The Development of the Metagraph Data and Knowledge Model

Yuriy Gapanyuk

Bauman Moscow State Technical University, 2-ya Baumanskaya ul., 5, Moscow, 105005, Russia

Abstract

Among the cognitive models in artificial intelligence, graph models of knowledge representation traditionally play an important role. Currently, models based on complex networks or complex graphs are attracting increased attention. One of the most developed models of this class is the metagraph model. Currently, one of the most developed modifications of the metagraph model is the annotated metagraph model. While the metavertices in this model are primarily intended to describe data and knowledge, the metaedges are more intended to describe processes. Thus, the metagraph model allows describing data, knowledge, and processes within a single model. In order to transform the metagraph model, the metagraph function agent and the metagraph rule agent are used. The metagraph agent allows generating one metagraph based on another (using open rules) or modify the metagraph (using closed rules).

Keywords[1]

complex network, complex graph, emergence, metagraph, metavertex, metaedge, metagraph agent.

1. Introduction

Graph models of knowledge representation traditionally play an essential role among cognitive models in artificial intelligence. Currently, models based on complex networks or complex graphs are attracting increased attention.

According to [1]: "a complex network is a graph (network) with non-trivial topological features – features that do not occur in simple networks such as lattices or random graphs but often occur in graphs modeling of real systems." The terms "complex network" and "complex graph" are often used synonymously. According to [2]: "the term 'complex network,' or simply 'network,' often refers to real systems while the term 'graph' is generally considered as the mathematical representation of a network. … The differences lay mainly in size (smaller for the graph and bigger for the network) and in the parameters and the tools employed to analyze both structures."

The most significant discrepancies are caused by the term "complex" in relation to graph models. As a rule, the term "complex" is interpreted in two ways:

Way I. Flat graphs (networks) of a huge dimension. Such networks may include millions or more vertices. The edges connecting the vertices can be non-directional or directional. Sometimes a multigraph model is used; in this case, two vertices can be connected not by one but by several edges.

It is precisely such a model in the literature that is most often called the "complex network." Studies of this model are carried out mainly by specialists in the field of mathematics. Researchers consider such parameters as the distribution of the number of links between vertices, the allocation of strongly connected subgraphs. Often, a quantitative metric is introduced for relationships, which is usually interpreted as the distance between the vertices. Dynamic models are actively investigated, in which vertices and edges are randomly added to an existing complex network. Such models are of interest in

Russian Advances in Fuzzy Systems and Soft Computing: Selected Contributions to the 10th International Conference «Integrated Models and Soft Computing in Artificial Intelligence» (IMSC-2021), May 17–20, 2021, Kolomna, Russian Federation
EMAIL: gapyu@bmstu.ru
ORCID: XXXX-XXXX-XXXX-XXXX

the study of social networks, global computer networks, various sociological and biological models. But they do not help very well in describing complex data models.

Way II. Complex graphs that use a complex description of vertices, edges, and their location. Often in such models, they refuse the flat arrangement of vertices and edges.

It is these models that can be most useful when describing complex data models.

Four such models have currently known: a hypergraph, a hypernetwork, a metagraph, and a layered network (which may be considered as a simplified version of a hypernetwork).

Currently, a single "umbrella concept" for models of this class has not yet appeared in the literature. Authors of models, as a rule, use their own names for each model, not always even pointing out the relationship of the proposed model with complex graphs (networks).

For this class of models, we can offer such an "umbrella concept" as "ensembles of complex networks (graphs)." For hypernetwork and metagraph models, the term "complex networks (graphs) with emergence" can be used since these models implement the principle of emergence, which is well known in general systems theory.

Also, quite typical is the situation when the same name is used for a family of similar models of complex networks. An example is a hypernetwork model. This model, with the same name as "hypernetwork," was independently proposed by Professor V.K. Popkov [3] and Professor J. Johnson [4]. The main difference between the two hypernetwork models is how they implement emergence. In the V.K. Popkov model, the emergence occurs because of the mappings between the adjoining levels of hypergraphs. The main idea of the J. Johnson model is the idea of hypersimplex (the term is adopted from polyhedral combinatorics). In a hierarchical system, the hypersimplex combines k elements at level N (base) with one element at level N+1 (apex). Thus, in J. Johnson's model, the hypersimplex establishes an emergence between two adjoining levels. Therefore, the concepts of the proposed options for the hypernetwork model are very similar. But, at the same time, the difference between the two versions of the models, the use of different mathematical apparatuses, and the very presentation of the material by the authors of the models convincingly indicate the absence of possible borrowings.

The history of the metagraph model is somewhat similar. Initially proposed by A. Basu and R. Blanning in 2007 [5], the model later received a number of extensions independently offered by various groups of researchers. Some of these extensions overlap. In the following sections, we will consider the history of the development of the metagraph model.

2. The Original Metagraph Model of A. Basu and R. Blanning

In the monograph by A. Bazu and R. Blanning [5, page 15], the following definitions are given that describe the metagraph model.

The ***generating set*** of a metagraph is the set of elements $X = \{x_1, x_2, \ldots, x_n\}$, which represent variables of interest, and which occur in the edges of the metagraph.

An ***edge*** *e* in a metagraph is a pair $e = \langle V_e, W_e \rangle \in E$ (where *E* is the set of edges) consisting of an invertex $V_e \subset X$ and an outvertex $W_e \subset X$, each of which may contain any number of elements. The different elements in the invertex (outvertex) are ***coinputs (cooutputs)*** of each other.

A ***metagraph*** $S = \langle X, E \rangle$ is then a graphical construct specified by its generating set *X* and a set of edges *E* defined on the generating set.

A ***simple path*** *h(x, y)* from an element *x* to an element *y* is a sequence of edges $\langle e_1, e_2, \ldots, e_n \rangle$ such that:

- $x \in invertex(e_1)$,
- $y \in outvertex(e_n)$, and
- for all $e_i, i = 1, \ldots, n-1, outvertex(e_i) \cap invertex(e_{i+1}) \neq \varnothing$.

The monograph [5] gives an example of a metagraph shown in Figure 1, for which the following set-theoretic interpretation is given:

- S = <X, E>, where

- X = {Exp, Notes, Prof, Rev, Pri, Vol, Wage}, and
- E = <{Pri,Vol},{Rev}>, <{Vol,Wage},{Exp}>, <{Rev,Exp},{Prof , Notes}>, <{Exp}, {Notes}>.

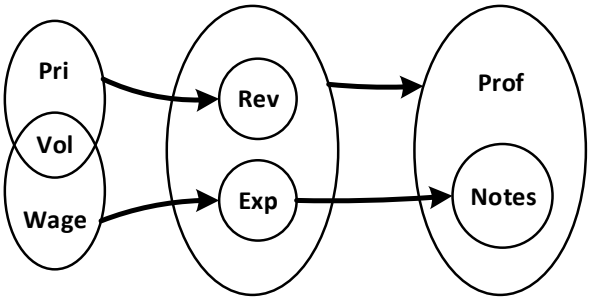

Figure 1: An example of a description of a metagraph in the interpretation of A. Bazu and R. Blanning

The emergence in the model of A. Bazu and R. Blanning is achieved through the use of edges. The concept of a metavertex is not present in this model. It can be noted that this version of the metagraph model is more suitable for describing directed processes than for describing complex graph data structures.

3. The Metagraph Model with Metavertices

The lack of a natural mechanism for describing complex graph data structures led to the appearance of extensions of the original model by A. Basu and R. Blanning. New elements appeared in the model – metavertices and metaedges. In [6], the concept of a metavertex appears, and the following definitions of a metagraph model are given.

The metagraph is a triple of sets of vertices, metaverses, and edges, respectively $S = \langle V, M, E \rangle$, where $V = \{v_r\}$ – the set of vertices of the metagraph (generating set); $M = \{m_q\}$ – the set of the metagraph metavertices; $E = \{e_h\}$ – the set of edges of the metagraph.

The metagraph metavertex $m_q = \{v_r \mid v_r \in V, r = 1, \ldots, N_{mq}\}$ is defined as a set of vertices included in a metavertex m_q, where N_{mq} is a cardinality of the set.

The following remark by the authors of the model [6] should be considered very interesting: "... if two or more metavertices correspond to the same set of vertices, then such vertices are considered identical, and only one of these metavertices is considered." We call this property of the model [6] an ***anti-annotability property***, the features of which will be considered later.

To define the edges, the authors of the model [6] introduce the concept of a ***metagraph node*** $mv \in (V \cup M)$ belonging to the combined set of vertices *V* and metavertices *M*. An edge is defined as $e_h = \langle mv_{out}, mv_{in} \rangle$, that is, characterized by outgoing and incoming nodes of a metagraph. But the use of the concept of a metagraph node for creating hierarchical metavertices is not proposed in this model.

Thus, the property of anti-annotability allows the construction of unambiguous rules but at the same time deprives the model of the depth of the hierarchy.

4. The Hierarchical Metagraph Model with Metavertices and Metaedges

In [7], not only the concept of a metavertex appears but also the concept of a metaedge and the idea of the metavertices hierarchy.

The metagraph in the model [7] is defined as $S = \langle X, X_M, E, E_M \rangle$, where X is the set of vertices of the metagraph (generating set); X_M is the set of metavertices; E is the set of edges; E_M is the set of metaedges defined on the union of metavertices and vertices $X_M \cup X$.

Thus, the metaedge in this model means an edge that can connect a vertex and a metavertex or two metavertices.

An essential feature of this model is that the authors introduce the concept of a ***nested metagraph***, which, according to the authors, is a "generalization of flat graphs, hypergraphs, and metagraphs."

In this model, the set of vertices X is considered hierarchical, and the index i is entered, which determines the nesting level of the vertex.

The property of anti-annotability is neither confirmed nor refuted by the authors of the model. At the same time, examples implicitly use a property of anti-annotability.

Thus, the central issue of this article is the description of hierarchies in the metagraph model.

5. The Annotating Metagraph Model

The annotating metagraph model proposed by us extends the ideas of the original model by A. Bazu and R. Blanning [5] and the ideas of the work [7]. The contents of the work [6] at the time of the appearance of the first version of the annotating metagraph model [8] were unknown to us.

According to [8, 9], the ***metagraph*** model may be 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.

Metagraph vertex is described by a set of attributes: $v_i = \{atr_k\}, v_i \in V$, where v_i – metagraph vertex; atr_k – attribute.

Metagraph edge is described by a set of attributes, the source, and destination vertices and edge direction flag: $e_i = \langle v_S, v_E, eo, \{atr_k\} \rangle, e_i \in E, eo = true \mid false$, 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***: $MG_i = \{ev_j\}, ev_j \in (V \cup E \cup MV \cup ME)$, where MG_i – metagraph fragment; ev_j – an element that belongs to the union of vertices, edges, metavertices, and metaedges.

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

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

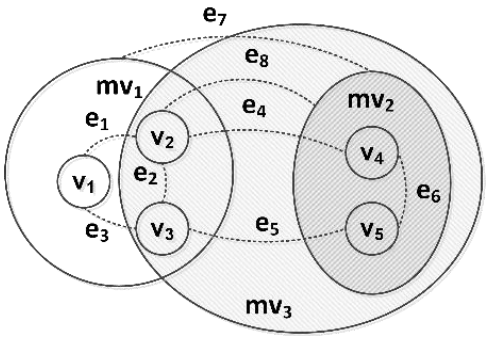

Figure 2: The example of data metagraph

The example of data metagraph (shown in Figure 2) 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 are contained in different metavertices mv_1 and mv_2. Edge e_7 is an example of the edge connecting metavertices mv_1 and mv_2. Edge e_8 is an example of the 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 holonic nature of the metagraph structure.

Note that, unlike [6], the anti-annotability property is not fulfilled in this model. The same set of vertices and edges can be included in several different metavertices, which can represent different situations and can be annotated with different attributes. This is why we called our model an ***annotating metagraph model***.

Also, unlike the previously considered models, in the model we propose, a metavertex can include both vertices and edges.

The vertices, edges, and metavertices are used for data description while the metaedges are used for process description. The metagraph metaedge: $me_i = \langle v_S, v_E, eo, \{atr_k\}, MG_j \rangle, me_i \in ME, eo = true \mid false,$ 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.

It is assumed that a metagraph fragment contains vertices (or metavertices) as process nodes and connecting them edges. A metagraph fragment can also contain nested metaedges, which makes the description of the metaedge recursive.

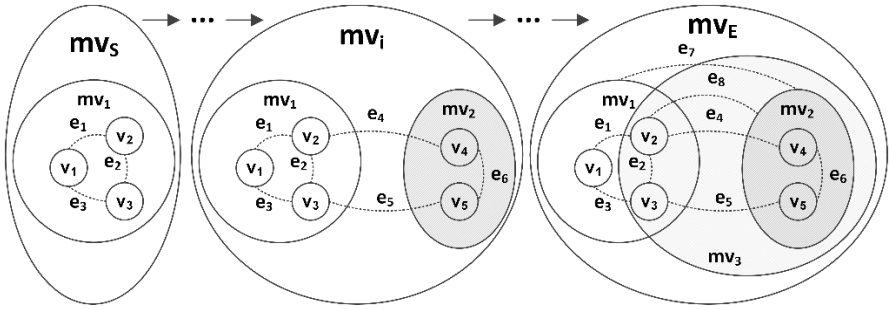

Figure 3: The example of the directed metaedge

The example of a directed metaedge is shown in Figure 3. The directed metaedge contains metavertices mv_S, … mv_i, … mv_E and connecting them edges. The source metavertex contains a nested metagraph fragment. During the transition to the destination metavertex, the nested metagraph fragment became more complex, new vertices, edges, and inner metavertices are added. Thus, metaedge allows binding the stages of nested metagraph fragment development to the steps of the process described with metaedge.

Thus, the hierarchical metaedge from the model [7] corresponds to the usual edge in our model. And a metaedge in our model means a sequence of changes of metagraph vertices.

6. Using Agents to Process Annotated Metagraph Model

The metagraph model is aimed for complex data descriptions. However, it is not aimed for data transformation. To solve this issue, the metagraph agent (ag^{MG}) aimed for data transformation is proposed. There are two kinds of metagraph agents: the metagraph function agent (ag^F) and the metagraph rule agent (ag^R). Thus, $ag^{MG} = ag^F \mid ag^R$.

The metagraph function agent serves as a function with input and output parameters in the 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 the form of metagraph.

The metagraph rule agent is rule-based: $ag^R = \langle MG, R, AG^{ST} \rangle, R = \{r_i\}, r_i : MG_j \to OP^{MG},$ where ag^R – metagraph rule agent; MG – working metagraph, a metagraph on the basis of which the rules of the 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 the rule is a condition over the metagraph fragment; the consequent of the rule is a set of actions performed on the metagraph. Rules can be divided into open and closed.

The consequent of the open rule is not permitted to change the metagraph fragment occurring in the 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 the closed rule is permitted to change the metagraph fragment occurring in the rule antecedent. The metagraph fragment changing in rule consequent causes to trigger the antecedents of other rules bound to the same metagraph fragment. But incorrectly designed closed rules systems can cause an infinite loop of metagraph rule agents.

Thus, the 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).

In order to combine the data metagraph model and metagraph agent model, we propose the concept of "active metagraph": $MG^{ACTIVE} = \langle MG^D, AG^{MG} \rangle, AG^{MG} = \{ag_i\}$, where MG^{ACTIVE} – an active metagraph; MG^D – data metagraph; AG^{MG} – set of metagraph agents ag_i, attached to the data metagraph.

Thus, active metagraph allows combining data and processing tools for the metagraph approach. Similar structures are often used in computer science. As an example, we can consider a class of object-oriented programming languages, which contains data and methods of their processing. Another example is a relational DBMS table with an associated set of triggers for processing table entries.

The main difference between an active metagraph and a single metagraph agent is that an active metagraph contains a set of metagraph agents that can use both closed and open rules. For example, one agent may change the structure of active metagraph using closed rules while the other may send metagraph data to another active metagraph using open rules. Agents work independently and can be started and stopped without affecting each other.

7. Conclusions

Graph models of knowledge representation traditionally play an important role among cognitive models in artificial intelligence. Currently, models based on complex networks or complex graphs are attracting increased attention. One of the most advanced models of this class is the meta-graph model.

The absence of a natural mechanism for describing complex graph data structures in the original model by A. Bazu and R. Blanning led to the appearance of extensions of this model due to the emergence of new elements - metavertices and metaedges.

Currently, one of the most developed modifications of the metagraph model is the annotated metagraph model. If the metavertices in this model are intended primarily for describing data and knowledge, then metaedges are primarily intended for describing processes. Thus, the metagraph model makes it possible to describe data, knowledge, and processes within a single model.

To transform the metagraph model, the metagraph function agent and the metagraph rule agent are used. The metagraph rule agent allows generating one metagraph based on another (using open rules) or modify a metagraph (using closed rules).

In order to combine the data metagraph model and metagraph agent model, the concept of "active metagraph" is proposed. The main difference between an active metagraph and a single metagraph agent is that an active metagraph contains a set of metagraph agents that can use both closed and open rules.

Thus, if the first versions of the metagraph model barely coped with the description of emergence, then modern versions provide both a description of complex nested data structures and their processing based on a multi-agent approach.

8. References

[1] B.S. Manoj, A. Chakraborty, R. Singh. Complex Networks: A Networking and Signal Processing Perspective. Pearson, New York, 2018.

[2] V. Chapela, R. Criado, S. Moral, M. Romance. Intentional Risk Management through Complex Networks Analysis. SpringerBriefs in Optimization, Springer, 2015.

[3] 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.
[4] J. Johnson. Hypernetworks in the Science of Complex Systems. London, Imperial College Press, 2013.
[5] A. Basu, R.W. Blanning. Metagraphs and Their Applications. Springer, 2007.
[6] L.S. Globa, M.Y. Ternovoy, O.S. Shtogrina. Metagraph based representation and processing of fuzzy knowledgebases. In: Proceedings of 9th Open Semantic Technologies for Intelligent Systems Conference (OSTIS-2015), pp. 237-240. BGUIR Publishing, Minsk 2015.
[7] S.V. Astanin, N.K. Zhukovskaya. Business processes control via modeling by fuzzy situational networks. Autom. Remote Control 75, (2014): 570–579.
[8] V.M. Chernenkiy, Yu.E. Gapanyuk, A.N. Nardid, A.V. Gushcha, Yu.S. Fedorenko. The Hybrid Multidimensional-Ontological Data Model Based on Metagraph Approach. In: A.K. Petrenko, A. Voronkov (eds.) Perspectives of systems informatics. 11th International Andrei P. Ershov Informatics Conference, PSI 2017, Moscow, Russia, June 27-29, 2017, Revised Selected Papers / edited by Alexander K. Petrenko, Andrei Voronkov, vol. 10742. Lecture notes in computer science, 0302-9743, vol. 10742, pp. 72–87. Springer (2018). doi: 10.1007/978-3-319-74313-4_6
[9] Yu.E. Gapanyuk. Metagraph Approach to the Information-Analytical Systems Development. In: Proceedings of the 6th International Conference Actual Problems of System and Software Engineering, Moscow, Russia, 2019, pp. 428-439.