A Description of the Protein Structures by Evolutionary-Programmed Turing Machine

LUKAS KOURIL, ROMAN JASEK, IVO MOTYL
Department of Informatics and Artificial Intelligence
Tomas Bata University in Zlin, Faculty of Applied Informatics
nam. T. G. Masaryka 5555, 760 01 Zlin
CZECH REPUBLIC
kouril@fai.utb.cz

Abstract: Proteins belong amongst one of the essential biological structures which influence processes occurring in living organisms. The elementary characterization of proteins has a form of sequences containing amino-acids which are elements of primary protein structures. This paper is concerned with the description of primary protein structures by the evolutionary-programmed Turing machine. The description has an appearance of rules which are used for the representation of the Turing machine's program. It therefore means that it is possible to use a Turing machine for the description of original proteins on the basis of amino-acid sequences or the eventual reconstruction of the original protein if damage occurred.

Key-Words: Turing machine, Differential Evolution, evolutionary algorithm, artificial intelligence, protein structures, optimization

1 Introduction

Proteins [1, 2, 3] form an important role in all existing organisms which exist on Earth. This role is crucial for processes which are present in organisms because they are the prerequisites for living and the preservation of life. Proteins control internal and external processes of organisms, metabolism, immune system, cell structures etc. Furthermore they can also act as e.g. accelerants of biochemical processes in the form of enzymes. In short, proteins interfere with all life-based activities.

This paper deals with primary protein structures [1, 4] and their processing by Turing machines [7, 8]. Because Turing machines represent a form of computer it is necessary to prepare a program which provides any requested operations by them. It is intended to use artificial intelligence to provide a suitable program for the Turing machine. A description for using artificial intelligence for solving this task is the aim of this paper.

1.1 Essential theory of proteins

Proteins are biopolymers compounded from sequences containing amino-acids [1, 5]. Amino-acids, as parts of proteins, are molecules composed of two functional groups [1, 3]. These are amine group NH2 and carboxyl group COOH. Amino-acids can be concatenated for the purpose of originating proteins thanks to the two afore-mentioned groups. Because amine group is a base (with the ability to react with acids and neutralize them) and carboxyl group is a second end of molecule, it is possible for them to react together by a condensation process to produce peptide C-N bond [1, 6]. Joined molecules represent the polypeptide chain [1] what is a primary protein structure [1, 4]. As mentioned at the beginning of this section, the paper is concerned with processing of primary protein structures by an evolutionary-programmed Turing machine.

2 Methods

The former parts [9-11] of our research were concerned with the examination of assessing the rules for programming the Turing machine and proving proposed concepts for the evolutionary-estimating rules for solving elementary tasks [9, 10] by the Turing machine. The part of the research which is being described deals with employing artificial intelligence in the evolutionary-estimating rules for the programming of the Turing machine for solving real problem.

The following provides an introduction to the methods and their settings which are used in this part of the research.

2.1 The Turing machine

The formal definition [7] of the Turing machine is stated as 7-tuple:

ISBN: 978-1-61804-027-5

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F) \quad (1)$$

Parameters of 7-tuple (1) are following:

- Set of Turing machine's inner states Q
- Set of input symbols $\Sigma \subseteq \Gamma \setminus \{B\}$
- Set of data-tape symbols Γ
- The transition function δ
  $\delta : Q \setminus F \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
- Initial state $q_0 \in Q$
- Blank symbols $B \in \Gamma$
- Set of final states $F \subseteq Q$

In accordance with definition (1), the Turing machine has four main components.

These parts are:

- The data tape
  contains pre-defined symbols Γ and serves as input and output medium.
- Internal stack
  operates with inner states Q of the Turing machines. Here is saved the current inner state of the Turing machine.
- Operational head
  provides a way of how the Turing machine can interact with the data tape by reading/writing symbols Γ from/to the data tape and moving over the data tape.
- Transition function δ
  controls the operational head.

As can be seen, the Turing machine works in accordance with the transition function δ. The transition function can be defined as:

$$\delta(q, X) = (p, Y, D) \quad (2)$$

with parameters:

- Current inner state q
- Currently loaded (read) symbol from the data tape X
- New inner state p
- Symbol which is intended for writing to the data tape Y
- Direction of operational head movement D

Equation (2) represents the behavior of the Turing machine. It means that the Turing machine subsequently reads the data tape symbols and passes to the inner states following (2). The response of the Turing machine is a recording of the new symbol to the data tape, transition to the new inner state and movement[1] of the head. The recorded symbol, new inner state and direction of the movement depend on the rules of the Turing machine.

The rules represent parameters of the transition function (2). In accordance with the rules (and the transition function), the Turing machine is controlled. It can be said that the program of the Turing machine is referred to a set of the rules.

Table 1 Example of sample rules

Argument of (2)		Response of (2)		
Current state	Loaded symbol	New state	Record symbol	Direction
q_1	X_1	p_5	Y_3	D_{left}
q_1	X_2	p_2	Y_1	D_{right}
q_1	X_3	p_1	Y_1	D_{right}
q_2	X_1	p_4	Y_2	D_{none}
q_2	X_2	p_1	Y_3	D_{left}

Substitution of the rules listed above in the transition function (2) will produce the following equation system:

$$\begin{aligned}
\delta(q_1, X_1) &= (p_5, Y_3, D_{left}) \\
\delta(q_1, X_2) &= (p_2, Y_1, D_{right}) \\
\delta(q_1, X_3) &= (p_1, Y_1, D_{right}) \\
\delta(q_2, X_1) &= (p_4, Y_2, D_{none}) \\
\delta(q_2, X_2) &= (p_1, Y_3, D_{left})
\end{aligned} \quad (3)$$

The first equation in (3) can be explained as: if the Turing machine reads symbol X_1 and occurs in the state q_1, its response is the recording of symbol Y_3 to the data tape, transition to the state p_5 and head movement to the left.

On the basis of the Turing machines operations (reading the symbol, moving of the head and transition to the new state in accordance with the current inner state and loaded symbol) and the first two columns of Table 1, it is perceivable that the arguments of the transition function (2) are combinations of all inner states Q and data tape symbols Γ (however it is possible to encounter rules where some combinations are not represented). In the case of transcription the rules in the form of the equation system similar to (3), the number of the equations is the product of Q and Γ.

Protein processing by the Turing machine provides 21 symbols of Γ (20 amino-acids and 1 blank symbol). Therefore it is necessary to provide supplementary variability of the Turing machine by

[1]Moving to the left, to the right or stagnation

ISBN: 978-1-61804-027-5

providing a sufficient amount of inner states (stated experimentally). On these conditions, it is a very complicated task to accurately design the rules of the Turing machine. This is a key problem for protein processing by the Turing machines.

2.2 Employing artificial intelligence

It was considered to use several methods based on artificial intelligence. The former research [9-11] we had worked on, utilized Differential Evolution [12-14] and Self-Organizing Migrating Algorithm [13-15]. These methods were mainly used for the examination of evolutionary-estimating of the rules for programming the Turing machine. The results published in this paper are based on utilizing of the Differential Evolution only. It was proved that the process of estimating the rules depends especially on the evaluative algorithm (Sect. 2.2.3), not on a concrete evolutionary algorithm.

The description of the Differential Evolution can be found here [13, 14]. Nevertheless, it is necessary to briefly mention the essential principles of Differential Evolution for the purpose of the next explanations.

2.1.1 Backgrounds of Differential Evolution

Evolutionary algorithms are inspired through natural processes. They are based on "survivance" of best adapted individuals in contrast to others. Individuals in the meaning of Differential Evolution represent parameters of one concrete solution. The ability to "survive" can be regarded as optimization where the best solution of the problem is looked for.

Before starting the evolution it is necessary to define a specimen which represents information in the form of individuals thus parameters and a range of their values. After that, a default generation is built of individuals containing parameters randomly set in accordance with the specimen. This first generation is subsequently optimized.

The principle of optimization is the evaluation of all individuals occurring in the recent generation in the sequence and the selection of them according to the cost value which represents their evaluation. This can be mathematically expressed [14] as:

$$x_{i,j}^{test} = \begin{cases} \left[x_{r_3,j}^{G} + F \cdot \left(x_{r_1,j}^{G} - x_{r_2,j}^{G}\right)\right]^{*} \\ x_{r,j}^{G} \end{cases} \quad (4)$$

$$^{*}\, if \;\; rand_j(0,1) \le CR \vee j = k$$

where

- $i = 1,...,NP$, $j = 1,...,D$ D - Dimension of individual
- $k \in \{1,...,NP\}$ k - Random index
- $r_1, r_2, r_3 \in \{1,...,NP\}$ Random selection of three individuals
- $r_1 \neq r_2 \neq r_3 \neq i$
- $CR \in \langle 0,1\rangle, F \in \langle 0,1\rangle$

$$x_i^{G+1} = \begin{cases} x_r^{test} \;\; if \;\; f_{cost}\left(x^{test}\right) \le f_{cost}\left(x_i^{G}\right) \\ x_i^{G} \end{cases} \quad (5)$$

Except for the current individual, three other individuals are randomly selected. The differential vector is originated by the subtraction of the first two randomly-selected individuals. Then the weighted differential vector is originated by mutation. The mutation is a multiplication of a mutation constant and the differential vector. As the next step, the noise vector (6) is created by the addition of a third randomly-selected individual and a weighted differential vector. The last operation is originating the test vector (4) by cross-over of the noise vector and the current individual. The test vector and the current individual are evaluated by the cost function (5) which utilizes our designed algorithm (Sect. 2.2.3). A better-evaluated individual or vector passes to the next generation where optimization repeats.

Differential Evolution requires the following set of parameters:

- Number of population(NP)
  NP expresses how many individuals occur in one generation
- Mutation constant (F)
  The constant represents diversity rate.
- Cross-over value (CR)
  CR influences originating the test vector.
- Generations (G)
  Number of generations the optimization process consist of.

Table 2 Parameters of Differential Evolution

Parameter	Value
NP	100
G	100
F	0.9
CR	0.9

The parameters were set in accordance with the observations. They are identical to former researches. There are variations of Differential

Evolution which vary in the method of noise vector equation. In this research, there was chosen DE/rand/1/bin variation. The noise vector is computed by equation (6).

$$v = x_{r_3,j}^{G} + F \cdot \left(x_{r_1,j}^{G} - x_{r_2,j}^{G} \right) \quad (6)$$

2.2.2 Encoding the rules of the Turing machine

The necessary question is how to encode rules for the Turing machine to the form of individual. We had tried many of approaches (some of them are described here [9, 10, 11]). In connection with the new evaluative algorithm (Sect. 2.2.3) a completely different way of encoding the rules was used. The individual has a form of vector containing two items only.

There is rewritten (2) to the following form:

$$\delta(q_i, X_j) = (p_k, Y_l, D_m) \quad (7)$$

where

- $q_i \in Q$ i = 1…Length of set Q
- $X_j \in \Gamma$ j = 1…Length of set Γ
- $p_k \in Q$ k = 1...Length of set Q
- $Y_l \in \Gamma$ l = 1…Length of set Γ
- $D_m \in D$ m = 1…Length of set D

$$D = \{left, none, right\}$$

Then items of the vector are:

- l-index – an index of symbol which will be written to the data tape by the Turing machine.
- m-index – an index which expresses the direction of the head movement.

It follows from the above that the rules which are encoded by individuals represent the response of the transition function with the exclusion of the inner state.

2.2.3 Evaluative algorithm

An important part of this research is an approach to evaluating individuals within the cost function. Our newly-designed evaluative algorithm (described here [11]) is utilized which enable per-partes programming of the Turing machine. The algorithm is written in F# language (as well as library[2] which implements the Differential Evolution).

During evaluation, individuals are decoded to the form of rule and brought to the Turing machine. Evaluation of individual is based on the behavior of the Turing machine which is initialized by the new rule. The evaluation consists of several conditions related to the Turing machine. These conditions are:

- If the new position of the head is out of data tape, the cost value (CV) = 4000.
- If the last three movements or symbols are same, CV = 2000.
- If the requested symbol and the new symbol produced by rule are not same, CV = 1000.
- If the head moves to the right CV = -3000.
  If the head moves to the left, CV = -2000.
  If the head stagnates, CV = -1000.
- If the symbol for writing is the blank symbol B, CV = CV + 1000.

The Differential Evolution is set to optimization by finding a global minimum thus negative CVs represent better evaluations. In the fourth condition, it is possible to set preferable directions of the head movement.

2.3 Per-partes programming

As can be seen in Sect. 2.2.2, the individual of Differential Evolution encode only one rule of the Turing machine thus the composition of complete set of rules is per-partes realized. The Differential Evolution optimizes individuals in order to estimate one rule which solves the behavior of the Turing machine for the best processing of actual symbol. It means that it is necessary to execute the optimization process several times; the one rule for processing of the current symbol is optimized once in each run of the Differential Evolution. Subsequently, the inner states of the Turing machine are assigned to estimated rules. These inner states are finally reduced, so that the whole set of rules is compacted. The reduction consists of linking rules (estimated responses of the Turing machine) so that all possible combinations of arguments of the transition function (2) are used for assigning the estimated responses.

3 Evolutionary description of proteins

In this research we have tried to obtain the Turing machines rules for processing of polypeptide 1MYM[3] that represents structural determinants of

[2] http://fsai.codeplex.com

[3] http://www.pdb.org/pdb/explore/explore.do?structureId=1MYM

the stretching frequency of CO bound to myoglobin. The polypeptide chain looks like the following:

MVLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFK
SHPETLEKFDRVKHLKTEAEMKASEDLKKHGVTVL
TALGAILKKKGHHEAELKPLAQSHATKHKIPIKYL
EFISEAIIHVLHSRHPGNFGADAQGAMNKALELFR
KDIAAKYKELGYQG

3.1 Settings of the Turing machine

There are considerable questions as to how to set parameters of the Turing machine. It is based on formal definition (1). Because our approach to per-partes programming adjusts inner states automatically, it is not required to specify a set of inner states Q, initial state q_0 and sets of final states F. Other parameters are:

- Set of input symbols (representing 20 amino-acids)
$$\Sigma = \begin{Bmatrix} A,C,D,E,F,G,H,I,K,L,M,N, \\ O,Q,R,S,T,V,W,Y \end{Bmatrix}$$
- Set of data tape symbols
$$\Gamma = \begin{Bmatrix} \#,A,C,D,E,F,G,H,I,K,L,M, \\ N,O,Q,R,S,T,V,W,Y \end{Bmatrix}$$
- Blank symbol $B = \#$
- Default head position (located at first input symbol)

Our goal is to reveal the rules which describe or reconstruct 1MYM polypeptide from a random sequence of amino-acids. The random sequence we used is the following:

FMYFQDLLDSSRMPVSDEDTQKIVKFMPADFIHYH
MHCPPERCAMWRNMKKPFLISWDQTYCFKTIQMSP
TETDFPLEQKMKCRKDNENMVQINESRIPKHYNPD
INLCHARDASVMWNAMPYLVYPSTYNRKHFMSGGI
QVIHLSQYYWYHFQ

Turing machine whose initialization parameters are mentioned above was afterward per-partes programmed.

3.2 Results

The per-partes programming process of the Turing machine was executed repeatedly. Several sets of rules for programming the Turing machine were successfully estimated. These rules differ to each other in a way for processing amino-acids. It means the processing is successful but the behavior of the Turing machine is different during initialization by using different rules (see where analyses of three sets of obtained rules are presented). Further difference is the amount of inner states. The average number of inner states is about 45 states.

As can be seen (Fig. 1), in the case of the first set of rules, the Turing machine moves the head to the left 67 times, to the right 223 times. The head stagnates 14 times. The second set of rules provides head movement in the left direction 60 times, to the right 213 times and stagnation 14 times. The last figured set of rules pursues moving the head in the left direction 69 times, to the right 222 times and stagnation 11 times.

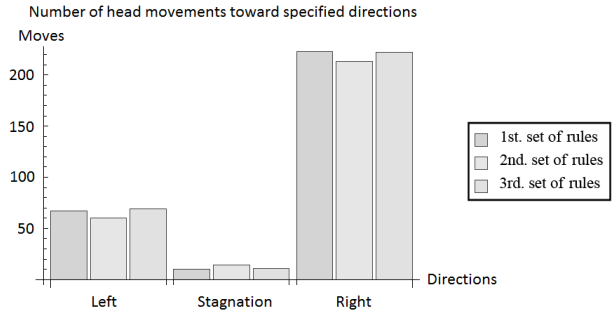

Fig. 1 Number of head movements toward specified directions

The following graphs represent progress of head movement during processing amino-acids. The processes are different although 1MYM polypeptide description by the Turing machine is successful in either case.

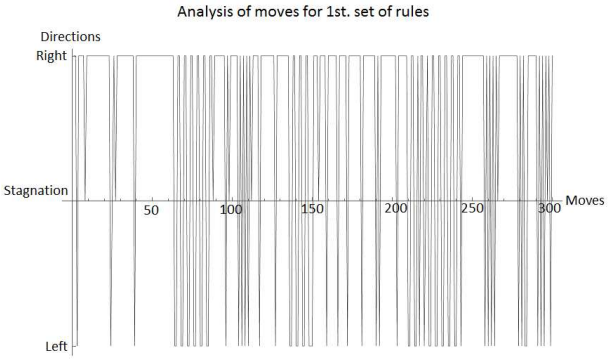

Fig. 2 Analysis of moves for 1st set of rules

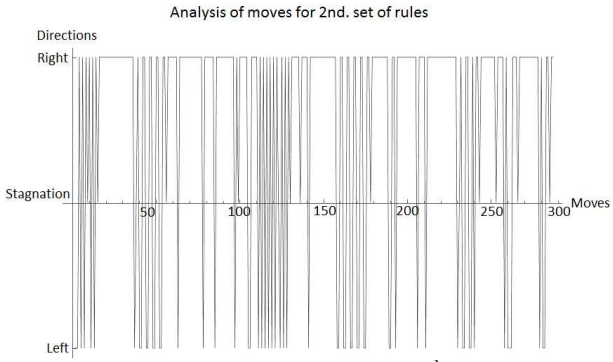

Fig. 3 Analysis of moves for 2nd set of rules

ISBN: 978-1-61804-027-5

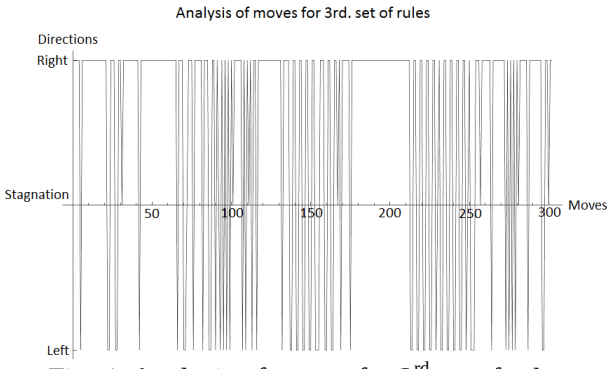

Fig. 4 Analysis of moves for 3rd set of rules

4 Conclusion

In this research we used findings based on results of previous researches for programming the Turing machine for solving complex tasks in the field of processing of polypeptide chains. We had chosen 1MYM polypeptide as a species of amino-acid sequences. The goal was to reveal the rules which allow the processing of random amino-acids in the form of 1MYM polypeptide by the Turing machine. As presented, this complex task was successfully accomplished. It succeeded in obtaining several sets of rules which differ in behavior to the Turing machine during processing amino-acids. Analyses of three sets were presented in this paper.

In further research we will try to optimize actual approaches to evolutionary programming of the Turing machine in order to process proteins. Also we will try to explore other approaches to solving this problem.

Acknowledgements: This reseach is the part of the project "Evolutionary Synthesis of Biomolecular Structures", no. IGA/42/FAI/10/D, supported by the Internal Grant Agency of Tomas Bata University in Zlin.

References:

[1] M. Zvelebil, J. O. Baum., *Understanding Bioinformatics*, Garland Science, Taylor & Francis Group, LLC, 2008. ISBN 0-8153-4024-9.

[2] S. A. Doebler, The Dawn of the Protein Era, *BioScience*, Vol. 50, No. 1, 2000, pp. 15-20. American Institute of Biological Sciences.

[3] T. A. Holme, *Proteins*, In: J. J. Lagowski (ed) Chemistry: Foundations and Applications, Vol. 4, 2004, pp. 34-38. Macmillan Reference USA. ISBN 0-02-8659319.

[4] E. S. Roberts-Kirchhoff, *Primary Structure*, In: J. J. Lagowski (ed) Chemistry: Foundations and Applications, Vol. 4, 2004, pp. 32. Macmillan Reference USA. ISBN 0-02-8659319.

[5] T. A. Holme, *Amino Acid*, In: J. J. Lagowski (ed) Chemistry: Foundations and Applications, Vol. 1, 2004, pp. 44-45. Macmillan Reference USA. ISBN 0-02-8659319.

[6] A. Schwabacher, *Peptide Bond*, In: J. J. Lagowski (ed) Chemistry: Foundations and Applications, Vol. 3, 2004, pp. 227. Macmillan Reference USA. ISBN 0-02-8659319.

[7] J. E. Hopcroft, R. Motwani, J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, 2nd s.l, Pearson Education, 2000. ISBN 0-201-44124-1.

[8] J. Peterka, *Turing machine*, In: Archiv clanku a prednasek Jiriho Peterky. http://www.earchiv.cz/a94/a432c120.php3. Cited 5 Jun 2011

[9] L. Kouril, I. Zelinka, Evolutionary Synthesis of Rules for Programming the Turing Machine, *Odborny vedecky casopis Trilobit*, No. 2, 2010, http://trilobit.fai.utb.cz/evolutionary-synthesis-of-rules-for-programming-a-turing-machine. ISSN 1804-1795.

[10] L. Kouril, I. Zelinka, Evolutionary-Estimated Programming the Turing Machine by Differential Evolution, *16th International Conference on Soft Computing MENDEL 2010*, pp. 41-48. Brno, Czech Republic. ISBN 978-80-214-4120-0.

[11] L. Kouril, I. Zelinka, An Evaluative Algorithm for Per-Partes-Programming the Turing Machine, *17th International Conference on Soft Computing MENDEL 2011*, pp. 30-37. Brno, Czech Republic. ISBN 978-80-214-4302-0.

[12] J. Lampinen, I. Zelinka, *Mechanical Engineering Design Optimization by Differential Evolution*, In: New Ideas of Optimization. 1st London, McGraw-Hill, 1999. ISBN 007-709506-5.

[13] I. Zelinka, *Umela intelligence v problemech globalni optimalizace*. Praha, BEN – technicka literature, 2002. ISBN 80-7300-069-5.

[14] I. Zelinka, Z. Oplatkova, M. Seda, P. Osmera, F. Vcelar, *Evolucni vypocetni techniky – principy a aplikace*. Praha, BEN – technicka literature, 2008. ISBN 80-7300-218-3.

[15] I. Zelinka, *SOMA – Self-Organizing Migrating Algorithm*, In: G. Onwubolu (auth), B. V. Babu (auth) New Optimization Techniques in Engineering, Springer-Verlag, 2004. ISBN 3-540-20167X.