The implementation of graphodynamic paradigm using the metagraph approach

Yuriy E. Gapanyuk
Computer Science and Control Systems Department
Bauman Moscow State Technical University
Moscow, Russia
gapyu@bmstu.ru

Yuriy T. Kaganov
Computer Science and Control Systems Department
Bauman Moscow State Technical University
Moscow, Russia
kaganov.y.t@bmstu.ru

Georgiy I. Revunkov
Computer Science and Control Systems Department
Bauman Moscow State Technical University
Moscow, Russia
revunkov@bmstu.ru

***Abstract*—This paper proposes an approach for implementation of graphodynamic paradigm using complex graphs. The metagraph model is used as complex graph model. The brief history of metagraph model development is discussed. It is shown that proposed version of the metagraph model provides the implementation of emergence principle using metavertices. Metavertices include vertices, edges and lower-level metavertices. Metaedges are used for process description. The metagraph and hypergraph models comparison is given. It is shown that the hypergraph model does not fully implement the emergence principle. The metagraph and hypernetwork models comparison are given. It is shown that the metagraph model is more flexible then hypernetwork model. Metagraph agents provide dynamical part of graphodynamic paradigm.**

***Keywords*—graphodynamic paradigm, metagraph, metavertex, metaedge, hypergraph, hypernetwork, metagraph agent**

I. Introduction

The graphodynamic paradigm was proposed by Professor Vladimir Golenkov with colleagues in monography [1]. Nowadays the ideas of graphodynamics are widely used in intelligent systems.

The graphodynamic paradigm assumes the following provisions:

- The graph-based model is used as a data model.
- The ways for graph-based data model transformation should be considered.

Currently, complex graph models are used increasingly instead of plain graph models.

The main idea of this paper is the combination of graphodynamic paradigm with complex graph model. We propose to use the metagraph model as a data model. The metagraph agents are used for model transformation.

II. The metagraph model

A. The brief history of metagraph model

At present time there is no single version of metagraph model. There are several "complex graphs with emergence" models that are similar in the basic provisions, but differ in details.

The original version of metagraph model (and term "metagraph") was proposed by A. Basu and R. Blanning in their monography [2].

The terms "metavertex" and "metaedge" were proposed in paper [3]. According to this model, metavertex is a set of vertices (which is isomorphic to the hyperedge of hypergraph). An edge connects two vertices while metaedge connects vertex and metavertex or two metavertices.

The model [4] (presented at OSTIS-2015) used term "metavertex" in the sense of model [3] for fuzzy knowledge-bases representation.

Our paper [5] also used term "metavertex" and "metaedge" but in a different sense compared to [3]. The definition of metavertex is recursive and metavertex may include vertices, edges and other metavertices.

In our model, the metavertex is used for complex data description while metaedge is used for process description. The set of metagraph agents are used for model transformation.

In the following sections we will describe our model in details.

B. The proposed version of metagraph model

The metagraph is described as follows: $MG = \langle V, MV, E, ME \rangle$, where MG – metagraph; V – set of metagraph vertices; MV – set of metagraph metavertices; E – set of metagraph edges; ME – set of metagraph metaedges.

A metagraph vertex is described by the set of attributes: $v_i = \{atr_k\}, v_i \in V$, where v_i – metagraph vertex and atr_k – attribute.

A metagraph edge is described by set of attributes, the source and destination vertices (or metavertices) and edge direction flag: $e_i = \langle v_S, v_E, eo, \{atr_k\} \rangle, e_i \in E, eo = true|false$, where e_i – metagraph edge; v_S – source vertex (metavertex) of the edge; v_E – destination vertex (metavertex) of the edge; eo – edge direction flag ($eo = true$ – directed edge, $eo = false$ – undirected edge); atr_k – attribute.

The metagraph fragment is defined as $MG_i = \{ev_j\}, ev_j \in (V \cup E \cup MV \cup ME)$, where MG_i – metagraph fragment; ev_j – an element that belongs to union of vertices, metavertices, edges and metaedges.

The metagraph metavertex: $mv_i = \langle \{atr_k\}, MG_j \rangle, mv_i \in MV$, where mv_i – metagraph metavertex belongs to set of metagraph metavertices MV; atr_k – attribute, MG_j – metagraph fragment.

The metagraph metaedge: $me_i = \langle v_S, v_E, eo, \{atr_k\}, MG_j \rangle, me_i \in ME, eo = true|false$, where me_i – metagraph metaedge belongs to set of metagraph metaedges ME; v_S – source vertex (metavertex) of the metaedge; v_E – destination vertex (metavertex) of the metaedge; eo – metaedge direction flag ($eo = true$ – directed metaedge, $eo = false$ – undirected metaedge); atr_k – attribute, MG_j – metagraph fragment.

C. The examples of proposed metagraph model

The example of metavertices representation is shown in figure 1.

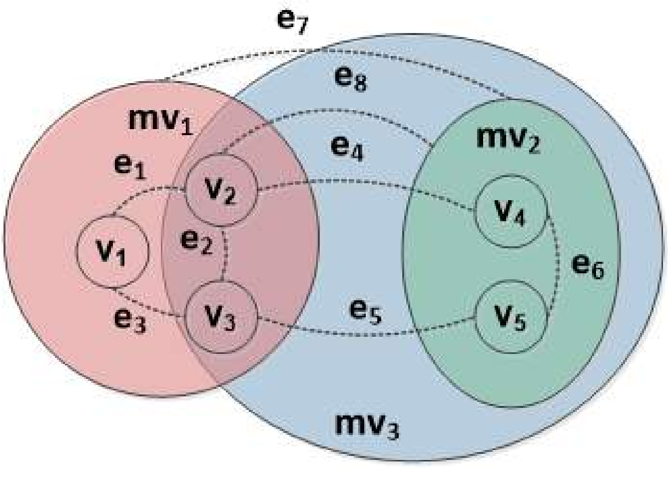

Figure 1. The example of metavertices representation.

This example contains three metavertices: mv_1, mv_2 and mv_3. Metavertex mv_1 contains vertices v_1, v_2, v_3 and connecting them edges e_1, e_2, e_3. Metavertex mv_2 contains vertices v_4, v_5 and connecting them edge e_6. Edges e_4, e_5 are examples of edges connecting vertices $v_2 - v_4$ and $v_3 - v_5$ respectively, and are contained in different metavertices mv_1 and mv_2. Edge e_7 is an example of an edge connecting metavertices mv_1 and mv_2. Edge e_8 is an example of an edge connecting vertex v_2 and metavertex mv_2. Metavertex mv_3 contains metavertex mv_2, vertices v_2, v_3 and edge e_2 from metavertex mv_1 and also edges e_4, e_5, e_8 showing the complex nature of the metagraph structure.

Thus a metavertex in addition to the attributes includes a fragment of the metagraph. The presence of private attributes and connections for a metavertex is distinguishing feature of a metagraph model. It makes the definition of metagraph holonic – a metavertex may include a number of lower level elements and in turn, may be included in a number of higher level elements.

From the general system theory point of view, a metavertex is a special case of the manifestation of the emergence principle, which means that a metavertex with its private attributes and connections become a whole that cannot be separated into its component parts.

The figure 1 helps us to show differences between metagraph model [3] and our model.

In sense of model [3] edges cannot be included in metavertex. In our model metavertex may include both vertices (metavertices) and edges.

Also in sense of model [3] edges e_7 (connecting two metavertices) and e_8 (connecting vertex and metavertex) are metaedges. In our model metaedge is used for process description. The example of metaedge is shown in figure 2.

The directed metaedge contains metavertices $mv_S, \dots mv_i, \dots mv_E$ and connecting them edges. The source vertex contains a nested metagraph fragment. During the transition to the destination vertex the nested metagraph fragment became more complex, new vertices, edges, and metavertices are added. Thus, metaedge allows binding the stages of nested metagraph fragment development to the steps of the process described with metaedge.

III. The comparison of metagraph model and other complex graph models

Currently, there are two well-known complex graph models exist: hypergraph model and hypernetwork model. In this section we will compare these models with the metagraph model.

A. The metagraph and hypergraph models comparison

Hypergraph definition according to [6]: $HG = \langle V, HE \rangle, v_i \in V, he_j \in HE$, where HG – hypergraph; V – set of hypergraph vertices; HE – set of non-empty subsets of V called hyperedges; v_i – hypergraph vertex; he_j – hypergraph hyperedge.

A hypergraph may be directed or undirected. A hyperedge in an undirected hypergraph only includes vertices whereas, in a directed hypergraph, a hyperedge defines the order of traversal of vertices. The example of an undirected hypergraph is shown in figure 3.

This example contains thee hyperedges: he_1, he_2, and he_3. Hyperedge he_1 contains vertices v_1, v_2, v_4, v_5. Hyperedge he_2 contains vertices v_2 and v_3. Hyperedge he_3 contains vertices v_4 and v_5. Hyperedges he_1 and he_2 have a common vertex v_2. All vertices of hyperedge he_3 are also vertices of hyperedge he_1.

Comparing metagraph and hypergraph models, it should be noted that the metagraph model is more expressive than the hypergraph model. Comparing figures 1 and 3 it is possible to note some similarities between the metagraph metavertex and the hypergraph hyperedge, but the metagraph offers more

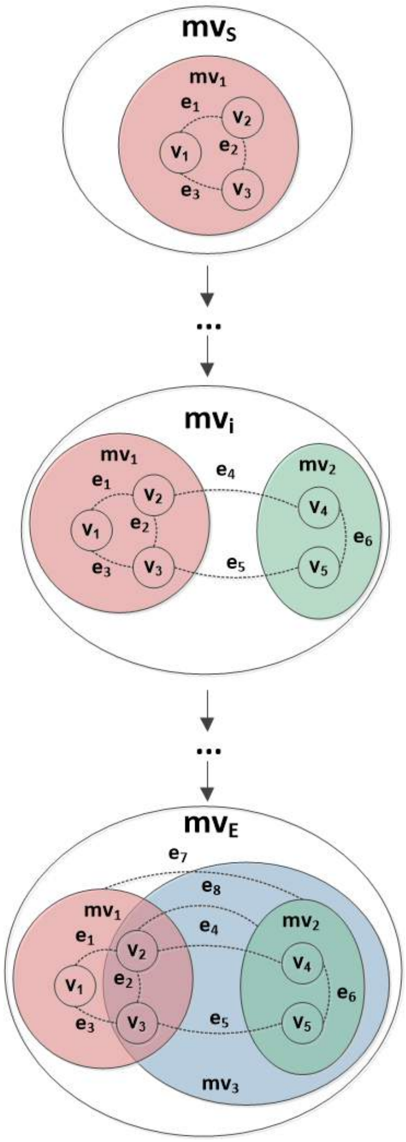

Figure 2. The example of metaedge representation.

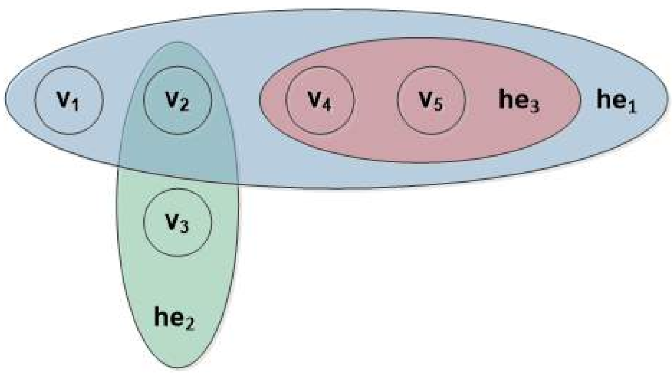

Figure 3. The example of the hypergraph.

details and clarity because the metavertex explicitly defines metavertices, vertices and edges inclusion, whereas the hyperedge does not. The inclusion of hyperedge he_3 in hyperedge he_1 is only graphical and informal, because according to hypergraph definition a hyperedge inclusion operation is not explicitly defined.

Thus the metagraph is a holonic graph model whereas the hypergraph is a near flat graph model that does not fully implement the emergence principle.

B. The metagraph and hypernetwork models comparison

Currently, there are two versions of hypernetwork model exist.

The first version of the hypernetwork model was proposed by Professor Vladimir Popkov with colleagues in 1980s. Professor V. Popkov proposes several kinds of hypernetwork models with complex formalization and therefore only main ideas of hypernetworks will be discussed in this section. According to [7] given the hypergraphs $PS \equiv WS_0, WS_1, WS_2, \dots WS_K$. The hypergraph $PS \equiv WS_0$ is called primary network. The hypergraph WS_i is called secondary network of order i. Also given the sequence of mappings between networks of different orders: $WS_K \xrightarrow{\Phi_K} WS_{K-1} \xrightarrow{\Phi_K - 1} \dots WS_1 \xrightarrow{\Phi_1} PS$. Then the hierarchical abstract hypernetwork of order K may be defined as $AS^K = \langle PS, WS_1, \dots WS_K; \Phi_1, \dots \Phi_K \rangle$. The emergence in this model occurs because of the mappings Φ_i between the layers of hypergraphs.

The second version of the hypernetwork model was proposed by Professor Jeffrey Johnson in his monography [8]. The main idea of Professor J. Johnson variant of hypernetwork model is the idea of hypersimplex (the term is adopted from polyhedral combinatorics). According to [8] a hypersimplex is an ordered set of vertices with an explicit n-ary relation and hypernetwork is a set of hypersimplices. In hierarchical system, the hypersimplex combines k elements at level N (base) with one element at level N+1 (apex). Thus, hypersimplex establishes an emergence between two adjoining levels.

The example of hypernetwork that combines the ideas of two approaches is shown in figure 4.

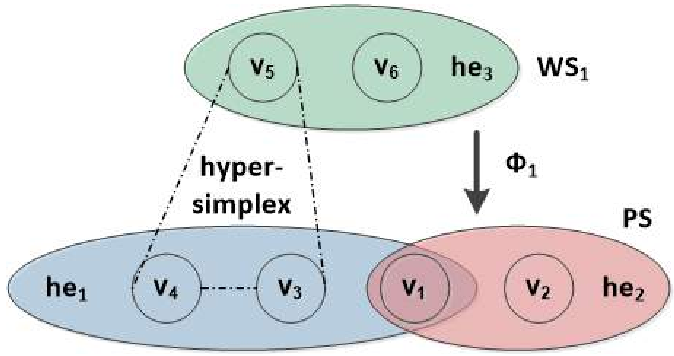

Figure 4. The example of hypernetwork.

The primary network PS is formed by the vertices of hyperedges he_1 and he_2. The first level WS_1 of secondary network is formed by the vertices of hyperedge he_3. Mapping Φ_1 is shown with an arrow. The hypersimplex is emphasized with the dash-dotted line. The hypersimplex is formed by the base (vertices v_3 and v_4 of PS) and apex (vertex v_5 of WS_1).

It should be noted that unlike the relatively simple hypergraph model the hypernetwork model is full model with emergence. Consider the differences between the hypernetwork and metagraph models.

According to the definition of a hypernetwork, it is a layered description of graphs. It is assumed that the hypergraphs may be divided into homogeneous layers and then mapped with mappings or combined with hypersimplices. Metagraph approach is more flexible. It allows combining arbitrary elements that may be layered or not using metavertices.

Comparing the hypernetwork and metagraph models we can make the following notes:

- Hypernetwork model may be considered as "horizontal" or layer-oriented. The emergence appears between adjoining levels using hypersimplices. The metagraph model may be considered as "vertical" or aspect-oriented. The emergence appears at any levels using metavertices.
- In hypernetwork model the elements are organized using hypergraphs inside layers and using mappings or hypersimplices between layers. In metagraph model metavertices are used for organizing elements both inside layers and between layers. Hypersimplex may be considered as a special case of metavertex.
- Metagraph model allows organizing the results of previous organizations. The fragments of flat graph may be organized into metavertices, metavertices may be organized in higher-level metavertices and so on. Metavertex organization is more flexible then hypersimplex organization because hypersimplex assumes base and apex usage and metavertex may include general form graph.
- Metavertex may represent a separate aspect of organization. The same fragment of a flat graph may be included in different metavertices whether these metavertices are used for modeling different aspects.

Thus, we can draw a conclusion that metagraph model is more flexible then hypernetwork model. However, it must be emphasized that from the historical point of view the hypernetwork model was the first complex graph with an emergence model and it helps to understand many aspects of complex graphs with an emergence.

IV. The Metagraph Model Transformation Using Metagraph Agents

The metagraph itself is not more than a complex data structure. To process and transform metagraph data the metagraph agents are used. There are two kinds of metagraph agents: the metagraph function agent (ag^F) and the metagraph rule agent (ag^R). Thus $ag^{MG} = ag^F | ag^R$.

The metagraph function agent serves as a function with input and output parameter in form of metagraph: $ag^F = \langle MG_{IN}, MG_{OUT}, AST \rangle$, where ag^F – metagraph function agent; MG_{IN} – input parameter metagraph; MG_{OUT} – output parameter metagraph; AST – abstract syntax tree of metagraph function agent in form of metagraph.

The metagraph rule agent uses rule-based approach: $ag^R = \langle MG, R, AG^{ST} \rangle, R = \{r_i\}, r_i : MG_j \rightarrow OP^{MG}$, where ag^R – metagraph rule agent; MG – working metagraph, a metagraph on the basis of which the rules of agent are performed; R – set of rules r_i; AG^{ST} – start condition (metagraph fragment for start rule check or start rule); MG_j – a metagraph fragment on the basis of which the rule is performed; OP^{MG} – set of actions performed on metagraph.

The antecedent of a rule is a condition over metagraph fragment, the consequent of rule is a set of actions performed on metagraph. Rules can be divided into open and closed. If the agent contains only open rules it is called open agent. If the agent contains only closed rules it is called closed agent.

The consequent of an open rule is not permitted to change metagraph fragment occurring in rule antecedent. In this case, the input and output metagraph fragments may be separated. The open rule is similar to the template that generates the output metagraph based on the input metagraph.

The consequent of closed rule is permitted to change metagraph fragment occurring in rule antecedent. The metagraph fragment changing in rule consequent cause to trigger the antecedents of other rules bound to the same metagraph fragment. But incorrectly designed closed rules system can cause an infinite loop of metagraph rule agent.

Thus metagraph rule agent can generate the output metagraph based on the input metagraph (using open rules) or can modify the single metagraph (using closed rules).

The distinguishing feature of metagraph agent is its homoiconicity which means that it can be a data structure for itself. This is due to the fact that according to definition metagraph agent may be represented as a set of metagraph fragments and this set can be combined in a single metagraph. Thus higher-level metagraph agent can change the structure of lower-level metagraph agents.

The example of metagraph rule agent is shown in figure 5.

The metagraph rule agent "metagraph rule agent 1" is represented as metagraph metavertex. According to definition, it is bound to the working metagraph MG_1, which is shown with edge e_4.

The metagraph rule agent description contains inner metavertices corresponds to agent rules (rule 1 … rule N). Each rule metavertex contains antecedent and consequent inner vertices. In given example mv_2 metavertex bound with antecedent which is shown with edge e_2 and mv_3 metavertex bound with consequent which is shown with edge e_3. Antecedent conditions and consequent actions are defined in form of attributes bound to antecedent and consequent corresponding vertices.

The start condition is given in form of attribute "start=true". If the start condition is defined as a start metagraph fragment then the edge bound start metagraph fragment to agent metavertex (edge e_1 in given example) is annotated with

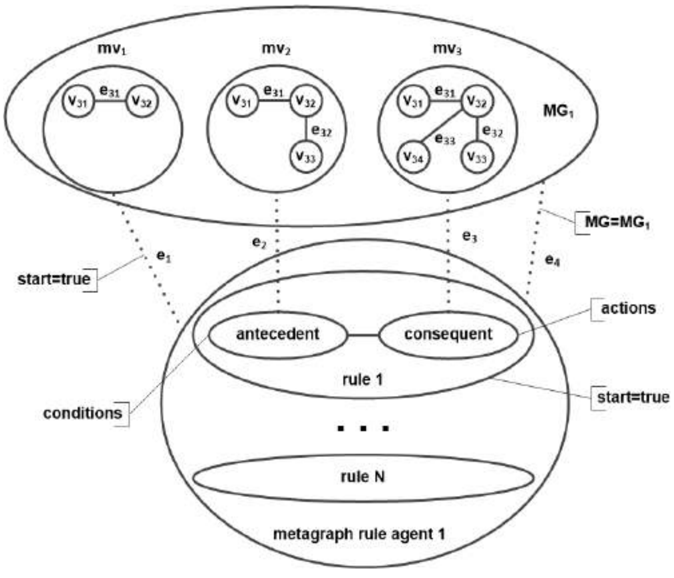

Figure 5. The example of metagraph rule agent.

attribute “start=true”. If the start condition is defined as a start rule then the rule metavertex is annotated with attribute “start=true” (rule 1 in given example), fig. 5 shows both cases corresponding to the start metagraph fragment and to the start rule.

Thus, metagraph agents provide “dynamical” part of graphodynamic paradigm.

V. Modelling the polypeptide chain synthesis for learning software

In this section, we will consider the example of metagraph approach usage for the learning software in the field of molecular biology.

Molecular biology is considered to be one of the most difficult to study topics of biological science. It’s hard to believe that the complexity of functioning of the biological cell invisible to the human eye exceeds the complexity of functioning of a large ERP-system, which can contain thousands of business processes. The difficulty of studying biological processes is also because in studying it is impossible to abstract from the physical and chemical features that accompany these processes. Therefore, the development of learning software that helps to understand even one complex process better is a valid task.

We will review the process of synthesis of a polypeptide chain which is also called “translation” in molecular biology. Translation is an essential part of the protein biosynthesis. This process is very valid from an educational point of view because protein biosynthesis is considered in almost all textbooks of molecular biology. The translation process is very complicated, and in this section, we review it in a simplified way.

The first main actor of the translation process is messenger RNA or mRNA, which may be represented as a chain of codons. The second main actor of the translation process is ribosome consisting of the large subunit and a small subunit. The small subunit is responsible for reading information from mRNA, and large subunit is responsible for generating fragments of the polypeptide chain.

According to [9] the translation process consists of three stages.

The first stage is initiation. At this stage the ribosome assembles around the target mRNA. The small subunit is attached at the start codon.

The second stage is elongation. The small subunit reads information from the current codon. Using this information the large subunit generates fragment of polypeptide chain. Then ribosome moves (translocates) to the next mRNA codon.

The third stage is termination. When the stop codon is reached, the ribosome releases the synthesized polypeptide chain. Under some conditions the ribosome may be disassembled.

From the graphodynamic paradigm point of view the translation process may be considered as a kind of graph automaton that reads codon information and generates polypeptide chain. We will use metagraph approach for translation process modelling. The representation is shown in figure 6.

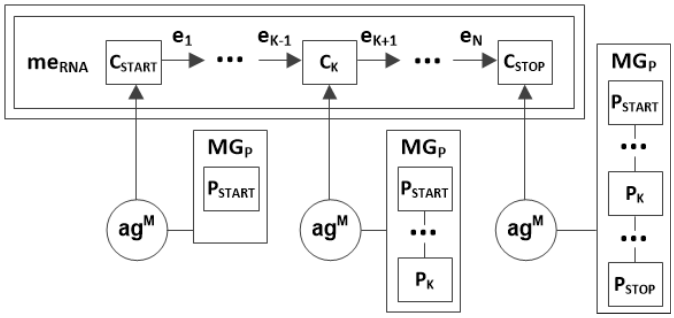

Figure 6. The representation of the translation process based on metagraph approach.

The mRNA is shown in figure 6 as metaedge $me_{RNA} = \langle C_{START}, C_{STOP}, eo = true, \{atr_k\}, MG_{RNA} \rangle$, where C_{START} – source metavertex (start codon); C_{STOP} – destination metavertex (stop codon); $eo = true$ – directed metaedge; atr_k – attribute, MG_{RNA} – metagraph fragment, containing inner codons of mRNA (C_K) linked with edges.

Codon (shown in figure 6 as elementary vertex) may also be represented as metavertex, containing inner vertices and edges according to the required level of detailing.

Ribosome may be represented as metagraph rule agent $ag^{RB} = \langle me_{RNA}, R, C_{START} \rangle, R = \{r_i\}, r_i : C_K \to P_K$, where me_{RNA} – mRNA metaedge representation used as working metagraph; R – set of rules r_i; C_{START} – start codon used as start agent condition; C_K – codon on the basis of which the rule is performed; P_K – the added fragment of polypeptide chain.

The antecedent of rule is approximately corresponds to the small subunit of ribosome modelling. The consequent of rule is approximately corresponds to the large subunit of ribosome modelling.

Agent ag^{RB} is open agent generating output metagraph MG_P based on input metaedge me_{RNA}. The input and output metagraph fragments don't contain common elements.

While processing codons of mRNA agent ag^{RB} sequentially adds fragments of polypeptide chain P_K to the output metagraph MG_P. Vertices P_K are linking using undirected edges.

The process represented in figure 6 is very higher-level. But metagraph approach allows representing linked processes with different levels of abstraction.

For example for each codon or peptide we can link metavertex with its inner representation. And we can modify this representation during translation process using metagraph agents.

Thus, metagraph approach allowed us to represent a model of polypeptide chain synthesis which can be the basis for the learning software. And this is a special case of graphodynamic paradigm.

VI. Conclusion

- The main idea of this paper is the combination of graphodynamic paradigm and complex graph model.
- As complex graph model, we propose to use metagraph model.
- The metagraph model includes vertices, edges, metavertices and metaedges.
- The proposed version of the metagraph model provides the implementation of emergence principle using metavertices. Metavertices include vertices, edges and lower-level metavertices.
- For process description metaedges are used.
- The hypergraph model does not fully implement the emergence principle.
- The hypernetwork model fully implements the emergence principle using hypersimplices. The metagraph model is more flexible then hypernetwork model.
- For metagraph model processing the metagraph function agents and the metagraph rule agents are used. Thus metagraph agents provide "dynamical" part of graphodynamic paradigm.

References

[1] Knowledge Representation and Processing in Graph-Dynamic Associative Computers. V.V. Golenkov, O.E. Eliseeva, V.P. Ivashenko and others. Edited by V.V. Golenkov. Minsk, BSUIR, 2001. 412 p.

[2] A. Basu, R. Blanning. Metagraphs and their applications. New York, Springer, 2007. 173 p.

[3] S.V. Astanin, N.V. Dragnish, N.K. Zhukovskaya. Vlozhennye metagrafy kak modeli slozhnykh ob"ektov [Nested metagraphs as models of complex objects]. Inzhenernyj vestnik Dona, 2012, vol. 23, no. 4-2 (23), p. 76.

[4] L.S. Globa, M.Y. Ternovoy, O.S. Shtogrina. Metagrafy kak osnova dlya predstavleniya i ispol'zovaniya baz nechetkih znaniy [Metagraph based representation and processing of fuzzy knowledgebases] Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems], 2015, pp. 237-240.

[5] E.N. Samokhvalov, G.I. Revunkov, Yu.E. Gapanyuk Ispolzovaniye metagraphov dlya opisaniya semantiki i pragmatiki informatsionnykh sistem [Metagraphs for information systems semantics and pragmatics definition]. Vestnik MGTU im. N.E. Baumana, seriya "Priborostroeniye" [Herald of the Bauman Moscow State Technical University, "Instrument Engineering"], 2015, no. 1, pp. 83-99.

[6] Vitaly I. Voloshin. Introduction to Graph and Hypergraph Theory. New York, Nova Science Publishers, 2009. 303 p.

[7] A.T. Akhmediyarova, J.R. Kuandykova, B.S. Kubekov, I.T. Utepbergenov, V.K. Popkov. Objective of Modeling and Computation of City Electric Transportation Networks Properties. In: International Conference on Information Science and Management Engineering (Icisme 2015), Destech Publications, Phuket, 2015, pp. 106–111.

[8] J. Johnson. Hypernetworks in the Science of Complex Systems. London, Imperial College Press, 2013. 349 p.

[9] I. Samish. Computational Protein Design. New York, Springer Science+Business Media, 2017. 450 p.

РЕАЛИЗАЦИЯ ГРАФОДИНАМИЧЕСКОЙ ПАРАДИГМЫ С ИСПОЛЬЗОВАНИЕМ МЕТАГРАФОВОГО ПОДХОДА

Гапанюк Ю.Е., Каганов Ю.Т., Ревунков Г.И.
Московский государственный технический университет имени Н. Э. Баумана, факультет "Информатика и системы управления".

В статье рассматривается реализация графодинамической парадигмы на основе сложных сетей. В качестве модели сложной сети используется метаграфовая модель. Рассматривается краткая история развития метаграфовой модели. Показано, что предложенная версия метагарфовой модели обеспечивает реализацию принципа эмерджентности с использованием метавершин. Метавершины могут включать вершины, ребра и метвершины нижнего уровня. Метаребра используются для описания процессов. Проведено сравнение моделей метаграфа и гиперграфа. Показано, что гиперграфовая модель не в полной мере реализует принцип эмерджентности. Проведено сравнение моделей метаграфа и гиперсети. Показано, что метаграфовая модель является более гибкой по сравнению с гиперсетевой моделью. Метаграфовые агенты реализуют динамическую часть графодинамической парадигмы.