Social networks analysis by graph algorithms on the example of the VKontakte social network

Maxim Kolomeets, Andrey Chechulin, and Igor Kotenko*
St. Petersburg Institute for Informatics and Automation (SPIIRAS)
St. Petersburg, 14 line V.O., 39, 199178, Russia
{kolomeec, chechulin, ivkote}@comsec.spb.ru

Received: April 11, 2019; Accepted: June 15, 2019; Published: June 30, 2019

Abstract

The development of social networks made it possible to form very complex structures of users and their content. As new services are added for users, the number of vertex types and edge types increase in the social network graph. Such structural increase opens up new opportunities for analysis. It becomes possible to obtain information about users, communities or trends by analyzing not the numerical or text information, but the structures that they form. Such structures can give a more accurate picture of the user, the community or the trend. To analyze these graph structures of social networks, one can use the entire arsenal of graph algorithms. In this paper, we consider their practical use in analyzing the growing structures of social networks and their limitations on the example of one of the largest social networks – VKontakte. The paper provides analysis and classification of graph algorithms in the context of social networks, as well as an approach to the analysis of the social network VKontakte using the graph database OrientDB.

Keywords social networks analysis, graph processing, network analysis, graph database.

1 Introduction

Social networks are a useful platform for disseminating information. They are used for commercial advertising, campaigning, research of communities, surveys and many other activities. The advantage of social networks over other services is the ability to analyze not only the parameters of users and the content, but also the structures that they form.

There are two types of methods that can be used to analyze social networks: the content analysis methods and the methods based on the analysis of graph structure topologies. Machine learning is used for content analysis, and graph algorithms for topology analysis.

Content analysis allows one to collect statistical data or interpret textual information. For example, in papers [1, 2, 3], the classification by the text of web pages is used to determine the text thematic. The paper [3] presents a general approach that combines the content analysis methods into a uniform integrated technique and allows one to determine the category of a web page with sufficiently high accuracy. These methods can also be applied to analyze the content in social networks, where instead of web pages the accounts of users and groups are used.

However, in social networks, the structures that are formed by objects are no less important than their categories. Such structures can be conveniently represented in the form of graphs with various types of

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 10:2 (June 2019), pp. 55-75
10.22667/JOWUA.2019.06.30.055

*Corresponding author: Laboratory of Computer Security Problems, St. Petersburg Institute for Informatics and Automation (SPIIRAS), St. Petersburg, 14 line V.O., 39, 199178, Russia, Tel: +7-812-328-7181

edges and vertices. Despite the small number of vertices and edges types, their combinations can form a huge number of structures that are different in nature and carry different information. The ability to analyze such structures, allows one to obtain new information about users, communities and processes occurring in social networks.

In this paper, we consider methods for analyzing social networks in terms of the structures that they form. This work is a part of the study "Monitoring and Counteraction to malicious influence in the information space of social networks" [4, 5, 6]. The novelty of this work lies in a new approach to the analysis of social networks, based on the formation of several different representations of their graph, which are based on the proposed hierarchical data model. The scientific contribution of the work is the analysis of the application of graph algorithms for social networks and the proposed approach to the formation of graphs based on the data model of the VKontakte social network, as well as the implementation of this approach using the graph database OrientDB.

The paper is organized as follows. Section 2 presents the systematization of graph algorithms and the analysis of the possibilities of their practical application. Section 3 considers the proposed approach to building the social network graph for its subsequent analysis, including a hierarchical data model of the social network. Section 4 presents an examples of the application of the proposed algorithms and the data model for analyzing the social circles of users. Section 5 shows the implementation of the proposed approach based on the OrientDB graph database using the example of the Connected Dominating Set algorithm. Section 6 presents the main advantages, disadvantages and plans for future work.

2 Graph algorithms for social network analysis

Since social networks are easily represented as graphs, the use of graph analysis algorithms is an obvious and effective way to analyze them. Algorithms for graph analysis are well studied. There are a large number of works in the field of clustering [7, 8, 9, 10], finding graph isomorphism [11, 12] and degree of graph similarity [13, 14, 15], finding certain structures [16, 17, 18, 19], determining cores [8, 18] and the most significant nodes [11, 20, 14]. Detailed systematization of these algorithms is presented, for example, in [11], where the basic analysis algorithms are presented, and the ways to implement them are described. These algorithms are applicable to any network, not just social networks.

We divided these algorithms into 5 categories, from the point of view of the results of applying these algorithms to social networks:

1. Centrality algorithms [11, 21, 22, 23, 24, 25] allow one to calculate the weight of the vertex based on the topology of the graph.

2. Clustering algorithms [7, 8, 9] provide an opportunity to divide the graph into clusters or to estimate the vertex taking into account the degree of its clustering.

3. Similarity algorithms [26, 12, 15] allow one to determine the similarity of two structures.

4. Algorithms of topological structure recognition [16, 17, 18] serve to find a subgraph of a specific structure or to assess how much a graph is similar to a structure of a certain type.

5. Algorithms of opinion leaders recognition [11, 20, 14] allow one to determine the weight of the vertex based on the structure of the graph, including the parameters of the edges.

Let us consider the particular types of these algorithms and what practical benefits they can bring when analyzing social networks.

Centrality Algorithms

The simplest measure is a Degree centrality. It determines the centrality of the vertex based on the number of edges for each vertex. Using Degree centrality, one can easily make a distribution of vertices. For example, if to take Degree centrality along a "user creates" edge (includes likes, posts, reposts, and comments created by the user) then one can create a distribution of user activity. However, Degree Centrality does not take into account the topology of the graph. To do this, one can use other algorithms, such as Eigenvector Centrality.

Eigenvector centrality calculates the centrality based on the Eigenvector Centrality of adjacent vertices. In the first step one can take Degree Centrality. After that, the measure is changing iteratively based on the parameters of the adjacent vertices. Like Degree Centrality, Eigenvector Centrality describes the centrality based on the number of edges, but takes into account the edges of neighboring vertices. In social networks, Eigenvector Centrality is useful in evaluating a potential audience by a friend graph. As Eigenvector Centrality accounts topology, it allows one to assess for each user how quickly information can spread from him/her through the repost.

An extension of Eigenvector Centrality is Katz Centrality. Katz Centrality is a special case of Degree Centrality, which takes into account all the vertices of the graph, and not just adjacent ones. When calculating it, the divider ratio is also set. It has a greater effect when the distance between the vertices increases.

Betweness centrality determines the centrality based on the number of minimum routes passing through the vertex. We can say that this measure determines the bottlenecks in the graph. In social networks, this measure can be used in the analysis of homophily (topologically separated sets of vertices on a particular basis). The bottleneck allows one to define users who serve as "bridges" between two social groups.

Closeness centrality determines the centrality based on the distance from the vertex to all other vertices in a connected graph. For the graph of friends, using Closeness Centrality, it is possible to determine the degree of isolation of the community and its users in relation to other communities. At the same time, isolation can be determined not only by "friendship", but also by likes or comments. Thus, Closeness Centrality allows one to determine when users are active only in a closed circle of people. It is important to note, that Harmonic centrality should be used for not connected graphs, which gives the same results as Closeness Centrality.

Clustering algorithms

Clustering Coefficient [7] allows one to determine the degree of clustering. There are two types of this measure: global and local. Clustering Coefficient is useful in assessing connectivity among users (among friends, users with shared groups, content, etc.) and content (for example, posts, likes, comments, reposts posted by some users). Finding K-cores [8] allows one to determine subgraphs in which each vertex has at least k edges. Thus, the graph is divided into layers, where the inner layers are the vertices with a large number of edges, and the outer ones – with the smaller ones. For example, such clustering of users by layers allows one to find the core that forms the community (in graph of friends). For the analysis of information flows, one can find the Connected Component [9] – an oriented subgraph in which there is a path between any two vertices. For example, finding a Connected Component in a repost forest (merging repost trees) allows one to find groups of users who repost each other's information. This includes using the K-connected Component – a subgraph in which there are k paths between any two vertices. Thus, the K-connected Component is the amplification of the Connected Component.

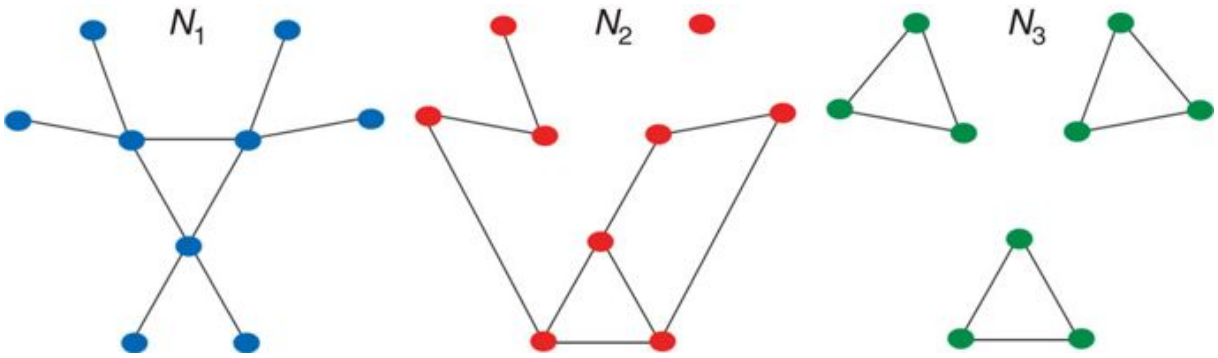

Figure 1: three topologically different graphs N1, N2 and N3 with the corresponding D-measure = 0.252, 0.565 and 0.473 [13]

Similarity Algorithms

Similarity algorithms can be used to compare graph topologies. A good practical application is the community comparison. Different social groups tend to form different structures. For these structures, one can make assumptions about the focus of this community. For example, if we have a community that we have identified as religious, it is appropriate to assume that a community with a similar structure may also have features of a religious community.

Classical comparison algorithms use Isomorphism Similarity [26, 12], which defines the exact coincidence of the graph topology, and the Graph edit distance [15], which determines how many vertices or edges should be removed or added to obtain an isomorphic graph. However, from the point of view of social networks, they have no practical use, since the compared graphs may have a similar topology, but vary in size. To compare graphs one can use Dissimilarity measure (D-measure).

D-measure [13] allows one to estimate the dissimilarity of graphs taking into account topological difference. For example, for the graphs shown in Figure 1, D-measure is 0.252 for N1, 0.565 for N2, and 0.473 for N3 (see Figure 1). D-measure takes into account the connectivity of the components and gives a smaller difference between N3 and N2 than between N3 and N1 or N2 and N1. Thus, Dissimilarity evaluates the topology without relying on the number of vertices, and evaluates only their connectivity. Due to this, the hierarchical or isolationist community will be very different from the usual (under the usual we mean the "small world"). However, the size of the communities will not be significant.

Algorithms of topological structure recognition

To find strongly connected communities, one can use the Cluque search [16] to determine a fully connected subgraph. However, there may be a lot of noise in social networks. For example, one of the users can hide data with privacy settings. Also, the structure of the graph may slightly change in dynamics, at different points of time breaking the full connectivity. For this, it makes sense to use K-plexes [17] - a subgraph such that each vertex of the subgraph is connected to at least n-k other vertices, where n is the number of vertices of the subgraph. K-plexes can be called a weakening of the Clique. K-plex search is a good alternative to Clique search, as it is more common for social networks and is almost Clique.

Algorithms for finding a subgraph-tree or determining whether a subgraph is a tree can be useful when searching for hierarchies. It should be noted that in social networks the trees are explicitly repost trees. Such trees in the repost forest can be identified by content parameters without using algorithms. It is more valuable to search trees and in implicit structures. For example, one can look at the graph

of friends in the community. By determining the tree structure in the community, one can assess the hierarchy of users.

Dominating set [18] allows one to determine the "skeleton" graph. Dominating Set is a subgraph whose adjacent vertices form the original graph. In total, there are three types of Dominating Set: (1) the usual set of vertices, the adjacent vertices of which form the original graph; (2) Connected Dominating Set [18] – a set of vertices forming a connected graph, the adjacent vertices of which form the original graph; (3) Weakly Connected Dominating Set [18] – a set of vertices whose adjacent vertices form the original graph and for each connected component of the Weakly Connected Dominating Set there is at least one common adjacent vertex. Dominating set is useful in defining a structural "skeleton of a graph", whether it be a graph of friends or a graph of content.

Homophily [11] is a measure expressing a partition of a graph into weakly connected subgraphs according to a particular criterion. The simplest example of homophily is the community by age, when people of the same age form more connected groups than people of other ages. Finding homophily in the community is possible by usage of various parameters, such as hobbies or music.

Algorithms of opinion leaders recognition

Laplacian Matrix [11] allows one to find leaders by determining the weight of the vertex based on their Laplace potentials of adjacent vertices. The Laplacian matrix can be used in directed graphs (for example, a graph of likes between users or reposts). The idea is that (using the graph of likes as an example) the weight of the user-supplied likes depends on the weight of the likes put to him. Thus, a vertex obtains the potential that depends on the potential of adjacent vertices. The higher the potential of the user, the greater the value of his/her likes or repost. Thus, using the Laplace Matrix in directed graphs, it is possible to calculate the leaders of opinions with greater potential. An example of the use of Laplacians is the popular PageRank algorithm [20, 14].

These graph analysis algorithms depend on the social network data model. Indeed, with a different presentation of the data one can obtain different results. In the next section, we present the proposed methods of representing social network graphs on the example of the data model of the VKontakte social network.

3 Social network graph representation

Based on the data of the social network, one can build various graphs. For example, one can build a graph of users, where the edges will be likes between them. At the same time, it is possible to construct a graph of users, where the edges will be comments between users. In this case, the same data – users and parameters of their posts (such as likes, comments, and links) – can give different types of graphs. Such graphs can differ structurally and in semantic content.

Thus, the results of analysis depend not only on the applied methods of graph processing, but also on the data model. In this section, we present the approach to the practical representation of social networks in graph databases.

This approach is based on the following components, which combine the method of representing social network graphs and the procedures for their formation:

1. A *hierarchical graph representation* that defines a *social network model* from which one can get various types of graphs.

2. *Real* and *virtual graphs* derived from the social network model and used to analyze a specific process.

3. *Aggregation of graph vertices* – obtaining new graphs by combining vertices.

These three components allow one to get different types of graphs based on the same data model. Consider these components on the example of the data model of the VKontakte social network.

Hierarchical graph representation

The basis of the proposed representation of a social network graph is the hierarchy of its elements – vertices and edges. Vertices and edges are represented as inheritable classes. This allows one to operate with classes of different levels of hierarchy.

Let us consider a hierarchical representation based on the analysis of the VKontakte social network [27]. VKontakte is a Russian social network. VKontakte users can microblog on their own behalf or on behalf of the community, participate in communities, post messages and media on friends' microblogs, upload and attach photo, video and audio content to posts, likes and repost posts. In general, VKontakte is similar to Facebook. Access can be obtained only to open data (users can change the privacy settings of their data). In this paper, we obtained data from the open API [27] by previously anonymizing them.

Graph G of the social network can be represented as a collection of interconnected vertices V of subjects $S = \{s_i\}, i = 1,\ldots,n$ and objects $O = \{o_j\}, j = 1,\ldots,m$. These categories are characteristic of any typical social network. s_i are represented in VKontakte by a user $u_k \in U$ or a community $c_l \in C$. E are the relationships of the action Act (are "friends", are participating in the community, etc.) and the relationships of inclusion $Incl$ (the post contains an image, the user profile includes videos, and so on.). At the highest level, the social network contains users U, communities C, objects O, relationships of action A and relationships of inclusion I:

$$G = (V,E), V = \{S,O\}, S = \{U,C\}, E = \{Act, Incl\}$$

By the user $u_k \in U$, we mean a subject $s_i \in S$, who has a personal page assigned to a specific account (person). By community $c_l \in C$, we mean a social subnetwork with its own pages. u_k can create and participate in c_l. We will also imply that the content of the pages of u_k and of the pages of c_l occurs according to similar principles.

By objects O, we mean various types of content that are used by S: media, media collections, microblogs, discussions, posts, etc. S and O can be organized into a hierarchical model of the graph's vertex classes. At low level, model can be organized in the next hierarchy of V (Figure 2):

1. Subjects S are accounts that represent real people; these can be both user accounts and community accounts (online shops, communities of interest, meetings and so on):
   (a) groups (or communities) C - meetings, events and other public pages, including their microblogs;
   (b) users U - user accounts, including their microblogs.
2. Objects O - a content that was created by the subjects:
   (a) Collections - collections of various kinds and having different functions:
      i. Discussion - a public space in the form of a message feed that was created by a subject for a specific topic or a discussion in which users can post the messages;
      ii. Libs — collections of media files, such as video albums, photo albums, and audio playlists:
         A. Photo-libs – collections of photos organized into photo albums;

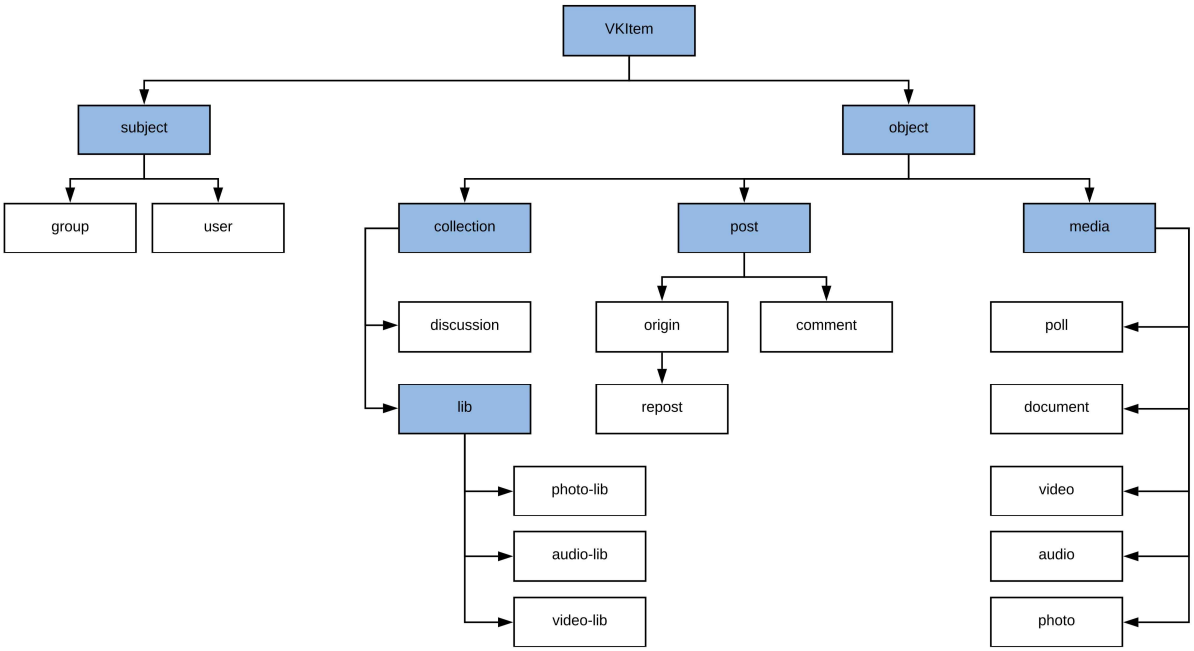

Figure 2: the hierarchy of the vertex classes of the VKontakte social network graph (abstract classes are highlighted)

B. Audio-libs – collections of audio recordings organized into playlists;
C. Video-libs – collections of video records organized into video albums;

(b) Posts – a container cards with the text which the subject publishes:
   i. Origins – posts created by the subject personally;
   ii. Reposts – posts posted by a subject, and referred to another post;
   iii. Comments – posts posted as a comment to another post or discussion;

(c) Media – media content files and the like:
   i. Poll – a multi-choice survey;
   ii. Document – a downloaded file of any format;
   iii. Video – a loaded video;
   iv. Audio – a loaded audio;
   v. Photo – a loaded image.

As already mentioned, S can perform actions Act on O and S can be in the relation inclusion $Incl$ with O. So,

$$S\ Act\ \{O,S\},\{O,S\}\ Incl\ O$$

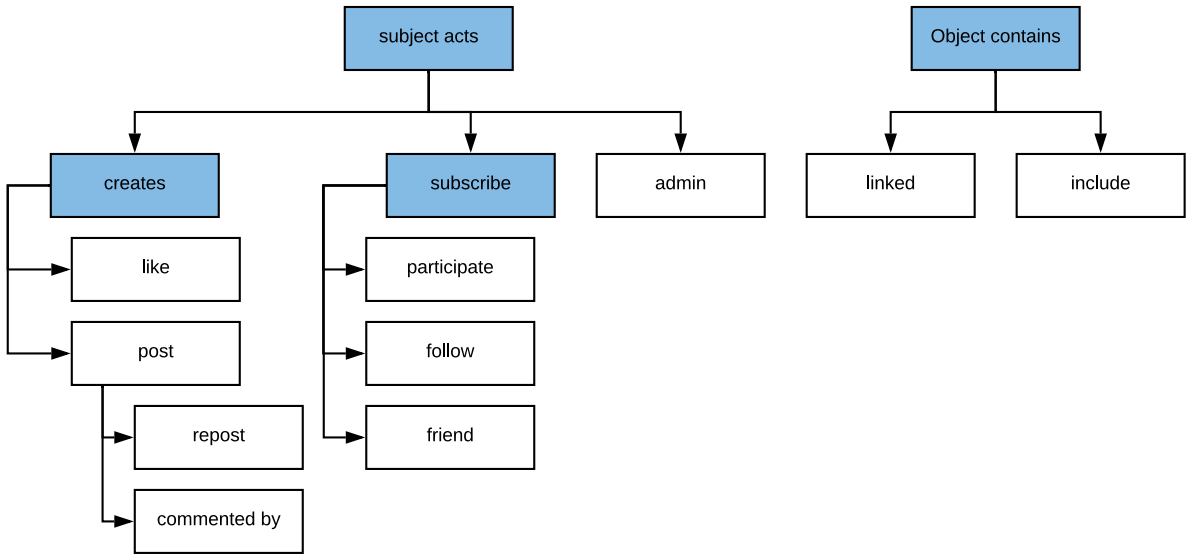

Figure 3: the hierarchy of edge classes of the social network graph VKontakte (abstract classes are highlighted)

At low level, the relationships E can also be organized into a model. For VKontakte, the hierarchy of the vertex classes is as follows (Figure 3):

1. The subject acts *Act* – the actions of the subject in relation to another subject or object:
   (a) Creates – actions by the subject on the object;
      i. Like – creation of like by subject;
      ii. Post – creation of post by subject;
         A. Repost – creation of repost (is referred to another post) by subject;
         B. Commented by – posting as a comment to another post or in a discussion;
   (b) Subscribe – creating relationships between subjects;
      i. Participating in a group / meeting - the relationship between users and groups;
      ii. Follow – one-way relationship between users;
         A. Friend – two-way relationship between users;
   (c) Administers — a relationship indicating that the user is the administrator of the group.
2. The object contains *Incl*:
   (a) Linked – an object contains a link to another subject or object;
   (b) Include – an object or a subject contains another object.

The final model of the class hierarchy of the VKontakte social network is represented in Figure 4 as a matrix. Rows and columns depict the hierarchy of the vertices V. Cells of the matrix reflect which classes of E can exist between one or another class of V. Based on this model, one can build G. The vertex classes are in matrix titles of rows and columns, the edges classes are in matrix cells.

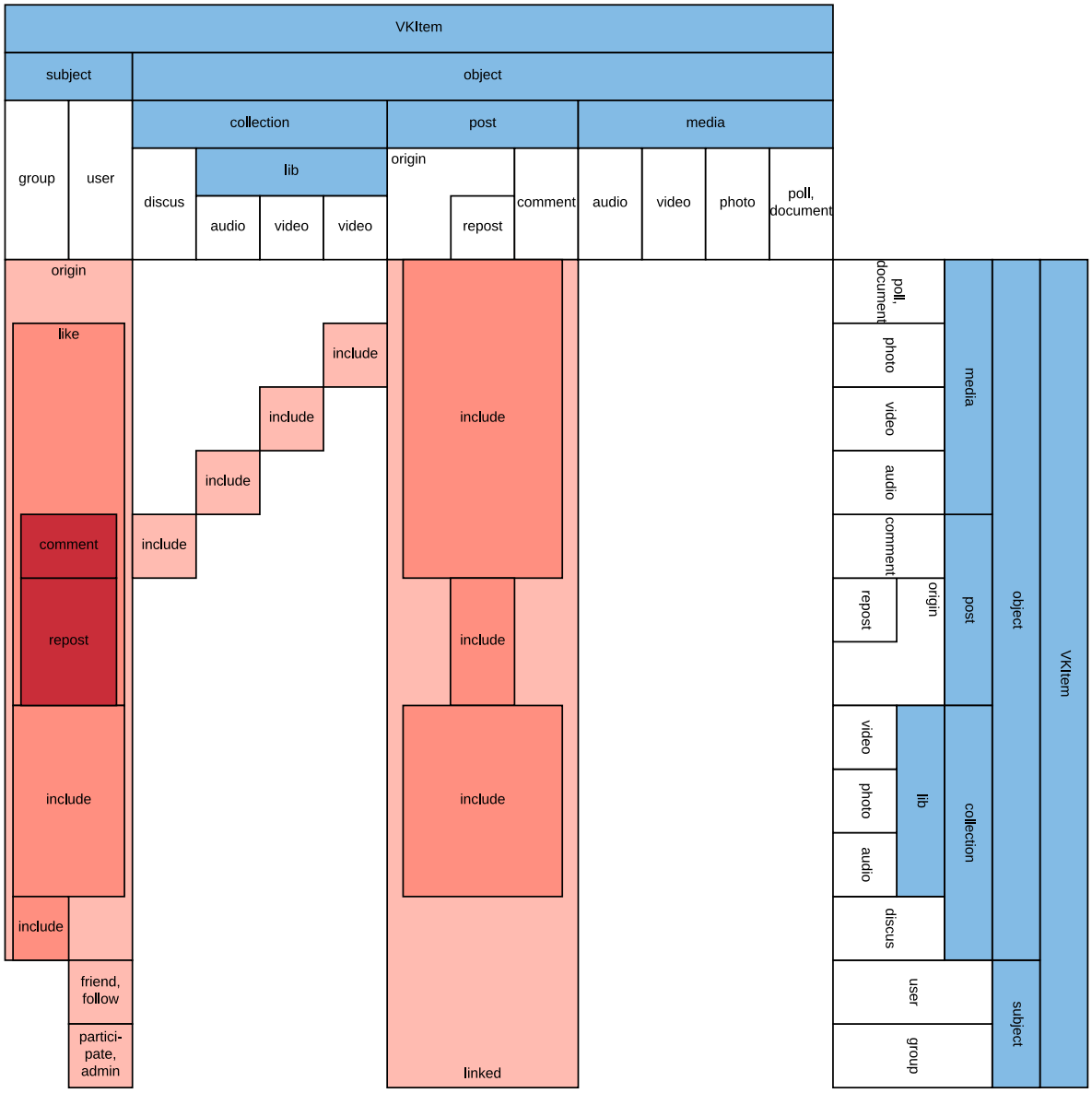

Figure 4: a hierarchical model of the VKontakte social network graph

Real and virtual graphs

We can designate two types of structures that are formed in social networks when one is trying to analyze data: real graphs and virtual graphs.

By *real graphs* we mean structures that are derived from data directly. Thus, the real graph is a graph that can be obtained by scanning a social network and which will be stored in a database according to a specific data model. An example is shown in Figure 5, where the real graph is formed by continuous edges: a post with like forms a real structure, which consists of the post, the user who created the post and the user who liked the post.

A *virtual graph* is a graph obtained by processing a real one, for analyzing a specific process. For

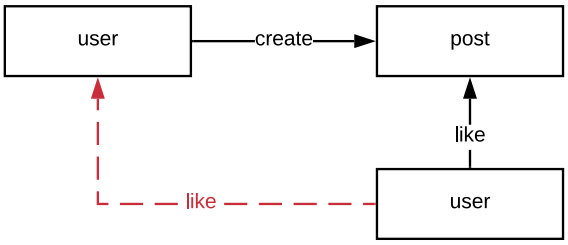

Figure 5: the real graph is represented by continuous edges, the virtual graph – by dashed edges

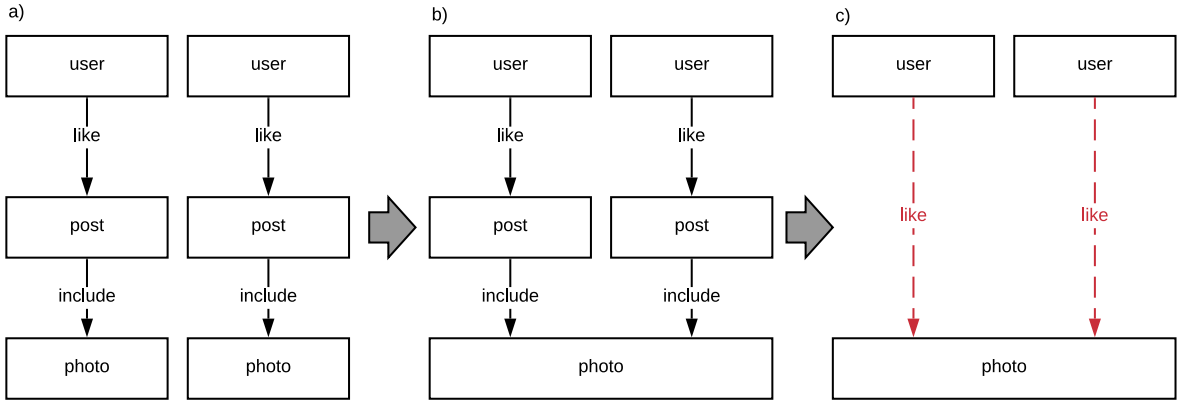

Figure 6: aggregation of the graph vertices by the class “photo”

example, in Figure 5, the virtual graph is represented by dashed edges and shows likes between users. Although the user cannot like another user (on the example of VKontakte), we can get them indirectly on the basis of likes to the content and form a graph of users likes to each other.

Instance aggregation

Algorithms, that determine the similarity of instances of a class, allow one to aggregate the vertices of the graph, thereby creating a new structure. An example of the simplest aggregation is shown in Figure 6: a user liked a post with an image that one group loaded, another user liked a post of another group that contains a similar image (Figure 6 a). If these two photos are identical, their vertices can be combined into one (Figure 6 b). Later, when creating virtual graphs, the post vertices can be deleted, forming the virtual graph of images the users like (Figure 6 c).

Based on the hierarchical data model of the social network, one can create various types of graphs (real and virtual) by combination of hierarchies or by creation of virtual edges or using instance aggregation. The results of the algorithms will depend on the context of the analyzed graph. In the next section, we give an example of graph analysis using the presented data model and graph representations.

4 Application examples

The goals and objectives of analyzing the graph structures in social networks can be reduced to two tasks: searching the structure and estimating the structure parameters. Examples of searching the structure include finding clusters, selecting subgraphs of trees, finding kernels or Dominating Sets followed by filtering. Examples of the structure evaluation are all methods for determining any parameters, for example all centrality algorithms or the assessment of the structure similarity. But most often, the analysis includes both the search for the structure and the interpretation of its parameters.

Thus, the sequence of actions is as follows:

1. Finding the structure - the result is a subgraph or a set of vertices.

2. Structure evaluation - the result is the parameters (both numerical and categorical) for the whole structure or separately for each vertex of the structure.

3. Structure interpretation - expert interpretation of the structure parameters depending on the context of the initial graph data.

Let us consider the analysis sequence on the example of *finding user social circles*.

User social circles are communities which can be thought of as channels of information dissemination. The user can associate himself with these communities - a circle of school graduates, colleagues, dog lovers, etc. At the same time, he/she may not have anything in common with such communities. For example, among dog lovers there may be zoo defenders, among colleagues there may be a community of vegetarians, etc. The idea is that social circles can interact with the user. If to determine the user social circles, it is possible to evaluate the user's information picture of the world.

The obvious way to search for social circles is to search among friends who form communities. Finding social circles means finding users where almost everyone knows everyone. For example, if a user studies in a university, almost certainly, all of his classmates will be in his friend list. Moreover, they will strive to form a fully connected graph. In practice, fully connected graphs are rare, since it makes sense to look for "almost fully connected graphs" (K-plex). These K-plexes are social circles through which the user receives information. These social circles can be characterized, for example, by highlighting attributes common to all users of this circle.

Thus, the algorithm for determining the user social circles is as follows:

1. Finding all adjacent vertices-friends of the analyzed user and constructing a graph with the type of the edge connection "friend".

2. Remove from the graph the analyzed user. It is necessary to eliminate the influence of user parameters on the evaluation of social circles.

3. Using the algorithm for finding K-plex with a gradual weakening of the requirements for full connectivity. It is necessary for finding more social circles.

4. Removing from the K-plex set such K-plex that completely intersects with more fully connected K-plex and with a large number of vertices. It is necessary to reduce the number of K-plex found, leaving only more connected and larger communities.

5. Analysis of the characteristics of the remaining K-plex on the basis of the allocation of common parameters of users.

It should be noted that in step 2, the user is removed from the graph in order to eliminate the influence of user parameters when analyzing the characteristics of k-plex in step 5. One of the interesting cases is finding a circle that does not have common attributes with the user. For example, this can happen if the user has recently joined a circle. In this case, deleting the user from the graph allows one to more accurately characterize k-plex and, accordingly, it is more likely to detect a discrepancy between user attributes and a circle.

The step 5 is the analysis of the parameters of K-plex. It consists in finding the parameters for all vertices and their subsequent interpretation by an expert. General parameters should be sought among:

1. Attributes of the vertices of the class “user” - common interests, age, gender and other fields.

2. Adjacent vertices of the class “user” along the edges of all classes except the class “friendship” - common groups, the post under which everyone liked, the post under which everyone wrote a comment and other adjacent vertices. For example, if in a circle all users are in the group dedicated to the protection of animals, we can assume that the social circle has a corresponding orientation.

The following algorithms can also be used to evaluate circles.

Centrality measures enable one to define the central vertices of the graph, the attributes of which can indicate the focus of the social circle.

Clustering algorithms and algorithms of topological structure recognition enable one to narrow the search by truncating the graph and then evaluate the remaining elements. For example, for a graph of group subscribers, getting a K-core and then deleting layers of a smaller order (for example, deleting vertices that are present only in 1-core and 2-core) allow one to get a more connected structure. Then, the use of Dominating Set allows one to get the ”skeleton” of users who form a group, as well as highlight the main highways through which information is distributed among users of the group. Analyzing the attributes of individual users in the Dominating Set can indicate the focus of the whole social circle.

Comparison algorithms can be used to determine the similarity of the analyzed social circle with the already analyzed social circle. For example, if there is an already analyzed group, the focus of which is defined as religious, it can be used for comparison using the Dissimilarity measure. For users of some religious groups, there is a strong cohesion. Therefore, if the Dissimilarity measure of the graph of friends of social circle (adjacent vertices by “friend” edge for users in social circle) is close to the Dissimilarity measure of religious group, it can be assumed that the focus of the social circle is close to religious.

Algorithms of opinion leaders recognition allow one to get the most influential users and evaluate the focus of social circle by attributes of these users. Influence can be determined using PageRank or Laplacian Matrix, where edges and weights can be likes, reposts comments and their number. To do this, it is necessary to build temporary edges of the corresponding type between users (virtual graph). For example, to build a graph of likes, it is necessary for each user to define the posts that he/she published, and then identify all the users who put like under these posts, and then merge the vertex-user and vertex-post into one. The resulting graph will be the users (vertices) and likes they put together (edges).

It is important to note that the evaluation of social circles is carried out by an expert with a comprehensive analysis of the characteristics and subsequent decision making. One characteristic found does not indicate the focus of the social circle For example, a general attribute that has been interpreted as “drug trafficking” may indicate both: a social circle the focus of which is related to the distribution of drugs, and the social circle that is referred with medical activity. Thus, the decision and complex assessment is made taking into account the set of estimates obtained from different characteristics.

5 Implementation

For the analysis of social networks, we developed a software prototype that provides data to the given data model, performs data analysis and visualizes the results.

The data obtained from the VKontakte API [27] is transferred to the graph database OrientDB [28]. Using OrientDB, we implemented the data model described in Section 3. Thus, the data after loading can already be used for analysis using graph algorithms.

Data analysis is performed directly in OrientDB using database queries. In order not to unload a large amount of data from the database and not load it back, we used the server built into OrientDB with support for JavaScript functions. Functions that implement algorithms send requests to the database and, depending on the answer, decide which query to do next. In order to preserve the integrity of the base, instead of deleting or copying the vertices, we use the coloring of the vertices.

Let us consider below the implementation of the Connected Dominating Set [18] algorithm in OrientDB.

0. Before executing the algorithm, vertices and edges of the graph are colored to white.
1. In the first step, we look for the vertex with the most edges and colored it to red.
2. In the second step, the edges of the red vertex are colored to green.
3. In the third step, the edges adjacent to the red vertex are colored to green.
4. In the fourth step, the edges between the two green vertices are colored to green.
5. At the fifth step among the green vertices, we look for the vertex with the most edges and colored it to red.

Then steps 2-5 are repeated in a loop until there are no white vertices in the graph.

This algorithm on the SQL-pseudocode is given below.

This approach based on the coloring allows one to analyze data in OrientDB directly, without uploading them to third-party modules for analysis.

The visualization module [29, 30, 6] is used to view the results of the algorithm. The visualization module is implemented on D3.js [31] and works in the browser. The module requests the graph in JSON format from the database and renders it using the force layout drawing method [6].

Let us consider the use of this prototype for the analysis of several groups in the VKontakte social network.

We analyzed the VKontakte groups by the structures that their users form. In the first step, it was necessary to get a graph that can represent interactions between the group members. To do this, let us apply the hierarchical data model.

The vertices of the desired graph are *users*, the interactions are various classes of edges. According to the data model (Figure 4) between *users* can be the following classes of edges (types of interactions): *follow*, *friend*. In the class hierarchy, the *follow* and *friend* classes are extended from the *subscribe* class (Figure 3). The *participate* class is also extended from the *subscribe* class. At the same time, the *participate* edge can only be between the *user* and the *group*, so querying *users subscribe users* will only give the *follow* and *friend* edges.

Another type of user interaction is that users leave posts and comments on each other's microblogs, posts and reposts. Within the framework of the hierarchical data model, it looks like in Figure 7 and 8,

Algorithm 1 Connected Dominating Set

```
1: function STEP 0
2:     SQL(update V set color='white')
3:     SQL(update E set color='white')
4: function STEP 1
5:     SQL(update (select V order by DegreeCentrality[] desc limit 1) set color='red')
6: function STEP 2
7:     SQL(update (select E from (select V where color='red') where color='white') set
       color='green')
8: function STEP 3
9:     SQL(update (traverse (select V where color='red') where color='white') set color='green')
10: function STEP 4
11:    SQL(update (select E where E.in_V='green' and E.out_V='green') set color='green')
12: function STEP 5
13:    SQL(update (select V where color='green' order by degreeCentrality["white"] desc limit 1)
       set color='red')
14: STEP 1
15: STEP 2
16: STEP 3
17: STEP 4
18: while SQL(count V where color = "white") != 0 do
19:    STEP 5
20:    STEP 2
21:    STEP 3
22:    STEP 4
```

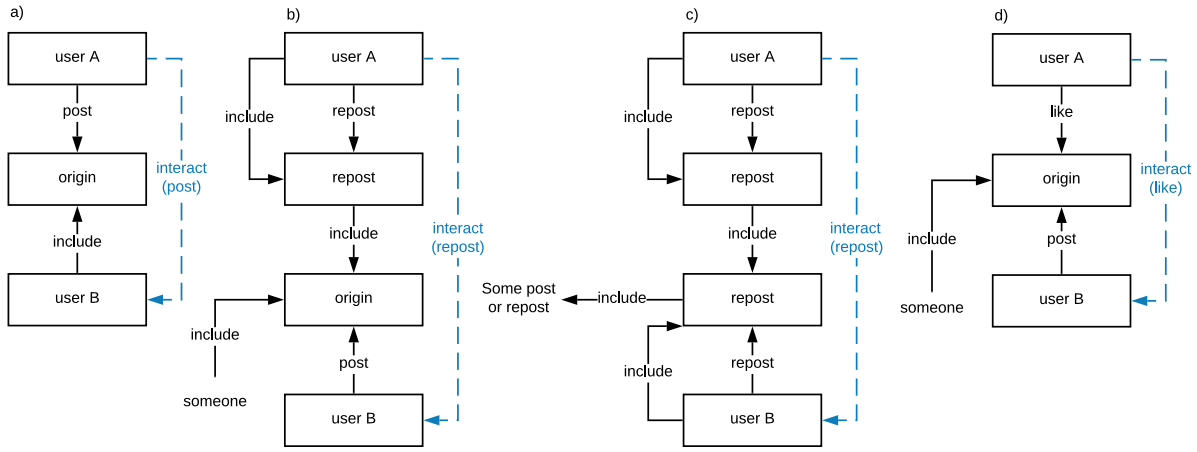

Figure 7: User to user interaction options *a-d*

where: the continuous edge *post* and *repost* show who posted the record; the continuous edge *include* indicates that the entry contains another entry or that the user's microblog contains this entry; the dashed edge shows the virtual edge it is needed to get.

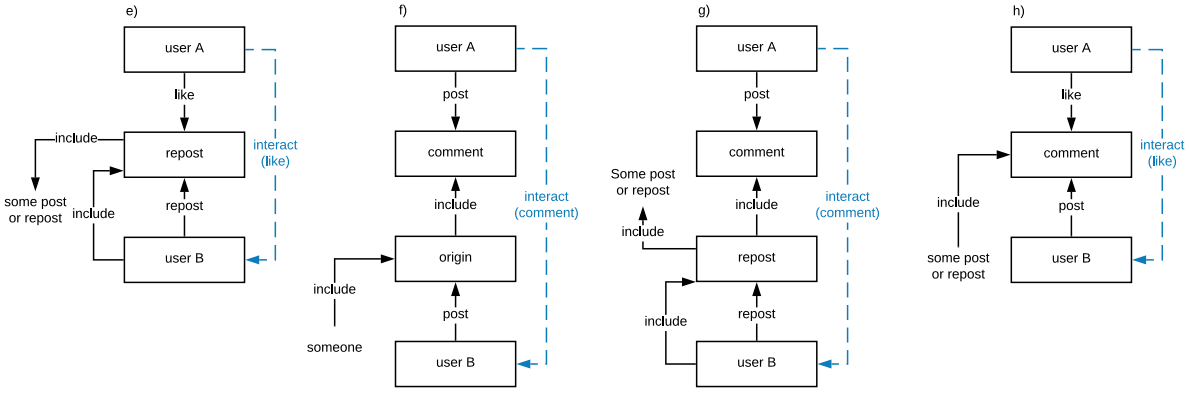

Figure 8: User to user interaction options *e-h*

The specifications of the options are as follows:

1. Option a – *user* A can *post* an *origin* on *user* B (Figure 7a).

2. Option b – *user* A can *post* a *repost* of *origin* that was *posted* by *user* B (Figure 7b). Note that *user* A can *post* a *repost* only on his own page. Also, *user* B may not necessarily *post* an *origin* on his own page — he can do it on the page of another *user* or *group*.

3. Option c – *user* A can *post* a *repost* of *repost* that was *posted* by *user* B (Figure 7c). For our purposes, from where *user* B *post* a *repost* does not matter, since we analyze only the interaction between *users* A and B.

4. Option d - *user* A may *like* the *origin* that was *posted* by *user* B (Figure 7d).

5. Option e - *user* A may *like* the *repost* that was *posted* by *user* B (Figure 8e).

6. Option f – *user* A can *post* a *comment* to the *post* that was *posted* by *user* B (Figure 8f).

7. Option g – *user* A can *post* a *comment* to the *repost* that was *posted* by *user* B (Figure 8g).

8. Option h - *user* A can *like* the *comment* that was *posted* by *user* B (Figure 8h).

The hierarchical structure allows to simplify all cases. On the interaction schemes, the *user* A performs the edge-actions: *post*, *repost* and *like*. All these actions are extended from the *creates* edge class. *User* A performs the *creates* action on objects of the classes *origin*, *repost* and *comment* which are extended from the vertex class *post*. These objects can have edges: *include* to the vertex *user* (Figure 7a), *include* to the vertex *post* (Figures 7b, c and 8f,g) or *creates* to the vertex *user* (Figures 7 d and 8e, h). Thus, the presented cases are reduced to three schemes:

1. *User A creates post that include user B.*

2. *User A creates post that creates user B.*

3. *User A creates post that include post that creates user B.*

Algorithm 2 Pseudocode for search for user interaction

```
1: SQL(select (traverse 'include' (traverse 'creates' 'user')) where class = 'user')
2: SQL(select (traverse 'creates' (traverse 'creates' 'user')) where class = 'user')
3: SQL(select (traverse 'creates' (traverse 'include' (traverse 'creates' 'user')) where class = 'user')
```

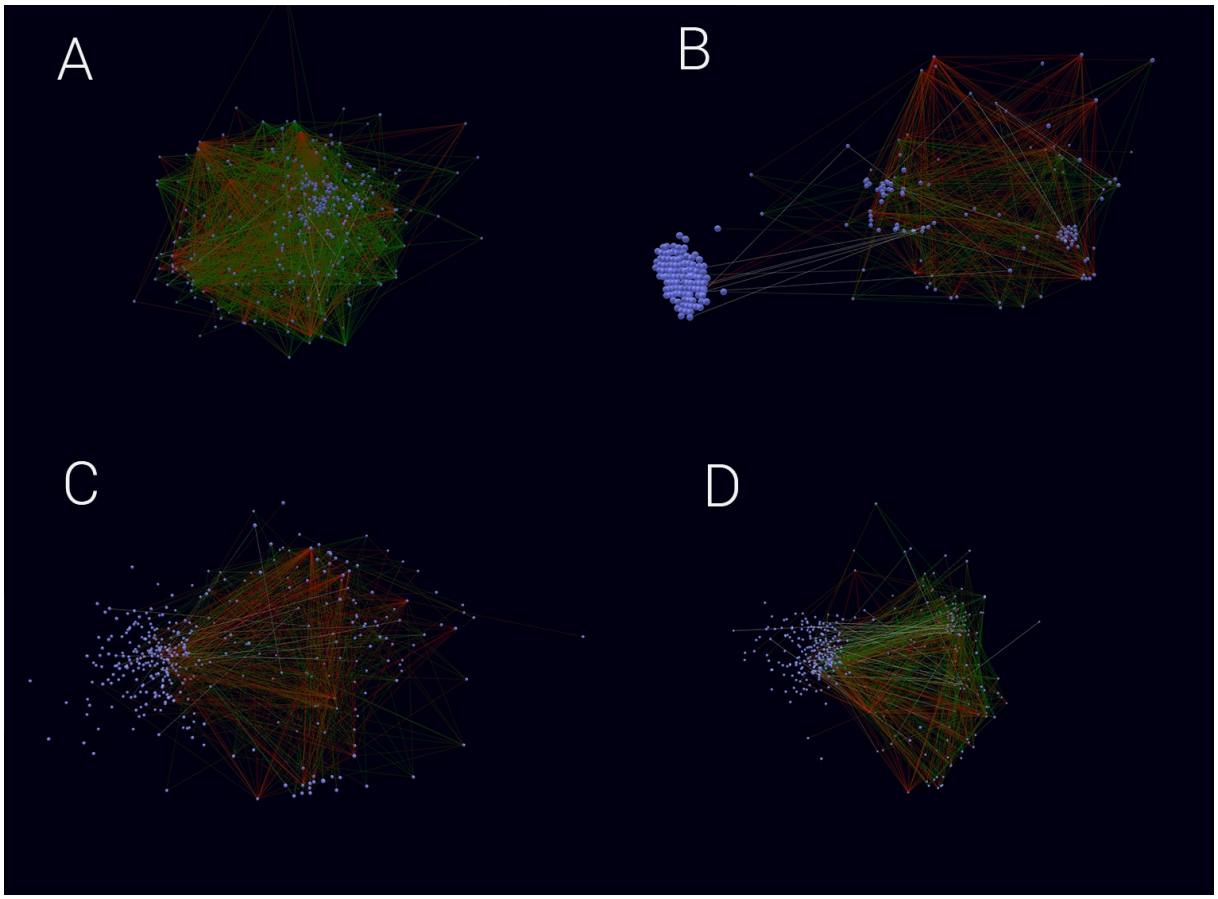

Figure 9: Interactions between the group members [32]

The search for the interaction of the user A with other users is reduced to traverse vertices by the following algorithms:

Based on the approach, we built virtual graphs for four VKontakte groups by combining the *graph of interaction* with the graph describing as *users subscribe users*. These groups contain from 200 to 5000 users. We used a visualization module to draw these graphs [32]. (Figure 9).

The result of the Connected Dominating Set for these graphs are shown in Figure 10 [32].

The Connected Dominating Set in Figure 10 demonstrates the connected "skeleton" of the graph. Adjacent vertices of the Connected Dominating set form the initial graph. Thus, it is possible to assess the structure of the community and identify the vertices that are its basis.

Thus, the developed prototype allows one to analyze social networks directly in a graph database. The approach based on the hierarchical data model helps to simplify virtual graph construction. This allows one to operate on different levels of abstraction and simplify requests. Due to the visualization

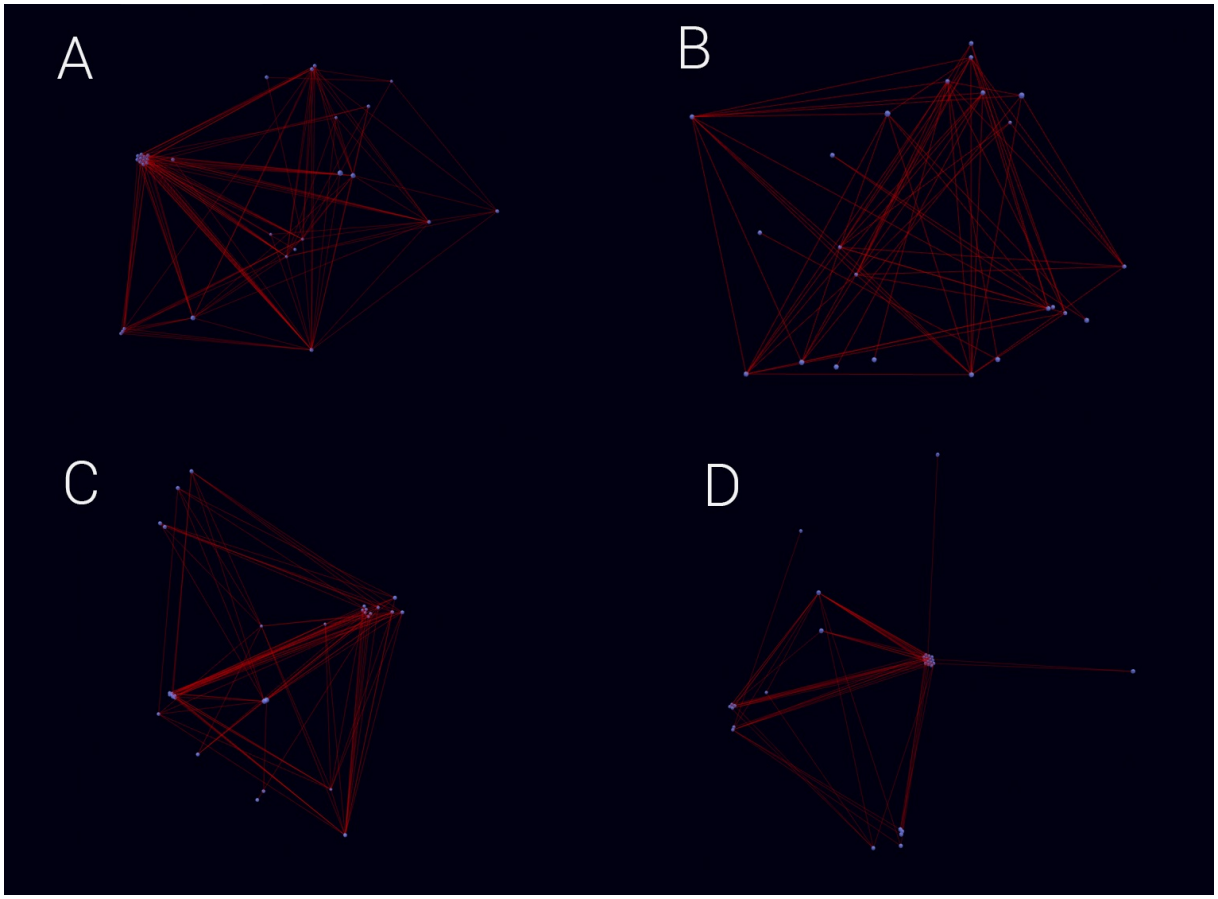

Figure 10: the result of the calculation of the Connected Dominating Set in OrientDB [32]

module, one can evaluate the results of the algorithms or decide on the next steps of the analysis.

6 Discussion

Social networks are a subclass of nformation sites in which there is not only content, but also various multidirectional kinds of links (relationships) between content. The presence of relationships changes the approach to analysis. The topology of the social network graphs itself can provide information regardless of the content. This feature opens up new opportunities for analysis.

Graph algorithms provide a different type of information than the content information. For example, using D-measure, one can evaluate the community topology and look for similar communities among other communities. With the use of Closeness Centrality, one can assess the user's isolation in relation to the community. With the help of K-plex, one can find user social circles, by which he/she can indirectly evaluate the user himself.

Another advantage of graph algorithms is that they are not tied to specific languages, while the content analysis depends on the language of the analyzed community. At the same time, graph algorithms can act as additions to content analysis. For example, numerical measures of Centrality Measures can be used along with numerical measures of content analysis for calculating integral metrics.

In general, analysis of the topology of social networks is reduced to the application of graph algo-

rithms over a specific subgraph - a subgraph of friends, a subgraph of reposts, a subgraph of likes, etc. One can create subgraphs based on different levels of the social network data model hierarchy, as well as using virtual graphs and vertex aggregation. In this case, the meaning of the result of the work of the same analysis algorithm will depend on the analyzed subgraph. For example, Degree Centrality in the graph of friends shows only the number of user's friends, while Degree Centrality in the "user creates" graph shows the user activity.

In the implementation section, we showed how exactly the hierarchical data model allows one to create various types of graphs for further analysis. The first application is a hierarchical model that can be used to obtain subgraphs at different levels of abstraction of the classes of vertices and edges. The second application is to simplify queries for constructing virtual graphs.

Based on these two approaches, we obtained a graph of interactions between subscribers of four VKontakte groups. Based on the first approach, we have received the graph of friends and follow (subscribers). Based on the second approach, we received the virtual graph of interactions between users, when users post comments, origins (posts), likes and repost each other's records. Combining these two graphs, we got the new graph, which was not originally provided by our data model. Applying the Dominating Set to them, we have significantly simplified its structure, forming a kind of "frame" of the graph. Using the visualization of the results, the hierarchical model and analysis algorithms allows one to analyze many variations of social network structures. Visualization of the results obtained in the implementation section is available at the link [32].

This approach shows that using the hierarchical model with the construction of virtual graphs and (if necessary) aggregation of vertices, it becomes possible to significantly expand the variations in the representation of social network structures. This approach allows one to make flexible analysis of the social network using the graph algorithms.

Among the disadvantages of the graph algorithms, it is possible to single out the fact that some of the algorithm's problems are difficult to solve. For example, a Clique search [19] belongs to the NP-complete class of tasks and it is possible to find Clique, for example using the Bron–Kerbosch algorithm [16], only on small graphs. However, these limitations can be circumvented in perspective. For example, one can increase the speed of the algorithms using graphics processing units (GPUs). So, Betweness centrality, during which it is necessary to calculate the minimum trees for each vertex, is easy to parallelize [33]. Another approach is based on controlled data truncation - when the structure is reduced to a specific structure (for example, a tree) on which the algorithm runs faster.

Also, the disadvantage is that the graph structure and the results of the algorithms are still difficult to percept. In order to effectively analyze the results and make decisions, one can use the methods of visual analytics [29, 6].

Besides, there remains the general problem of completeness of information in the analysis of structures. Note that the results shown in Figure 9 contain unlinked vertices and small components that are not connected to the main big connected component. On the force layout rendering techniques, they are organized into small clouds. Such elements are obtained when users hide their pages with privacy settings, thus it is impossible to get data about their friends or records.

In general, the analysis of data structures provides qualitatively new information about the processes occurring in social networks. It can be used as an independent method of analysis and as a supplement to content analysis.

The hierarchical model of a social network presented in the paper allows one to greatly diversify the data structures by semantic content. Together with a large variety of algorithms, it provides opportunities for analyzing social networks.

7 Conclusion

The analysis of social networks is not limited to the analysis of user parameters and content. Using a variety of graph analysis algorithms, it becomes possible to analyze the structures themselves that these users and content form. The ability to analyze the structures allows one to otherwise obtain data about the processes occurring in social networks.

In this paper, we reviewed the methods for analyzing social networks from the point of view of the structures. We systematized the graph analysis algorithms in the context of social networks and considered practical examples of how to use them. We suggested a new approach to analysis of social networks, based on the formation of several different representations of their graphs, which are based on the proposed hierarchical data model. Thus, the hierarchical data model suggested in the paper allows one to create various types of graphs and, therefore, new types of structures for analysis. The proposed algorithms and the data model were applied to analyze the user social circles. The implementation of the proposed approach based on the OrientDB graph database was considered.

In the future, it is planned to solve some of the problems described in the Discussion section: explore the possibilities and limitations of working with NP-complete algorithms using GPUs and truncating information, as well as developing visual analytics techniques for presenting analysis results.

Acknowledgment

The work is performed by the grant of RSF #18-71-10094 in SPIIRAS.

References

[1] I. Kotenko, A. Chechulin, A. Shorov, and D. Komashinsky, "Analysis and evaluation of web pages classification techniques for inappropriate content blocking," in *Proc. of the 14th Industrial Conference on Industrial Conference on Data Mining (ICDM'14), St. Petersburg, Russia*, ser. Lecture Notes in Computer Science, vol. 8557. Springer, Cham, July 2014, pp. 39–54.

[2] D. Novozhilov, I. Kotenko, and A. Chechulin, "Improving the categorization of web sites by analysis of html-tags statistics to block inappropriate content," in *Proc. of the 9th International Symposium on Intelligent Distributed Computing (IDC'15), Guimarães, Portugal*, ser. Studies in Computational Intelligence, vol. 616. Springer, Cham, October 2015, p. 257–263.

[3] D. Komashinsky, I. Kotenko, and A. Chechulin, "Categorisation of web pages for protection against inappropriate content in the internet," *International Journal of Internet Protocol Technology*, vol. 10, no. 1, pp. 61–71, 2017.

[4] I. Kotenko, I. Saenko, A. Chechulin, V. Desnitsky, L. Vitkova, and A. Pronoza, "Monitoring and counteraction to malicious influences in the information space of social networks," in *Proc. of the 10th International Conference on Social Informatics, Part II, St. Petersburg, Russia*, ser. Lecture Notes in Computer Science, vol. 11186. Springer, Cham, September 2018, p. 159–167.

[5] A. Pronoza, L. Vitkova, A. Chechulin, and I. Kotenko, "Visual analysis of information distribution channels in social networks to counter unwanted information," in *Proc. of the 3rd International Scientific Conference on Intelligent information technologies for industry (IITI'18), Part II, Sochi, Russia*, ser. Advances in Intelligent Systems and Computing, vol. 875, September 2018, pp. 95–105.

[6] M. Kolomeyets, A. Chechulin, and I. Kotenko, "The technique of structuring social network graphs for visual analysis of user groups to counter inappropriate, dubious and harmful information," in *Proc. of the 2nd International Scientific-Practical Conference Fuzzy Technologies in the Industry (FTI'18), Ulyanovsk, Russia*, ser. CEUR Workshop Proceedings, vol. 2258. CEUR-WS.org, October 2018, pp. 87–95.

[7] G. Chalancon, K. Kruse, and M. M. Babu, "Clustering coefficient," *Encyclopedia of Systems Biology*, pp. 422–424, 2013.

[8] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, "k-core organization of complex networks," *Physical Review Letters*, vol. 96, no. 4, February 2006.

[9] L. He, Y. Chao, K. Suzuki, and K. Wu, "Fast connected-component labeling," *Pattern Recognition*, vol. 42, no. 9, pp. 1977–1987, 2009.

[10] G. Destino and G. T. F. D. Abreu, "Network boundary recognition via graph-theory," in *Proc. of the 5th Workshop on Positioning, Navigation and Communication (WPNC'08), Hannover, Germany*. IEEE, March 2008, pp. 271–275.

[11] M. E. J. Newman, *Networks an introduction*. Oxford University Press, 2018.

[12] L. A. Zager and G. C. Verghese, "Graph similarity scoring and matching," *Applied Mathematics Letters*, vol. 21, no. 1, pp. 86–94, 2008.

[13] T. A. Schieber, L. Carpi, A. Díaz-Guilera, P. M. Pardalos, C. Masoller, and M. G. Ravetti, "Quantification of network structural dissimilarities," *Nature Communications*, vol. 8, pp. 13 928:1–13 928:10, 2017.

[14] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," in *Proc. of the 7th International World Wide Web Conference, Brisbane, Australia*, April 1998, pp. 161–172.

[15] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Analysis and Applications*, vol. 13, no. 1, p. 113–129, 2009.

[16] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[17] B. Balasundaram, S. Butenko, and I. V. Hicks, "Clique relaxations in social network analysis: The maximumk-plex problem," *Operations Research*, vol. 59, no. 1, pp. 133–142, 2011.

[18] W. Duckworth and B. Mans, "Connected domination of regular graphs," *Discrete Mathematics*, vol. 309, no. 8, pp. 2305–2322, 2009.

[19] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, "The maximum clique problem," in *Handbook of Combinatorial Optimization*, 1999, pp. 1–74.

[20] W. Xing and A. Ghorbani, "Weighted pagerank algorithm," in *Proc. of the 2nd Annual Conference on Communication Networks and Services Research (CNSR'04), Fredericton, NB, Canada*. IEEE, May 2004, pp. 305–314.

[21] S. P. Borgatti, "Centrality and network flow," *Social Networks*, vol. 27, no. 1, pp. 55–71, 2005.

[22] B. Ruhnau, "Eigenvector-centrality - a node-centrality?" *Social Networks*, vol. 22, no. 4, pp. 357–365, 2000.

[23] U. Brandes, "A faster algorithm for betweenness centrality," *The Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[24] L. C. Freeman, S. P. Borgatti, and D. R. White, "Centrality in valued graphs: A measure of betweenness based on network flow," *Social Networks*, vol. 13, no. 2, pp. 141–154, 1991.

[25] M. J. Newman, "A measure of betweenness centrality based on random walks," *Social Networks*, vol. 27, no. 1, pp. 39–54, 2005.

[26] L. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, p. 1367–1372, 2004.

[27] "VKontakte | API methods | Developers," https://vk.com/dev/methods [Online; Accessed on June 24, 2019].

[28] "OrientDB | Graph Database | Multi-Model Database," https://orientdb.com/ [Online; Accessed on June 24, 2019].

[29] A. Chechulin, M. Kolomeec, and I. Kotenko, "Visual analytics for improving efficiency of network forensics: Account theft investigation," *Journal of Physics: Conference Series*, vol. 1069, pp. 012 062:1–012 062:9, 2018.

[30] M. Kolomeec, A. Chechulin, A. Pronoza, and I. Kotenko, "Technique of data visualization: Example of network topology display for security monitoring," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 7, no. 1, p. 58–78, 2016.

[31] M. Bostock, "D3.js | Data-Driven Documents," https://d3js.org/ [Online; Accessed on June 24, 2019].

[32] "Graphs of users | Approach results visualization," http://comsec.spb.ru/files/jowuavis/visualization.html, [Online; Accessed on June 24, 2019].

[33] A. Mclaughlin and D. A. Bader, "Scalable and high performance betweenness centrality on the gpu," November 2014.

Author Biography

Maxim Kolomeets is currently a PhD student under joint supervision of ITMO University and University Toulouse 3 Paul Sabatier. He junior researcher at the Laboratory of Computer Security Problems of St.Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS). His research interests include distributed system security, and security visualization. He is the author of more than 20 refereed publications.

Andrey Chechulin received his B.S. and M.S. in Computer science and computer facilities from Saint-Petersburg State Polytechnical University and PhD from St.Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS) in 2013. In 2015 he was awarded the medal of the Russian Academy of Science in area of computer science, computer engineering and automation. At the moment he holds a position of leading researcher at the Laboratory of Computer Security Problems of SPIIRAS. He is the author of more than 70 refereed publications and has a high experience in the research on computer network security and participated as an investigator in several projects on developing new security technologies. His primary research interests include computer network security, intrusion detection, analysis of the network traffic and vulnerabilities.

Igor Kotenko graduated with honors from St.Petersburg Academy of Space Engineering and St. Petersburg Signal Academy. He obtained the Ph.D. degree in 1990 and the National degree of Doctor of Engineering Science in 1999. He is Professor of computer science and Head of the Laboratory of Computer Security Problems of St. Petersburg Institute for Informatics and Automation. He is the author of more than 350 refereed publications, including 12 textbooks and monographs. Igor Kotenko has a high experience in the research on computer network security and participated in several projects on developing new security technologies. For example, he was a project leader in the research projects from the US Air Force research department, via its EOARD (European Office of Aerospace Research and Development) branch, EU FP7 and FP6 Projects, HP, Intel, F-Secure, etc. The research results of Igor Kotenko were tested and implemented in more than fifty Russian research and development projects.