describes our molecular algorithm. The generation of optimal DNA sequences is presented in Section IV. Sections V and VI provide the experimental results and conclusion, respectively.

II. Encoding scheme

As mentioned, the LPP is an NP-Complete problem and is to find a simple path from the vertex v_s (source vertex) to the vertex v_t (target vertex) considering the maximum weight in a given weighted graph *G* = (*V*, *E*). This is an optimization problem and formally we can write it as:

$$\max \sum_{e_{ij} \in P} w(e_{ij})$$

where *P* is a path from v_s to v_t and $w(e_{ij})$ shows the weight of the edge e_{ij} which is appeared in the path *P*. Figure 1 shows an instance of a graph that has six vertices. Let the source vertex be *1* and the target vertex be *6*. The path with maximum weight is $1 \rightarrow 4 \rightarrow 6$ and the weight of this path is *100100*. Note that there is no need to have |*V*| vertices in the maximum weighted path.

We now explain an encoding scheme for solving LPP by molecular algorithm. We present an encoding scheme which is very similar to the encoding given by Lee et al. in [13]. To implement the fixed length, we use a melting temperature control encoding method. This method uses fixed-length DNA strands and represents weights in the graph by melting temperatures of the given DNA strands. The basic idea is to design the sequences corresponding to the weights in the graph, such that the DNA strands for higher-weighted values have lower melting temperatures than those for lower weighted values [5], [11]. Several empirical methods are proposed to calculate the melting temperature. A classical method is the GC-content method which uses the content of guanine (G) and cystine (C) in the given DNA strand as a main factor for determining the melting temperature of the strand [13]. The nucleotide Adenine (A) is always paired to the nucleotide Thymine (T) with *2* hydrogen bonds, as well as the nucleotides C and G with *3* hydrogen bonds. Because of the number of hydrogen bonds, the base pairing between C and G is stronger than the one between A and T. So in this method the DNA sequences with lower amount of GC content have lower melting temperature.

Based on the above discussion, the encoding scheme for the vertices and edges of a given weighted graph *G* = (*V*, *E*) are designed as follows. As mentioned, this encoding scheme is very similar to the encoding scheme given by Lee et al. [13] except the encoding of the vertices in the graph is rather different.

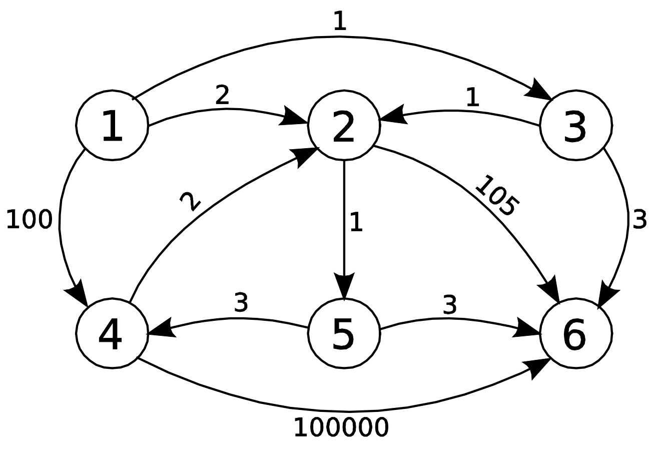


Figure. 1: A simple graph example.

1. Each weight w_i is encoded with a strand of length *2d* (*d* is a constant value proportional to the number of vertices |*V*|) and is called *weight sequence*, denoted by δ_{w_i} (if $w_i = w_j$ clearly δ_{w_i} is the same as δ_{w_j}). Since the length of each *weight sequence* is constant, therefore the number of A/T nucleotides in each sequence (AT-content) shows the value of the weight. The sequences corresponding to the higher-weighted values have more AT-content (and so less GC-content), therefore the melting temperatures of these strands are lower than the others. As it is clear, this encoding scheme can express the real-valued weights.
2. The index of vertex v_i (which is denoted by l_i) is encoded by a strand of length *2d* and is called *label sequence*, denoted by δ_{l_i} . It should be considered that the label strands in each vertex are encoded with the equal number of A/T and G/C nucleotides (i.e. *50%* for AT-content and *50%* for GC-content), such that the half right end (5'-end) of δ_{l_i} is the reverse complement of the half left end (3'-end) of δ_{l_i} ($\delta^R{}_{l_i}$ = Reverse($\bar{\delta}^L{}_{l_i}$)), where $\delta^R{}_{l_i}$ and $\delta^L{}_{l_i}$ are half right end and half left end of δ_{l_i} respectively). For example, if $\delta^L{}_{l_i}$ is equal to GACAGTT then $\delta^R{}_{l_i}$ is equal to AACTGTC. The reason for using this structure is to eliminate the cycles in the constructed paths as discussed later in the simulation of our algorithm. The restriction of using equal amount of A/T and G/C nucleotides is implied for not violating the number of A/T nucleotides in weight sequences.
3. Employing the *weight sequences* and *label sequences* constructed in steps 1 and 2, for each vertex v_i whose in-edge has weight w_k and out-edge has weight w_j , a strand of length *4d* is assigned. This sequence is called *vertex sequence* and is illustrated in below:

$\bar{\delta}^R_{w_k}$	$\delta^L_{l_i}$	$\delta^R_{l_i}$	$\bar{\delta}^L_{w_j}$

where the strands $\delta_{l_i} = \delta^L{}_{l_i} + \delta^R{}_{l_i}$ from index *d + 1* to *3d* are the *label sequence* of vertex v_i and $\bar{\delta}^R{}_{w_k}$ is the half right end of the complementary weight sequence corresponding to w_k (δ_{w_k}) and $\bar{\delta}^L{}_{w_j}$ is the half left end of the complementary weight sequence corresponding to w_j (δ_{w_j}). For the source (target) vertex, $\bar{\delta}^R{}_{w_k}$ ($\bar{\delta}^L{}_{w_j}$) is omitted. It should be mentioned that the maximum number of constructed sequences for any vertex v_i in the graph is equal to *in-degree*(v_i)×*out-degree*(v_i) = O(|*V*|^2) which imposes the construction of O(|*V*|^3) sequences for all vertices. By breaking down the vertex sequences into two subsequences, it is possible to construct O(|*V*|^2) sequences (O(|*V*|) for each vertex) and produce all the required O(|V|^3) vertex sequences. Instead of constructing the whole vertex sequence, it is sufficient to construct the subsequences $\bar{\delta}^{R}_{w_k}\ \delta^{L}_{l_i}$ and $\delta^{R}_{l_i}\ \bar{\delta}^{L}_{w_j}$ separately, and then use the complement sequence corresponding to l_i ($\bar{\delta}_{l_i}$) to connect the constructed subsequences and produce the vertex sequence. Since the length of all vertex sequences are *4d* instead of *2d* for label sequences, we can add the ligase enzyme in to the test tube, melt the sequences and use the Gel-Electrophoresis to separate the vertex sequences from the complementary label sequences.

4. Any edge e_{ij} from vertex v_i to vertex v_j with weight w_k is encoded with a strand of length *4d* and is called edge sequence such as illustrated in below:

$\bar{\delta}^{R}_{l_i}$	$\delta^{L}_{w_k}$	$\delta^{R}_{w_k}$	$\bar{\delta}^{L}_{l_j}$

where $\bar{\delta}^{R}_{l_i}$ and $\bar{\delta}^{L}_{l_j}$ are the half right end of $\bar{\delta}_{l_i}$ and half left end of $\bar{\delta}_{l_j}$ which are the complementary label sequences of vertices v_i and v_j, respectively.

5. Auxiliary sequences γ_i with length (*i*×*4d*)+*d* (*i*=*1*, *2*, …, *n*−*2*) are constructed for employing in the algorithm. The first *d* nucleotides of each sequence is the half right end of the complementary label sequence of the target vertex (i.e. $\bar{\delta}^{R}_{v_t}$) and the remaining nucleotides are arbitrary constructed with *25%* of AT-content and *75%* of GC-content for previously mentioned reason (not violating the GC-content in the weight sequences). This encoding helps us to distinguish between the strands of the same length but with different melting temperatures. More details are given during the discussion of the algorithm in the next section.

As it is mentioned, this encoding scheme for weight sequences can also express the real value weights. The method of DNA sequence generation for this encoding scheme is given in Section IV.

III. Molecular algorithm

Prior to performing our molecular algorithm, it is assumed that the required DNA sequences corresponding to the vertices and edges and their complements are generated with the method discussed in the next section. The molecular algorithm for finding the longest path between source vertex v_s and target vertex v_t in a given graph *G* = (*V*, *E*) with *n* vertices is summarized in bellow:

Algorithm DNA-LPP:

1. Pour all the strands of length *4d* corresponding to the vertices and edges (vertex and edge sequences) to the test tube T_1 with ligase enzyme. In suitable condition, hybridization and ligation occurs in the tube and double strands corresponding to all the possible paths in the graph are constructed.
2. Keep only the strands of length less than or equal to *4d*×*n*−*2d* that represent the paths of length less than or equal to *n*.
3. Keep only those paths which enter all of the vertices of the graph at most once and begin with the sequence δ_{v_s} and end with δ_{v_t} (δ_{v_s} and δ_{v_t} are the sequences corresponding to the source and target vertices, respectively).
4. Remaining strands are separated and poured in different test tube with respect to their length (in other word, with respect to the number of vertices in each path). Therefore *n*−*1* test tubes T_2, T_3, …, T_n are required for keeping the strands corresponding to the paths of length *2*, *3*, …, *n*.
5. In each test tube, strands with lowest melting temperature are kept. This means that in each test tube T_i (*2* ≤ *i* ≤ *n*), longest path with exactly *i* vertices are kept.
6. Each test tube T_i (*2* ≤ *i* ≤ *n*) contains strands with different length. By adding ligase enzyme and the auxiliary strands *i* to each test tube T_{n-i}, the strands of length *4d*×*n*−*2d* are constructed in each test tube.
7. Pour all the test tubes T_2, T_3, …, T_n into the test tube T_0.
8. Select the strands with lowest melting temperature that represent the solution strands.

In Step 1, with respect to the construction of the vertex and edge sequences, the corresponding complement sequences are hybridized and ligated and then the double strands corresponding to all the possible paths of the given graph are created.

To implement Step 2 of the algorithm, the product of Step 1 is run on a Gel- electrophoresis and the strands of length less than or equal to *4d*×*n*−*2d* are kept. The selected double stranded DNA sequences encode paths entering at most *n* vertices.

To implement Step 3 of the algorithm, the product of Step 2 is amplified by Polymerase Chain Reaction (PCR) using primers δ_{v_s} and $\bar{\delta}_{v_t}$. If the concentration of DNA sequences in the test tube is kept low enough to allow hairpin conformation, the PCR is expected to amplify only non-hairpin sequences [16]. The non-hairpin sequences represent the paths which enter all of the vertices of the graph at most once. It should be noted that, because of the structure of label sequences ($\delta_{l_i} = \delta^{L}_{l_i}$ +Reverse ($\bar{\delta}^{L}_{l_i}$)) corresponding to each vertex v_i, the existence of two or more similar label sequences in a strand, forms a hairpin (as it is illustrated in Figure 2). Thus only those sequences encoding paths which have no repeated vertex and begin with vertex v_s and end with vertex v_t are amplified.

In Step 4, the content of test tube in the Step 3 is run on a Gel-electrophoresis again. The $(8d)_{bp}$, $(12d)_{bp}$, …, $(4d \times n)_{bp}$ bands are all separated and kept in the different test tubes T_2, T_3, …, T_n. As it is mentioned, each test tube T_i corresponds to the path of length i ($2 \leqslant i \leqslant n$).

To implement the Step 5, we use the Denaturation Temperature Gradient PCR (DTG-PCR). The DTG-PCR is a modified PCR method that the denaturation temperature is started at low temperature in the beginning cycles of PCR and gradually is increased in each cycle of amplification [13]. Thus the sequences with lower melting temperature (corresponding to the higher-weighted values) will be amplified more frequently. Later, the sequences with most amount of A/T nucleotides are chosen by Temperature Gradient Gel-Electrophoresis (TGGE) in each test tube T_i ($2 \leqslant i \leqslant n$), which represent the longest path in T_i. In this case, each test tube T_i contains the strand corresponding to the longest path with exactly i vertices ($2 \leqslant i \leqslant n$). It should be noted that all the strands in one test tube have the same length but their length are different from the strands in the other test tubes. The AT-content of the constructed sequences corresponding to a path is proportional to the value of its weight. But the melting temperature of a sequence is also proportional to the length of that sequence. So, to obtain the exact solution for LPP, the strands should have the same size. For this reason in Step 6, the auxiliary strands i and the ligase enzyme are added to each test tube T_{n-i}. By providing suitable condition, primer extension is occurred and the strands of length $4d \times n - 2d$ are constructed in all the test tubes.

In Step 7, all the test tubes are merged into the test tube T_0 and eventually in Step 8, strands with lowest melting temperature can be selected to represent the solution. Note that in Step 8, the T_0 contains all the strands with the same size corresponding to the longest path of different length. Clearly, the actual longest path is a path with maximum weight which is the strand with most amounts of A/T nucleotides and low melting temperature, and this is recognized in Step 8 by using DTG-PCR and TGGE as discussed in Step 5. With respect to the above operations, the number of molecular operations in our algorithm is given in the following theorem.

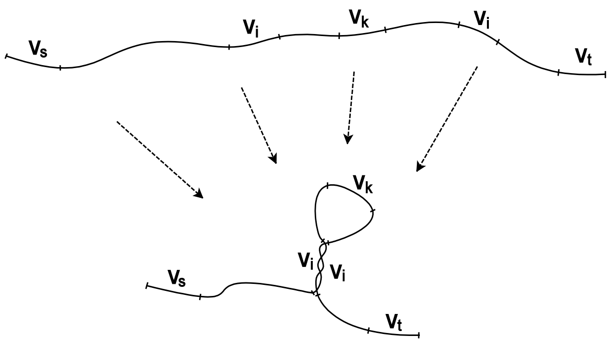


Figure. 2: Hairpin conformation.

Theorem 1. *For a given weighted graph* G = (V, E)*, the molecular algorithm DNA-LPP is performed in* $O(|V|^2)$ *molecular operations.*

Proof. As mentioned, all the required DNA sequences for our algorithm can be constructed in $O(|V|^2)$ molecular operations. Considering the molecular operations employed in the DNA-LPP algorithm, we can see that all the steps 1, 2, 5, and 8 are performed in $O(1)$ and the steps 3, 4, 6, and 7 are performed in $O(|V|)$ molecular operations. Therefore the algorithm DNA-LPP is performed in $O(|V|)$ molecular operations. By summing of the complexity of constructing the DNA sequences to the complexity of the algorithm DNA-LPP, we obtain $O(|V|^2)$ in total. □

IV. Generation of DNA Sequences

Sequence design is strongly needed for successful DNA computing. Recently, many design requirements for DNA sequences are proposed [7], [8], [9], [18]. Design requirements are divided into several aspects. Roughly speaking, the purpose of these requirements is to prevent misshybridization and undesired secondary structure, as well as keeping the uniform chemical characteristics. DNA sequence design can be considered as an optimization problem. Alternatively, DNA sequence corresponding to any encoding scheme can be generated by a genetic algorithm that minimizes the potential of error in DNA sequences for reliable molecular operations and produce reliable sequences. Deaton et al. [7], [8], [9] has explored the use of genetic algorithm for the optimization of the encoding. In this section, we describe an optimization method for generating DNA sequences corresponding to weights and labels of a given graph $G = (V, E)$ using a genetic algorithm. This genetic algorithm uses the conventional genetic operations as single point crossover and single point mutation. The fitness function is defined based on measurements which are given in [18]. We first present the genetic algorithm for producing DNA sequences and then describe the measurements in this section. The genetic algorithm is summarized in bellow:

```
1. Generate the sequences randomly with length $2d$.
2. Set $counter = 1$.
3. While ($counter <= max\_count$) do
        a. Evaluate the fitness of each sequence.
        b. Apply genetic operators (crossover and
           mutation) to produce a new population.
        c. $counter = counter + 1$.
4. Let the best codes be the fittest encodings.
```

This genetic algorithm employs a fitness function which is the summation of different functions based on five different measures which are introduced by Shin et al. [18] as follows:

$$Fitness = f_{AT-content} + f_{H-measure} + f_{3'-end} + f_{Similarity} + f_{Continuity},$$

where

- $f_{AT-content}$ is a fitness function for counting the amount of A/T nucleotides in the sequence.
- $f_{H-measure}$ is an important fitness function for preventing the mismatched hybridization.

- $f_{3'-end}$ is a fitness function for preventing hybridization at 3'-end of DNA sequences.
- $f_{Similarity}$ is a fitness function for preventing undesired hybridization by keeping the sequence as unique as possible.
- $f_{Continuity}$ is a fitness function for preventing occurrence of same bases continuity in a sequence.

Note that the best fitness value is zero; therefore the genetic algorithm tries to minimize the fitness function. The most important fitness function in this genetic algorithm is $f_{AT-content}$. This function is implemented for controlling the melting temperature in label and weight sequences. Therefore we discuss this fitness function more precisely and for the details of the other fitness functions see [18]. AT-content affects the chemical properties of DNA sequences. The AT-content measure is as follows:

$$f_{AT-content} = \sum_i (AT_{target}(x_i) - AT_{generated}(x_i))^2$$

where $AT_{target}(x_i)$ is the desired AT-content for the sequence x_i and $AT_{generated}(x_i)$ is the generated AT-content for the sequence x_i in the current population of genetic algorithm. As it is mentioned, the label sequences are encoded with equal number of A/T and G/C nucleotides. Therefore, the AT_{target} for those sequences are considered as *50%* (i.e. *d* nucleotides of type A/T where *2d* is the length of the generated sequence). The amount of AT-content in weight sequences is also optimized by this measurement. This is done so that the weight sequences with higher value have more AT-content and thus have higher probability of being content in the final solution. For this purpose, AT_{target} function is defined to promote the paths formed with higher costs and low melting temperature, so that the path with maximum cost could be found. A simple method for allocating the AT-content is to count the number of hydrogen bonds in the sequences. For example, the weight value of *20* can be encoded with *20* hydrogen bonds. However this scheme can not encode real values, as well as it requires very long DNA strands to encode the large values. Alternatively, the AT_{target} corresponding to the weight values can be estimated by the relative number of hydrogen bonds in a sequence over the entire sequences. This method is suitable for the uniformly distributed weights. In the other hand, for the weights with non uniform values, this method can not present a suitable encoding. For example, the AT_{target} for the weights *1*, *5*, and *100000* are *0*, *0*, and *20*, respectively. So from the melting temperature point of view, there is no difference between the weights 1 and 5 and the incorrect solution could be produced. In order to simultaneously handle the very small and very large weight values in the graph, we propose a new AT-content allocation method. Suppose that $W = [w_1, w_2, \ldots, w_t]$ is a sorted list of distinct weights and n_i $(1 \leqslant i \leqslant t)$ is the number of edges with weight w_i in the given graph. Before computing the AT-contents, some extra weight values are added to the list W to ensure that all the existing weight values can be decomposed with respect to the smaller values. For any weight wk, if $w_k < \sum_{i=1}^{k-1} n_i w_i$ then the value $w_k - x$ is added to the list W as an extra weight value (where x is the largest combination of weight values such that $0 < x < w_k$) and its occurrences is considered as *1*. The AT-content corresponding to the k-th weight sequence $(AT_{target}(w_k))$ can be computed as follows:

$$AT_{target}(\delta_{w_k}) = \begin{cases} w_1 & k = 1, \\ \sum_{i=1}^{k-1} n_i AT_{target}(\delta_{w_i}) + 1 & w_k > \sum_{i=1}^{k-1} n_i w_i, \\ \sum_{i=1}^{k-1} c_i AT_{target}(\delta_{w_i}) & otherwise, \end{cases} \quad (1)$$

where c_i comes from the decomposition of w_k with respect to the previous weight values ($w_k = \sum_{i=1}^{k-1} c_i w_i$, such that $0 \le c_i \le n_i$). The AT_{target} for each sequence is computed with respect to its corresponding weight value and this encoding is started from the smallest value to the largest value. As proved in the following theorem, this new scaling method helps us to correctly recognize the longest path. It should be noted that if there is no big gap between the weight values (i.e. each weight value can be decomposed with respect to the smaller weight values), then the behavior of our new scaling method is similar to the linear scaling method. As an example, our scaling method is not suitable for the weight values such as *1, 2, 4, 8, …, 2^t*, because it does not change the weight values and the scaling weights will increase exponentially. We should consider that, by using the percentage of AT-content, this problem is not avoidable for approximately consecutive weight values. For our sample graph given in Figure 1, the computed AT_{target} corresponding to weight values are shown in Table 1.

Theorem 2. *Suppose that $W = [w_1, w_2, \ldots, w_t]$ is a sorted list of weights and $N = [n_1, n_2, \ldots, n_t]$ is their corresponding number of occurrences in the given weighted graph G = (V, E). For any two collections P and Q of W, if $\sum_{w_i \in P} w_i > \sum_{w_i \in Q} w_i$ then $\sum_{w_i \in P} AT_{targer}(w_i) > \sum_{w_i \in Q} AT_{target}(w_i)$.*

Table 1. Computed AT_{target} for weight sequences.

w_i	n_i	AT_{target}
1	3	1
2	2	2
3	3	3
100	1	17
105	1	22
100000	1	56

Proof. Suppose that $W(P)=\sum_{w_i\in P} w_i$ and $W(Q)=\sum_{w_i\in Q} w_i$. Since the elements of *P* and *Q* are selected from the list *W*, so $W(P)=\sum_{i=1}^{t} p_i w_i$ and $W(Q)=\sum_{i=1}^{t} q_i w_i$, where $0\le p_i, q_i \le n_i$. Now suppose that *j* be the greatest index in the decomposition of $W(P)$ and $W(Q)$ such that $p_j \ne q_j$. There are two possibilities:

1. If $w_j > \sum_{i=1}^{j-1} c_i w_i$, then it should be the case $p_j > q_j$ (if not, the inequality does not hold), and from the Formula 1, we have $AT_{target}(w_j) > \sum_{i=1}^{j-1} n_i w_i$ and so $\sum_{w_i\in P} AT_{target}(w_i) > \sum_{w_i\in Q} AT_{target}(w_i)$.
2. If $w_j = \sum_{i=1}^{j-1} c_i w_i$, then it is possible to rewrite the decompositions by applying the following rules:
   - If $p_j > q_j$ then reduce p_j to obtain the equality $p_j = q_j$ and add the values $(p_j - q_j)c_i$ to the coefficients p_i *(1 ⩽i ⩽ j–1)*.
   - If $p_j < q_j$ then reduce q_j to obtain the equality $p_j = q_j$ and add the values $(p_j - q_j)c_i$ to the coefficients q_i *(1 ⩽i ⩽ j–1)*.

By performing the above steps for the remaining indices, we either prove the the theorem or we obtain $p_i = q_i$ *(2 ⩽i ⩽t)* and $p_1 = q_1 + \alpha$ (where α > 0). In the second case, since all the coefficients become equal (except the last one), we can write:

$$\begin{aligned}\sum_{w_i\in P} AT_{target}(w_i) &= \sum_{i=1}^{t} p_i AT_{target}(w_i)\\ &= \sum_{i=1}^{t} q_i AT_{target}(w_i) + \alpha AT_{target}(w_i)\\ &> \sum_{i=1}^{t} q_i AT_{target}(w_i)\\ &\sum_{w_i\in Q} AT_{target}(w_i)\end{aligned}$$

□

Thus, by employing the mentioned fitness, crossover, and mutation functions in the genetic algorithm, it is possible to design ideal sequences for the encoding scheme for LPP.

V. Experimental Result

We developed a tool for simulating our molecular algorithms. This tool is executed for several instances of weighted graphs with different sizes which are generated randomly. For each generated graph of size n, our algorithm found the longest path (if there is any) between two specified vertices. The steps of our algorithm for finding the longest path between the source vertex *1* and target vertex *6* for the graph shown in Figure 1 are presented in this section. First we have generated the weight and label sequences required for encoding the sample graph given in Figure 1 by the genetic algorithm discussed in previous section. The generated sequences of length *20* are shown in Table 2 and Table 3, respectively. Employing the weight sequences and label sequences shown in these tables, the vertex and edge sequences are constructed and symbolically demonstrated in Table 4 and Table 5, respectively.

By pouring vertex and weight sequences into a test tube and in a suitable condition, the hybridization and ligation are occurred in the test tube. The constructed double-stranded DNA sequences show all the possible paths with any weight in the graph. Among these sequences, the sequences beginning with vertex *1* and ending with vertex *6* which have the length less than *6×40* and have no repeated vertex are separated in a different test tube. By adding the auxiliary sequences to their corresponding test tubes, the length of all sequences become same. Among these sequences corresponding to the paths, the longest path is the sequence with most amounts of A/T nucleotides which is as follows:

$$5' - \overbrace{ATACTCTGCACCGTACCAAG}^{\delta_{l_1}}\overbrace{CTCCGGTCAAGTGACCGGAG}^{\overline{\delta}_{w_4}} -$$
$$\overbrace{TGATTCCACCAGACCATGTC}^{\delta_{l_4}}\overbrace{ATATTACTATAGTATAATAT}^{\overline{\delta}_{w_6}} -$$
$$\overbrace{GCATTCGCACCTGAACGCTA}^{\delta_{l_6}} - 3'\ .$$

This sequence encodes the path *1 → 4 → 6* which is obviously the longest path in our sample graph.

Table 2. Weight sequences.

i	w_i	AT-content %	symbol	Generated weight sequence ($5'-\delta_{w_i}-3'$)
1	1	1.79 %	δ_{w_1}	GCTGCGCGCGCGCGTCGCCG
2	2	3.57 %	δ_{w_2}	CGCCGCGCAGGGCTGGAGGG
3	3	5.36 %	δ_{w_3}	CTGCGGAGTCGCACCGGCCG
4	100	30.37 %	δ_{w_4}	GAGGCCAGTTCACTGGCCTC
5	105	39.29 %	δ_{w_5}	TGCTACCTCAATGACCGTCG
6	100000	100%	δ_{w_6}	TATAATGATATCATATTATA

Table 3. Label sequences.

Vertex label i	AT-content %	symbol	Generated label sequence ($5' - \delta_{l_i} - 3'$)
1	50 %	δ_{ℓ_1}	ATACTCTGCACCGTACCAAG
2	50 %	δ_{ℓ_2}	ACTAGTGACGAACTTACGGC
3	50 %	δ_{ℓ_3}	AGGTAGATTCAGCATACGCG
4	50 %	δ_{ℓ_4}	TGATTCCACCAGACCATGTC
5	50 %	δ_{ℓ_5}	ACGAACCAGCACATCTCAGT
6	50 %	δ_{ℓ_6}	GCATTCGCACCTGAACGCTA

Table 4. Vertex sequences.

Index	$w_k \to v_i \to w_j$	Symbolic representation	Index	$w_k \to v_i \to w_j$	Symbolic representation
1	$1 \to 2$	$\delta_{\ell_1} \overline{\delta}^L_{w_2}$	12	$4 \to 2 \to 5$	$\overline{\delta}^R_{w_2} \delta_{\ell_2} \overline{\delta}^L_{w_1}$
2	$1 \to 3$	$\delta_{\ell_1} \overline{\delta}^L_{w_1}$	13	$4 \to 2 \to 6$	$\overline{\delta}^R_{w_2} \delta_{\ell_2} \overline{\delta}^L_{w_5}$
3	$1 \to 4$	$\delta_{\ell_1} \overline{\delta}^L_{w_4}$	14	$1 \to 3 \to 2$	$\overline{\delta}^R_{w_1} \delta_{\ell_3} \overline{\delta}^L_{w_1}$
4	$2 \to 6$	$\overline{\delta}^R_{w_5} \delta_{\ell_6}$	15	$1 \to 3 \to 6$	$\overline{\delta}^R_{w_1} \delta_{\ell_3} \overline{\delta}^L_{w_3}$
5	$3 \to 6$	$\overline{\delta}^R_{w_3} \delta_{\ell_6}$	16	$1 \to 4 \to 2$	$\overline{\delta}^R_{w_4} \delta_{\ell_4} \overline{\delta}^L_{w_2}$
6	$4 \to 6$	$\overline{\delta}^R_{w_6} \delta_{\ell_6}$	17	$1 \to 4 \to 6$	$\overline{\delta}^R_{w_4} \delta_{\ell_4} \overline{\delta}^L_{w_6}$
7	$5 \to 6$	$\overline{\delta}^R_{w_3} \delta_{\ell_6}$	18	$5 \to 4 \to 2$	$\overline{\delta}^R_{w_3} \delta_{\ell_4} \overline{\delta}^L_{w_2}$
8	$1 \to 2 \to 5$	$\overline{\delta}^R_{w_2} \delta_{\ell_2} \overline{\delta}^L_{w_1}$	19	$5 \to 4 \to 6$	$\overline{\delta}^R_{w_3} \delta_{\ell_4} \overline{\delta}^L_{w_6}$
9	$1 \to 2 \to 6$	$\overline{\delta}^R_{w_2} \delta_{\ell_2} \overline{\delta}^L_{w_5}$	20	$2 \to 5 \to 4$	$\overline{\delta}^R_{w_1} \delta_{\ell_5} \overline{\delta}^L_{w_3}$
10	$3 \to 2 \to 5$	$\overline{\delta}^R_{w_1} \delta_{\ell_2} \overline{\delta}^L_{w_1}$	21	$2 \to 5 \to 6$	$\overline{\delta}^R_{w_1} \delta_{\ell_5} \overline{\delta}^L_{w_3}$
11	$3 \to 2 \to 6$	$\overline{\delta}^R_{w_1} \delta_{\ell_2} \overline{\delta}^L_{w_5}$			

Table 5. Edge sequences.

Index	$v_i \to v_j$	Symbolic representation	Index	$v_i \to v_j$	Symbolic representation
1	$1 \to 2$	$\overline{\delta}^R_{\ell_1} \delta_{w_2} \overline{\delta}^L_{\ell_2}$	7	$3 \to 6$	$\overline{\delta}^R_{\ell_3} \delta_{w_3} \overline{\delta}^L_{\ell_6}$
2	$1 \to 3$	$\overline{\delta}^R_{\ell_1} \delta_{w_1} \overline{\delta}^L_{\ell_3}$	8	$4 \to 2$	$\overline{\delta}^R_{\ell_4} \delta_{w_2} \overline{\delta}^L_{\ell_2}$
3	$1 \to 4$	$\overline{\delta}^R_{\ell_1} \delta_{w_4} \overline{\delta}^L_{\ell_4}$	9	$4 \to 6$	$\overline{\delta}^R_{\ell_4} \delta_{w_6} \overline{\delta}^L_{\ell_6}$
4	$2 \to 5$	$\overline{\delta}^R_{\ell_2} \delta_{w_1} \overline{\delta}^L_{\ell_5}$	10	$5 \to 4$	$\overline{\delta}^R_{\ell_5} \delta_{w_3} \overline{\delta}^L_{\ell_4}$
5	$2 \to 6$	$\overline{\delta}^R_{\ell_2} \delta_{w_5} \overline{\delta}^L_{\ell_6}$	11	$5 \to 6$	$\overline{\delta}^R_{\ell_5} \delta_{w_3} \overline{\delta}^L_{\ell_6}$
6	$3 \to 2$	$\overline{\delta}^R_{\ell_3} \delta_{w_1} \overline{\delta}^L_{\ell_2}$			

VI. CONCLUSION

A molecular algorithm for solving the longest path problem for a weighted graph $G = (V, E)$ is presented. Our algorithm can be performed in $O(|V|^2)$ molecular operations. In the utilized encoding scheme, the relative values of AT-content against GC-content are taken into account to handle the non uniform weights in the graph. For this reason, a new method for scaling the weights in the graph is presented. This new scaling method can be used to solve any similar problem in the weighted graphs. The algorithm is based on constructing all the possible paths in a given graph. Later the cycles are eliminated with respect to the encoding of the vertices. It should be noted that, although we are searching for a longest path, the algorithm can eliminate the cycles and so the correct longest path can be obtained.

References

[1] L.M. Adelman, Molecular computation of solutions to combinatorial problems, *Science*, 266, pp. 1021– 1024, 1994.

[2] H. Ahrabian, M. Ganjtabesh, and A. Nowzari-Dalini, DNA algorithm for an Unbounded Fan-in Boolean circuit, *BioSytems*, 82, pp. 52–60, 2005.

[3] H. Ahrabian, M. Ganjtabesh, and A. Nowzari-Dalini, Molecular solution for double and partial digest problems in polynomial time, *Computing and Informatics*, 28, pp. 1001–1020, 2009.

[4] H. Ahrabian, A. Nowzari-Dalini, F. Zare-Mirakabad, A constant time algorithm for DNA add, *International Journal of Foundations of Computer Science*, 20, pp. 549–558, 2009.

[5] E.T. Bolton and B.J. McCarthy, A general method for the isolation of RNA complementary to DNA, *Proceedings of Natural Academic Science*, 48, pp. 1390– 1397, 1962.

[6] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, McGraw-Hill, Clifford, 2001.

[7] R. Deaton, M. Garzon, J.A. Rose, D.R. Franceschetti, R.C. Murphy, and S.E. Stevens Jr., Reliability and efficiency of a DNA based computation, *Physical Review Letters*, 80, pp. 417–420, 1998.

[8] R. Deaton, R.C. Murphy, M. Garzon, D.R. Franceschetti, and S.E. Stevens Jr., Good encodings for DNA-based solutions to combinatorial problems, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 44, 247–258, 1999.

[9] R. Deaton and J.A. Rose, "Simulations of statistical mechanical estimates of hybridization error", In *Proceedings of the Sixth International Meeting on DNA Based Computers*, pp. 251–259, 2001.

[10] H. Genand and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York, 1997.

[11] M.A. Innis, D.H. Gelfand, J.J. Sninsky, and T.J. White, *PCR Protocols-A Guide to Methods and Applications*, Academic Press, San Diego, 1989.

[12] I. Katsanyi, "Solutions of some classical problems in various theoretical DNA computing models", In *Proceeding of the Second Annual Meeting of Molecular Computing Network*, pp. 27–49, 2003.

[13] J.Y. Lee, S.Y. Shin, T.H. Park, and B.T. Zhang, Solving traveling salesman problems with DNA molecules encoding numerical values, *BioSystems*, 78, pp. 39–47, 2004.

[14] R.J. Lipton, DNA solution of hard computational problems, *Science*, 268, pp. 542–545, 1995.

[15] C.C. Maley, DNA computation: theory, practice and prospects, *Evolutionary Computation*, 6, pp. 201– 229, 1998.

[16] A. Narayanan and S.Zorbalas, "DNA algorithms for computing shortest paths", In *Proceedings of the Third Annual Conference of Genetic Programming*, pp. 718–723, 1998.

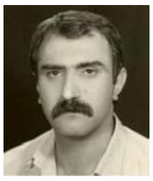
[17] J. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, and M. Hagiya, Molecular computation by DNA hairpin formation, *Science* ,288 pp. 1223–1226, 2000.

[18] S.Y. Shin, D.M. Kim, I.H. Lee, and B.T. Zhang, "Evolutionary sequence generation for reliable DNA computing", In *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 79–84, 2002.

Author Biographies

Hayadeh Ahrabian is a professor of School of Mathematics and Computer Science, University of Tehran. She is the head of Bioinformatics research group in this school. Her research interest includes bioinformatics, combinatorial algorithm, parallel algorithms, DNA computing, and genetic algorithms.

Mohammad Ganjtabesh is an assistant professor of School of Mathematics and Computer Science, University of Tehran. His research interest includes bioinformatics, parallel algorithms, DNA computing, image processing, and neural networks.



Abbas Nowzari-Dalini is an associate professor of School of Mathematics and Computer Science, University of Tehran and is currently the director of Graduate studies in this school. His research interest includes bioinformatics, combinatorial algorithm, parallel algorithms, DNA computing, neural networks, and computer networks.