Mathematical Problems of Computer Science 23, 2004, 134–143.

A Dynamic Programming Approach for Computing Similarity of the Protein Sequences Based on Continuous Functions Comparison

Robert K. Gevorgyan

Yerevan State University, Dep. of Applied Mathematics and Informatics.
e-mail robertg@ysu.am

Abstract

This paper introduces a dynamic programming approach for computing "continuous" similarity of the two protein sequences. The discrete dynamic programming method considers items of each comparable sequence independently; meantime there is a strong interrelation between them. To overcome this disadvantage a "continuous" sequence comparison method is developed. Particularly, a certain continuous function is correlated to each comparable protein sequence, and then the comparison is made between those functions. Through compressions and expansions the comparable functions are brought to the most similar representation in the meaning of a certain similarity function. By this approach the sequence comparison problem is reduced to a functional maximization problem, which is numerically solved using dynamic programming method. Finally some practical results are presented with the application of described method.

1 Introduction

The immense volume of biological sequence databases (see e.g. [3] and [5]) requires intelligent computational methods for automatically analyzing and classification of the accumulated sequence data. In the last three decades a number of approaches and methods have been developed, which are addressed for solving various problems appeared in Molecular Biology. All those methods referred to Bioinformatics (Computational Biology) discipline, though Bioinformatics considers much wider range of problems (see [4]).

Formally the biological sequence is considered as a finite word $a = a_1, a_2, ..., a_n$ of a finite alphabet $U = \{u_1, u_2, ..., u_r\}$ (i.e. $a_i \in U$, $i = 1, 2, ..., n$). For protein sequences the alphabet consists of 20 amino acids and for DNA sequences - 4 nucleotides. The comparison of biological sequences is one of the main methods for detecting biological homogeneity. The most of the sequence comparison method are based on the following fact: high similarity of the biological sequences usually implies significant functional or structural similarity (see e.g. [10]).

In the [20] are presented several methods for sequence comparison. The alignment method is one of the widely used approaches for biological sequence comparison (see [7], [20], [21], and [24]). In 1966 Levenshtein defined a distance between two sequences (so-called

edit distance), by aligning the comparable sequences (see [14]). The plausible alignment of the biological sequences reflects the evolutionary changes, which could occur in the sequences, mainly insertion, deletion and substitution. The quality of an alignment is assessed based on a similarity (distance) matrix. In the case of protein sequences it is a 20×20 dimensional matrix, which expresses degree of closeness of amino acids. The similarity (distance) matrices are calculated based on statistic data or biochemical properties of the amino acids.

In 1970 Needelman and Wunsch introduced a general algorithm for biological sequence comparison called Needelman-Wunsch algorithm (see [17]). It based on dynamic programming method. The algorithm computes distance (similarity) of two sequences and obtains the best global alignment.

The idea of dynamic programming method was developed by Wilder, then by Bellman [6], Mikhalevich [15], Moiseev [16]. Also Bellman introduced and described class of problems, which can be solved by dynamic programming method (see [6]).

Several variations of Needelman-Wunsch algorithm were developed, which solve related problems in Bioinformatics, such as semiglobal alignment or overlapped alignment (see [7], [21], [22] and [24]). Particularly Smith-Waterman algorithm was developed to find the best local alignment of comparable sequences (see [22]).

Several heuristic modifications of Needelman-Wunsch algorithm were built, such as BLAST (see [2]) and FASTA (see [18]), which are highly applicable in database search tasks (i.e. to find biologically relevant sequences in the database for a given query sequence). Hidden Markov model (HMM) based probabilistic methods are widely used for detecting certain conserved regions of a sequence (see [7] and [8]).

In this paper a continuous analogue of the alignment method is considered from Bioinformatics perspectives. Mentioned Needelman-Wunsch algorithm for discrete sequence comparison considers items of the sequences independently, meanwhile there are strong correlations between amino acids of the protein sequence (see e.g. [7]). Our goal is to overcome this shortcoming.

To take into account interrelation of amino acids in the protein sequence it is natural to correspond to each protein sequence a continuous function. The value of this function at each point will be influenced by some neighborhood of that point. Then, the comparison will be made between corresponded functions. The idea is to find the closest representation of the comparable functions by compression and expansion of some parts of them. The closeness is meant in the meaning of a similarity function.

From mathematics perspectives the idea of comparison of continuous functions based on compressions and expansions is compiled into the following functional maximization problem:

$$\sup_{\varphi(t),\psi(t)} \int_0^T S\big(a(\varphi(t)), b(\psi(t))\big)\lambda_{\varphi\psi}(t)dt, \tag{1}$$

where $a(t)$ and $b(t)$ are comparable functions, $S(x,y)$ is a similarity function, $\varphi(t)$ and $\psi(t)$ are functions that implement the warping of the functions $a(t)$ and $b(t)$ respectively, $\lambda_{\varphi\psi}(t)$ is a penalty function.

Historically, problems similar to problem (1) were considered by Kantorovich and Rubinstein (see [13]) in 1957 and by Wasserstein ([23]) in 1969. In their studies a distance between two probabilistic measures μ and ν in the n-dimensional space R^n is defined as the "length" of the shortest path along which μ can be transposed into ν, and the "path length" is calculated by the aid of a given distance function $G(x,y)$, defined on $R^n \times R^n$.

Then problem (1) was posed in Time Warping discipline (see [20]) to compare two continuous functions of a continuous argument. As a rule, time plays the role of the argument. Time Warping originated mainly from Speech Recognition discipline (see [19]) and applicable to the continuous function comparison problems, when comparable functions differ by some warping. Problems with similar structure also appeared in other discipline such as dialectology, bird's songs study, gas chromatography etc. (see [20]). In [20] is offered to apply Time Warping approach to biological sequence comparison problem.

Evidently, the first attempt of applying Time Warping approach to biological sequence comparison problem was made in [11]. Particularly procedures are described for transition from protein sequence to a continuous function and for constructing distance function. Also a heuristic algorithm is developed for solving received problem. The ideas and methods of [11] are significantly used here. In [9] is presented another heuristic method for solving problem (1), which is actually an iterative local variations method.

In the labors listed above considered problems were reduced to a functional minimization problem. Unlike them, here, proceeding from practical reasons, a functional maximization problem is considered, though general ideas remain unchanged.

In this paper problem (1) is considered for the case $\psi(t) = t$, i.e. only function $a(t)$ is subjected to compressions and expansions. An algorithm is presented for solving stated problem. The algorithm is based on dynamic programming method and the idea of the algorithm was introduced by Mikhalevich (see [15]) for solving minimization problem for additive functions, then developed by Moiseev (see [16]). Similar algorithm was applied to solve a class of speech recognition problems (see [19]). Also presented algorithm can be treated as a solution for optimal path finding problem in the graph theory (see [16]).

2 Formulation of the problem

The main disadvantage of dynamic programming algorithm in the biological sequence comparison problems is that it considers item of comparable sequences independently. Meanwhile there is a strong interdependency between amino acids of the protein sequence (see [7]).

To take into account interrelation of amino acids in the protein sequence a continuous function is corresponded to each sequence. Then the comparison is made between those functions. By compressions and expansions the comparable functions are brought to the most similar representation in the meaning of corresponding similarity function.

The first attempt for comparison two continuous functions in the context of Bioinformatics was made in [11]. Here the ideas developed in [11] are significantly employed.

Let $U = \{u_1, u_2, ..., u_r\}$ is the alphabet, $S = \{s_{i,j}\}$ is a $r \times r$ dimensional similarity matrix, where $s_{i,j} > 0$ and $a = a_1 a_2 ... a_n$ is a sequence, $a_i \in U$, $i = 1, 2, ..., n$. The schema for correspondence a function to the sequence a is the following:

0^0. *Reordering of the alphabet.* To reorder the alphabet U and correspondingly the matrix S, such that "close" elements of the U will be placed at the close positions, whereas the difference of the indices of "far" elements will be bigger. In [11] a heuristic method is offered for reordering alphabet U based on distance matrix and probabilities of appearances of amino acids in nature. In this paper a condition is given, that the reordered alphabet should satisfy. The reordering of the matrix does not affect the biological information behind it.

Let $I = (i_1, i_2, ..., i_r)$ is a permutation of numbers $(1, 2, ..., r)$, $c(I)$ is a numerical measure of goodness of the reordered alphabet $U_I = \{u_{i_1}, u_{i_2}, ..., u_{i_r}\}$ in the meaning described above

and defined as

$$c(I) = \sum_{k=1}^{r} \sum_{j=k}^{r} |i_k - i_j| \cdot s_{i_k, i_j}.$$

Our goal is to find the permutation I^*, such that

$$c(I^*) = \min_{I \in \Upsilon} c(I),$$

where Υ is the set of all permutations of numbers $(1, 2, ..., r)$. The minimization of $c(I)$ enforces the corresponding reordered alphabet to satisfy to the requirement stated above. Posed problem known as quadratic assignment problem. This problem is NP-complete and can be solved by branch and bound method (see [12]). For our further considerations we will assume that $U = U_{I^*}$, since it has the best measure of goodness. The matrix S will be rearranged accordingly.

1^0. *Transition to numerical alphabet.* Suppose $U = \{u_1, u_2, ..., u_r\}$ is the reordered alphabet according to step 0^0. By the following formula we transfer from symbolic alphabet U to numerical alphabet $V = \{v_1, v_2, ..., v_r\}$:

$$v_1 = e_1, \quad v_i = v_{i-1} + e_i s_{i,i-1}, \quad i = 2, 3, ..., r,$$

where number v_i corresponds to symbol u_i and $e_i > 0$, $i = 1, 2, ..., n$. This transferring method is adopted from [11]. Particularly in [11] $e_1 = 10$ and $e_i = 1/10$, $2 \le i \le r$.

2^0. *Transition to numerical sequence.* To receive numerical sequence $\alpha = \alpha_1, \alpha_2, ..., \alpha_n$ corresponded to sequence $a = a_1, a_2, ..., a_n$ it is necessary to substitute each element $a_i \in U$ of the sequence a with corresponding element $\alpha_i \in V$.

3^0. *Continuous function correspondence.* Let $\alpha = \alpha_1, \alpha_2, ..., \alpha_n$ is a numerical sequence ($\alpha_i \in V$, $i = 1, 2, ..., n$). To correspond a continuous function to the succession α one can interpolate points (t_i, α_i), ($t_i < t_j$, when $i < j$, $i, j = 1, 2, ..., n$). The received function $a(t)$ should satisfy to the following condition:

$$a(t) \in \left[\min_{1 \le i \le r} \{\alpha_i\} ; \max_{1 \le i \le r} \{\alpha_i\} \right] \quad t \in [t_1; t_n].$$

Particularly in [11] $t_i = i - 1$, $i = 1, 2, ..., n$ and for interpolation Newton-Aitken method was used.

Let a and b are two sequences. As described above the sequences a and b are compared based on a symmetric similarity matrix S. To compare corresponding functions $a(t)$ and $b(t)$ it is necessary to have appropriate similarity function $S(x, y)$. The function $S(x, y)$ can be obtained by lined interpolation of points (v_i, v_j, s_{ij}), $i, j = 1, 2, ..., r$. The interpolation method is adopted from [11]. More exactly, let $v_k \le x \le v_{k+1}$ and $v_l \le y \le v_{l+1}$, $1 \le k, l \le r - 1$. Then the function $S(x, y)$ is defined as follow:

$$S(x,y) = \frac{[s_{k,l+1}(v_{k+1} - x) + s_{k+1,l+1}(x - v_k)](y - v_l)}{(v_{k+1} - v_k)(v_{l+1} - v_l)} + \\ + \frac{[s_{k,l}(v_{k+1} - x) + s_{k+1,l}(x - v_k)](v_{l+1} - y)}{(v_{k+1} - v_k)(v_{l+1} - v_l)}.$$

Obviously, for any $x, y \in [v_1; v_r]$, $S(x, y) = S(y, x)$.

Let $a(t)$ and $b(t)$ are two continuous functions corresponding to given sequences $a = a_1, a_2, ...a_n$ and $b = b_1, b_2, ...b_m$ respectively. Here is assumed, that functions $a(t)$ and $b(t)$ are defined on the interval $[0; T]$, $(T > 0)$, otherwise it can be achieved by linear transformation.

Let $\Phi[p;q]$ is defined as the follow class of functions:

$$\Phi[p;q] = \left\{\omega(t) \in KC^1[p;q], 0 < \omega'(t) < +\infty, \omega(p) = p, \omega(q) = q\right\},$$

where $KC^1[p;q]$ is the piecewise smooth class of functions, defined on the interval $[p;q]$.

Now our goal is to compare functions $a(t)$ and $b(t)$. More precisely by compressions and expansions of some parts of function $a(t)$ it is tried to find the closest representation to the function $b(t)$ in the meaning of the similarity function $S(x,y)$. For that purpose the following functional is considered:

$$J(\varphi) = \int_0^T S\big(a(\varphi(t)), b(t)\big)\lambda_\varphi(t)dt, \tag{2}$$

where $\varphi(t) \in \Phi[0;T]$. The function $\varphi(t)$ is called trajectory or path. It implements compressions and expansions of the function $a(t)$. Suppose $[t_1;t_2] \subset [0;T]$ and $t \in [t_1;t_2]$. When $\varphi'(t) > 1$ the function $a(t)$ is expanded from the interval $[t_1;t_2]$ to the interval $[\varphi(t_1);\varphi(t_2)]$, when $\varphi'(t) < 1$ the function $a(t)$ is compressed from the interval $[t_1;t_2]$ to the interval $[\varphi(t_1);\varphi(t_2)]$ and for $\varphi'(t) = 1$ neither compression nor expansion take place. The compression and expansion are continuous analogue of the deletion and insertion in the discrete alignment method (for more detailed discussion see [20]).

The function $\lambda_\varphi(t)$ is a penalty function, analogue of the gap penalty of the discrete dynamic programming method (see [7], [20], [21] and [24]). Here the penalty function is defined as follows:

$$\lambda_\varphi(t) = \begin{cases} \sin\frac{\pi}{2}\varphi'(t), & \varphi'(t) \le 1 \\ \sin\frac{\pi}{2\varphi'(t)}, & \varphi'(t) > 1 \end{cases}.$$

This definition of penalty function ensures equal penalty for symmetric compression and expansion. Also if for an interval $[t_1;t_2] \subset [0;T]$ $\varphi'(t) \neq 1$, $t \in [t_1;t_2]$, then $\lambda_\varphi(t) < 1$. Otherwise $\lambda_\varphi(t) = 1$, i.e. no penalty is applied.

To achieve the closest representation of functions $a(t)$ and $b(t)$ the following functional maximization problem is posed:

$$J(\varphi) \to \sup, \qquad \varphi(t) \in \Phi[0;T]. \tag{3}$$

The function $\varphi^*(t)$, which implements the maximum of functional (2) (if exists) is called optimal trajectory (path).

Actually, the problem (3) is a classic task of the optimal control (see [1] and [16]). The classic methods for functional optimization are not applicable to problem (3), because they have long running time or require strict conditions on the derivatives of the function under the integral. Meanwhile in the practice those conditions don't take place for functions in the definition of functional $J(\cdot)$.

3 Algorithm

To numerically solve problem (3) a dynamic programming algorithm is presented. The idea of the algorithm was introduced by Mikhalevich [15], then developed by Moiseev [16].

The preliminary step of the algorithm is to construct a grid system. Let the numbers $0 = t_0, t_1, ..., t_n = T$ are chosen such that $t_i = i\tau$, $i = 0, 1, ..., n$, where $\tau = T/n$ is the step for axis t. Similarly, $x_j = jh$, $j = 0, 1, ..., m$, where $h = T/m$ is the step for axis φ. Let us

denote $P_{i,j} = (t_i, x_j)$, $1 \le i \le n-1$, $0 \le j \le m$ and $P_{0,0} = (0,0)$, $P_{n,m} = (T,T)$. The grid system is defined as

$$P = \{P_{i,j} \mid 1 \le i \le n-1,\ 0 \le j \le m\} \cup \{P_{0,0},\ P_{n,m}\}.$$

For the fixed i $(1 \le i \le n-1)$ the set of nodes $\{P_{i,j}\}_{j=0}^{m}$, is called i-th level and denoted by P^i. The 0-th and n-th levels consist of points $P_{0,0}$ and $P_{n,m}$ respectively.

Let $\Phi_P[0;T] \subset \Phi[0;T]$ is the set of functions $\varphi(t)$, which are linear on the each interval $[t_i; t_{i+1}]$ and $\varphi(t_i) \in P^i$. Obviously the set $\Phi_P[0;T]$ is finite. The presented algorithm also allows to find the function $\varphi_P^*(t) \in \Phi_P[0;T]$, which implements the maximum of functional (2) over the class of functions $\Phi_P[0;T]$.

The function $l(\xi,\eta)$ is introduced as a degree of closeness between nodes $\xi = (t_i, x_j) \in P^i$ and $\eta = (t_{i+1}, x_k) \in P^{i+1}$, where $0 \le i \le n-1$, $0 \le j \le \|P^i\|$, $0 \le k \le |P^{i+1}|$ and $k = m$, when $i+1 = n$:

$$l(\xi,\eta) = \begin{cases} \int\limits_{t_i}^{t_{i+1}} S\big(a(\varphi_{\xi,\eta}(t)), b(t)\big)\lambda_{\varphi_{\xi,\eta}(t)}dt, & j < k \\ 0, & j \ge k \end{cases},$$

where

$$\varphi_{\xi,\eta}(t) = \frac{x_k(t-t_i) + x_{k+1}(t_{i+1}-t)}{\tau}.$$

Then the function $d(\xi)$, $\xi \in P$ is introduced as a degree of closeness between nodes $(0,0)$ and $\xi = (t_i, x_j)$:

$$d(\xi) = \max_{\varphi_\xi} \int\limits_0^{t_i} S\big(a(\varphi_\xi(t)), b(t)\big)\lambda_{\varphi_\xi}(t)dt, \tag{4}$$

where the maximum is taken over all functions from $\Phi_P[0;T]$ considered on the interval $[0;t_i]$. It is obvious that $d(\xi)|_{\xi=(0,0)} = 0$.

Now the problem can be reformulated as follows: to find the value of function $d(\xi)$ for $\xi = (T,T)$ and the function $\varphi_P^*(t) \in \Phi_P[0;T]$, which implements maximum in (4).

Presented dynamic programming algorithm recurrently calculates the values $d(\xi)$ for all nodes and allows to find the optimal trajectory $\varphi_P^*(t)$.

The following three steps describe the algorithm:

1.To calculate values $d(P_{i,j})$ and $\gamma_i(j)$ for $i = 1,2,..,n$, $0 \le j \le |P^i|$ and $j = m$, when $i = n$:

$$d(P_{i,j}) = \max_{0 \le k < |P^{i-1}|} \{d(P_{i-1,k}) + l(P_{i-1,k}, P_{i,j})\}, \tag{5}$$

$$\gamma_i(j) = \arg\max_{0 \le k < |P^{i-1}|} \{d(P_{i-1,k}) + l(P_{i-1,k}, P_{i,j})\}. \tag{6}$$

2. (Traceback procedure) For i-th level $(0 \le i \le n)$ to find the node $w_i \in P^i$, that the optimal trajectory passes through:

$$w_n = (T,T), \quad k_n = 1,$$

$$w_{i-1} = P_{i-1,\gamma_i(k_i)}, \quad k_{i-1} = \gamma_i(k_i), \quad i = n, n-1, ..., 1.$$

3. To obtain optimal trajectory by linear interpolation of points $(t_0; w_0)$, $(t_1; w_1)$, ..., $(t_n; w_n)$.

The following theorem takes place:

Theorem. There is a $O(nm^2)$ running time algorithm for finding maximum and optimal trajectory for the functional (2) over the class $\Phi_P[0;T]$.

Proof. Described above algorithm satisfies to conditions of the theorem. The proof is made by method of induction.

For the case $i = 1$ $d(P_{1,j})$ should be equal to $l(P_{0,0}, P_{1,j})$, since there is only one possible trajectory, which connects nodes $(0,0)$ and $P_{1,j}$, j=1,2,...,m. Indeed, by the formula (4) $d(P_{1,j}) = l(P_{0,0}, P_{1,j})$, since $d(P_{0,0}) = 0$ and $|P_{0,0}| = 1$.

Assume the formula (5) takes place for any i $(i < n)$. Let us prove the formula (5) for the case $i+1$:

$$d(P_{i+1,j}) = \max_{0 \le k < |P^i|} \{d(P_{i,k}) + l(P_{i,k}, P_{i+1,j})\}. \tag{7}$$

The optimal trajectory, which connects nodes $(0,0)$ and $P_{i+1,j}$ should necessarily have a node on the i-th level. Let k' is the index of that node. Then the $d(P_{i+1,j})$ can be represented as follows:

$$d(P_{i+1,j}) = d(P_{i,k'}) + l(P_{i,k'}, P_{i+1,j}) \tag{8}$$

and $d(P_{i+1,j}) \ge d(P_{i,k}) + l(P_{i,k}, P_{i+1,j})$ for $1 \le k < |P^i|$, which follows from definition of function $d(\xi)$. From (8) and the last inequality follows that (7) takes place.

The second step of the algorithm allows to obtain nodes $w_i \in P^i$, that the optimal trajectory passes through. The nodes w_i are recurrently obtained based on values $\gamma_i(j)$, which are calculated by the recurrence formula (6) simultaneously with $d(P_{i,j})$. The optimal trajectory $\varphi_P^*(t)$ is constructed in the third step of the algorithm. Obviously $\varphi_P^*(t) \in \Phi_P[0;T]$. The theorem is proven.

4 Practical results

To demonstrate validity of the developed method some practical results are presented. Particularly several profile hidden Markov models (HMM) (see [7] and [8]) are chosen from Pfam (version 10) profile HMM and domains database (see [5]) and then each HMM is run through SWISS-PROT (see [3]) protein sequence database to find the set sequences which belong to corresponding sequence family. From the found set 10 sequences are chosen with the highest significance and 10 sequences with the significance lower than a certain threshold (E-value = 10 threshold is used). Also the consensus sequence is generated for each HMM.

Using developed method the consensus sequence is compared against unaligned fragments of the selected sequences, which match to HMMs. It is natural to expect that our method will produce higher score for sequences with the highest significance and lower score for the less significance sequences.

Five random HMMs with moderate length are chosen from Pfam database and the described above procedure is performed for each HMM. As it is expected in the most cases the significant sequences produce higher similarity score than the less significant ones. In other words there is a certain threshold for each HMM, such that the scores of the significant sequences are higher than the threshold and the scores of the less significant sequences lower that the threshold. Exception made one HMM, for which two sequences don't stick to that regularity. Also it is worth to note that for another two HMMs the threshold is not so sharp.

In the table 1 is presented the detailed analysis of the comparison results. All scores are length normalized.

Accession number	Threshold	Exceptions	Lower	Higher
PF00857	5.4	0	-	-
PF01648	4.93	2	1	1
PF01842	5.23	0	-	-
PF05882	5.3	0	-	-
PF03554	5.6	0	-	-

Table 1. Accession number - accession numbers of the selected HMMs (or domains), Threshold - threshold, which separates high and less significant sequences, Exceptions - the number of sequences that do not keep to threshold rule, Lower - the number of high significant sequences with the score lower than the threshold, Higher - the number of less significant sequences with the score higher than the threshold.

Conclusion. Presented method gives precise solution of the problem at the scope of the model. Besides of its immediate application in sequence comparison tasks, it can be serve as a measure for heuristic modifications of the method. The "continuation" of the sequence and penalty function choice are very important steps of the method. The appropriate continuation and penalty function may significantly affect the comparison result.

5 Acknowledgement

The present investigation was supported by INTAS Young Scientist 2003 grant, reference number 03-55-2278.

References

[1] Alexeev V.M., Tikhomirov V.M. and Fomin S.V., *Optimal Control.*, Nauka, 1979.

[2] Alstchul S.F., Glish W., Miller W., Myers E.W. and Lipman D.J. *Basic local alignment search tool.* J. Mol. Biol. 215, 403-410, 1990.

[3] Bairoch A. and Apweiler R. *The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999.* Nucleic Acid Res., 27, 49-54, 1999.

[4] Baldi P. and Brunak S. *Bioinformatics, The Machine Learning Apprach.* MIT Press, 2001.

[5] Bateman A., Birney E., Cerruti L., Durbin R., Etwiller L., Eddy S.R., Griffiths-Jones S., Howe K.L., Marshall M. and Sonnhammer L.L.E. *The Pfam protein families database.* Nucleic Acids Research, vol. 30, no. 1, 276-280, 2002.

[6] Bellman R. *Dynamic Programming.* Prinston Univ. Press, 1957.

[7] Durbin R. Eddy S.R., Krogh A., Mitchison G. *Biological Sequence analysis.* Cambridge University Press, 1998.

[8] Eddy S.R. *Profile hidden Markov models.* Bioinformatics, vol. 14, no. 9, 755-763, 1998.

[9] Gevorgyan R.K., *A Continuous Method for Evaluating Dissimilarity of the Protein Sequences.* Proceedings of the ISAAC Conference on Analysis, Yerevan, Armenia, 29-40, 2004.

[10] Gusfield D. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, 1997.

[11] Heymann S., Gabrielyan O. R., Ghazaryan H. and others. *Towards a Metrical Space of Biological Sequences.* Proceedings of the ISAAC Conference on Analysis, Yerevan, Armenia, 1-18, 2004.

[12] Horst R., Pardalos P.M. and Thoai N.V. *Introduction to global optimization.* Kluwer Academic Publishers, 1995.

[13] Kantorovich L.V. and Rubinstein G.S. *On a function space and certain extremum problem.* Dokl.Akad. Nauk SSSR, N5, 115, 1058-1061, 1957.

[14] Levenstein V.I. *Binary codes capable of correcting insertions and reversals.* Sov. Phys. Dokl., 10:707-710, 1966.

[15] Mikhalevich V.S. *Sequential optimization algorithms and their applications.*, Kibernetika, N 1, 2, 1965.

[16] Moiseev N. N. *Calculus of approximations in the theory of optimal tasks.* Nauka, Moscow, 1971.

[17] Needelman S.B. and Wunsch C.D. *A general method applicable to the search for similarities in the amino acid sequences of two proteins.* J. Mol. Biol., 48, 443-453, 1970.

[18] Pearson W.R. and Lipman D.J. *Improved tools for biological sequence comparison.* Proc. Nat. Acad Sci. USA, 85, 2444-2448, 1988.

[19] Rabiner R. and Juang B.-H. *Fundamentals of speech recognition.* Prentice Hall PTR Englewood Cliffs, New Jersey 07632, 1993.

[20] Sankoff D. and Kruskal J.B. *Time Warps, String Edits and Macromolecules.* CSLI Publications, 1997, ISBN 1-57586-217-4.

[21] Setubal J.C. and Meidanis J. *Introduction to computational molecular biology.* PWS Publishing company, 1997.

[22] 21. Smith T.F. and Waterman M.S. *Identification of common molecular subsequences.* Journal of Molecular Biology, 147: 195-197, 1981.

[23] 22. Wasserstein L.N. *Markov processes over denumerable products of spaces describing large systems of automata.* Problems of Information Transission 5, 47-52, 1969.

[24] 23. Waterman M. *Introduction to computational biology.* Chapman and Hall, 1995.

Սպիտակուցային հաջորդականությունների՝ անընդհատ ֆունկցիաների վրա հիմնված նմանության հաշվարկումը դինամիկ ծրագրավորման մեթոդով:

Ռ. Կ. Գևորգյան

Ամփոփում

Այս հոդվածում ներկայացված է կենսաբանական հաջորդականությունների համեմատման մի մեթոդ: Հայտնի դիսկրետ դինամիկ ծրագրավորման մոթոդը յուրաքանչյուր համեմտվող հաջորդականությունների անդամները դիտարկում է իրարից անկախ, այնինչ դրանց միջև կան որոշակի կապեր: Այդ թերությունը հաղթահարելու համար դիտարկվում է հաջորդականությունների համեմատման մի 'անընդհատ' մեթոդ: Այն է՝ յուրաքանչյուր համեմատվող հաջորդականությանը համապատասխանության մեջ է դրվում մի անընդհատ ֆունկցիա և ապա համեմատությունը կատարվում է այդ ֆունկցիաների միջև: Մեղմումների և ճգումների միջոցով համեմատվող ֆունկցիաները բերվում են ամենանման տեսքի՝ տրված նմանության ֆունկցիայի իմաստով: Այս մոտեցման միջոցով խնդիրը բերվում է ֆունկցիոնալի օպտիմիզացիայի խնդրի, որը թվապես լուծելու համար ներկայացված է մի դինամիկ ծրագրավորման մեթոդ: Աշխատանքում բերված են նաև որոշ պրակտիկ արդյունքներ՝ ներկայացված մեթոդի կիրառմամբ: