Core Method for Community Detection

A. A. Chepovskiy\textsuperscript{1}, S. P. Khaykova\textsuperscript{2}, and D. A. Leshchev\textsuperscript{3}

\textsuperscript{1} Department of Applied Mathematics, HSE Tikhonov Moscow Institute of Electronics and Mathematics (MIEM HSE), Myasnitskaya Street 20, Moscow 101000, Russian Federation
c4hapa@gmail.com
\textsuperscript{2} MIEM HSE, Moscow, Russia
\textsuperscript{3} Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia

Abstract. The processing of networks of interacting objects makes it possible to solve topical issues in the modern world of identifying opinion leaders and channels for the dissemination and exchange of information. In this work, the structure of networks of interacting objects and their possible analysis with the help of weighted graphs based on the interaction of elements of such networks are considered. At the beginning of the work, a methodology for working with a weighted graph was proposed. It is called by the authors the “core method” and provides an algorithm for the analyst’s actions to identify communication groups, opinion leaders and disseminate information in the network. The key concepts of the $\gamma$-core of the graph, the interaction coefficients and the density of communities and the core are introduced. In the second part of the work, the main capabilities of the software developed by the authors, which allows the operator to carry out the procedures required for the method, visualize the results and export the obtained data, are presented. The third part shows the application of the “core method” on a weighted graph, based on the data about the coverage of the activities of the Moscow city authorities in the fight against the new coronavirus infection Covid-19 imported from Twitter. This example shows how opinion leaders on a weighted graph can be identified using the core method and the implemented application.

Keywords: Social network analysis · Algorithms for network analysis · Networks visual representation · Community detection

1 Theory

1.1 About Revealing Communities and Key Applied Tasks

The problem of identifying implicit communities in networks of interacting objects has been covered in many publications over the past 20 years [1–7]. Various algorithms have been developed to solve problems related to this topic. Separately, it is possible to highlight the currently relevant area of analysis of social networks [8–10]. When working with social networks graphs, the key applied problems are the following:

1. the task of determining the proximity of user profiles, the coincidence of their interests, the degree of (face-to-face) acquaintance;
2. the task of recognizing opinion leaders;
3. the task of identifying channels for the distribution and exchange of information between users.

To solve these problems, one must first decide what kind of graph should be built when importing data from the original source. There is usually no point in working with the graph of the entire social network due to its large size. Therefore, as a rule, a subgraph, the construction of which is carried out using a breadth-first search from a set of vertices given in advance, is unloaded. We will work with weighted graphs $G(V, E)$, the set of vertices $V$ of which consists of the original objects – users of the social network. In this case, the weight on the set of edges $E$ is given by the function $w$ with nonnegative values and corresponds to the degree of intensity of interaction of objects with each other. The method for determining the values of $w(e)$ depends on the specific source network. The weight $w(v)$ of the vertex $v$ is then defined as the sum of the weights of all edges incident to it. And let the weight of a given set of vertices be defined as the sum of the weights of all these vertices. The concept of a community is defined in many works [11–15]. Community $S$ is a subgraph (containing set of vertices), with the density of edges between them higher than in the whole graph. In this work, we assume that the communities do not overlap, i.e. after the selection of communities, each vertex is in a single community. Therefore, the community weight $w(S)$ is determined accordingly.

We also define the concept of the internal weight of a community $w^*(S)$ – the sum of the weights of the edges, both vertices of which lie inside the community and the internal weight of the vertex $w^*(v)$ – the sum of the weights of the edges incident to it that lie in the community of the given vertex. For the tasks listed above, you can build the corresponding weighted graphs: graphs of general similarity of users; graphs of user sympathy; graphs of information interaction of users.

To solve the first task and build a graph of general similarity of users, even an unweighted graph of mutual friends or subscriptions is often sufficient. When solving this problem, a weighted graph, the values of the weights of the edges of which are determined based on the general attributes of the original network, can also be used [16]. To basically solve Task 2, both the user sympathy graph and the graph of their information interaction can be used. These weighted graphs are based on the values of the network attributes. But for a qualitative analysis of the graph of information interaction, it is necessary to perform several procedures, which are described further in our work. This allows you to qualitatively solve tasks 2 and 3.

The high-quality construction of the specified graphs requires the selection of parameters for the weight of the edges, depending on the initial network of object interaction. For example, when importing data from the Twitter network, it is possible to use information about existing likes, retweets, comments, user subscriptions. These types of interactions will constitute many attributes. In this case, one of the options for constructing weights on the edges between two vertices is to calculate a weighted sum for these vertices based on the attributes.
1.2 Removing “Garbage” Vertices and Allocating the Core

If we consider the original graph \( G(V, E) \) and apply popular methods of community revealing to it, then the picture will usually be distorted by a large number of leaf vertices obtained during data import. Other vertices, for which the weight of incident edges is significantly lower than the others in the graph, are also possible. Typically, these will be vertices of users who minimally interact with the rest. For graphs of information interaction, these users are not interesting. We will conditionally call such vertices as “garbage”. In contrast, the key vertices of opinion leaders, which have a lot of weight, as well as the structure of communities, including the “heaviest” of them. Revealing such vertices is one of the important tasks, because around them “heavy” communities are formed, and other vertices are attracted. It will be more accurate to say that a vertex \( v \) will be called \( \delta \)-garbage if its weight is less than \( \delta \). Then the set \( \text{Junk}_\delta(G) \) of all garbage in the graph is defined as follows:

\[
\text{Junk}_\delta(G) = \{ v \in V | w(v) < \delta \}
\]  

(1)

Mirror situation takes place for vertices with large weight value. Let us call \( \alpha \)-star or simply a “star” such vertices \( v \) of the graph \( G \), that \( v \) has a weight greater than some value \( \alpha \). The set of stars \( \text{Star}_\alpha(G) \) is then defined as follows:

\[
\text{Star}_\alpha(G) = \{ v \in V | \omega(v) > \alpha \}
\]  

(2)

Such vertices attract other to their communities, unless they, in turn, are in communities with a significant total weight. In this case, the weight of the edges between adjacent vertices from different communities will be important [17]. One of the goals of the core method is to select the key core community (or several such communities) that has the greatest weight among the other communities in graph \( G \). Further, for a given partition, we denote the community with the maximum weight by \( \text{Core}(G) \), and its weight by \( w(\text{Core}(G)) \). It may happen that in the initial graph several communities \( S_i \) are revealed, whose weight is very close or even coincides with the maximum weight \( w(\text{Core}(G)) \). In these cases, we can talk about the presence of several communities-cores in the graph \( G \). The admissible degree of proximity of these values is denoted by \( \gamma \). We define the \( \gamma \)-core \( \text{Core}_\gamma(G) \) as the set of vertices from those communities that satisfy the following relation:

\[
\text{Core}_\gamma(G) = \left\{ v \in V | \exists v \in S_i : \frac{w(S_i)}{w(\text{Core}(G))} > \gamma \right\}
\]  

(3)

In addition to cores and stars, the graph often contains other smaller communities, the connection inside which is quite dense, and the weight is high enough that cores and other large communities cannot absorb these smaller communities.

1.3 Graphs of Information Interaction

It is possible to simplify the graph analysis task by ignoring weak interactions of the original objects by removing garbage vertices. This can be done in two ways. The first
is simply by removing garbage vertices and then the remaining from them edges. But we will use another method: for all edges having a weight less than a given $\beta$ we will consider this weight equal to zero, i.e. remove such edges and obtain a new set of edges $E'$. After that, some of the vertices will become isolated and can be removed from the analyzed graph. Thus we get a new set of vertices $V'$. Let us denote the graph obtained after these operations as $G' \left( V', E' \right)$ – the graph of active information interaction of network objects.

The second method is better suited for forming such a graph, because by taking $\beta$ equal to $\delta$, we can remove not only $\text{Junk}_\delta(G)$, but other inactive edges and vertices from the graph.

We define the interaction coefficient $k_{\text{int}}(G')$ as the ratio of the doubled number of edges to the square of the number of vertices in the resulting graph of active information interaction $G'$:

$$k_{\text{int}}(G') = \frac{2|E'|}{|V'|^2} \quad (4)$$

It is easy to see that in the complete graph $k_{\text{int}}(G') = 1 - \frac{1}{n}$, which for large n is close to 1. But graphs of networks of interacting objects are sparse, so usually this coefficient will be closer to 0. High values of $k_{\text{int}}(G')$ will indicate a significant connection between actively interacting vertices. This indicator is generally responsible for the activity within the analyzed graph. Similarly, you can determine the coefficients of interaction within individual communities of the graph. For the core case, high values will indicate a high level of subjectivity of the graph content [17].

Let us define $k_S(G')$ – the density coefficient of the community $S$ as the ratio of the total internal weight of the vertices of this community to the doubled weight of the graph edges, which is equal to the ratio of the weight of the edges within the community to the weight of the edges of the graph:

$$k_S(G') = \frac{\sum_{v \in S} w^*(v)}{2 \sum_{e \in S} w(e)} = \frac{\sum_{e \in S} w(e)}{\sum_{e \in E'} w(e)} \quad (5)$$

Similarly we define $k_{\text{Core}}(G')$ – the density coefficient of the $\gamma$-core as the ratio of the total weight of the vertices of the $\gamma$-core $\text{Core}_{\gamma}(G')$ to the doubled total weight of the edges of the graph $G'$:

$$k_{\text{Core}}(G') = \frac{\sum_{v \in \text{Core}_{\gamma}(G')} w(v)}{2 \sum_{e \in E'} w(e)} \quad (6)$$

High values of $k_{\text{Core}}(G')$ show, that the core in such a graph plays a significant role in comparison with other communities and garbage vertices. Based on these coefficients, the classification and methodology for working with graphs will be built.

### 1.4 Meta-vertices and Meta-graph

Sometimes, to analyze the graph of interacting objects, it may be interesting to consider communities as meta-vertices and work with this new meta-graph (we will denote such
a meta-graph as $G_{1,1}$ for the first iteration). Then, with each iteration, the number of vertices will decrease: each meta-vertex in the new graph is a group of vertices of the previous graph. Consequently, the meta-vertex is itself a certain graph, within which it is useful to apply the algorithm and analyze the selected communities.

After revealing communities in the meta-graph and analyzing their connections with each other, we get new meta-vertices. Repeating the operation of forming meta-vertices in this way, we obtain the graph $G_{2,1}$. Similarly, at the $i$-th step, the graph $G_{i,i}$ is obtained. Thus, at one of the iterations, you can get a general view of the interaction of the largest groups of users. Of course, this makes sense for a large initial graph $G$.

1.5 Core Method

Let us present the sequence of analysts’ actions to solve the main tasks 2 and 3. It is this technique that we will call the “core method”. Acting according to the algorithm presented below, the analyst will be able to identify both opinion leaders and ways of disseminating information within and between communities.

1. **Remove isolated vertices.** The presence of such may be due to the peculiarities of the source network and the data import process. We get the graph $G$.

2. **Calculate the initial interaction coefficient of the graph** $k_{ini}(G)$. This will be important as a reference point for a graph without isolated vertices.

3. **Remove garbage vertices.** To do this, it is necessary to determine the value of $\beta$ for which to perform the operation of removing edges. We get the graph $G'$.

4. **Calculate the updated interaction coefficient of the graph** $k_{int}(G')$. The recommended variation range is within the following limits: $0, 8 < \frac{k_{int}(G')}{k_{ini}(G)} < 0, 9$. In case the coefficient has changed outside this range, we recommend returning to step 3 and making it with a different value of $\beta$.

5. **Apply algorithm to reveal communities.** It is supposed to use an algorithm that identifies not overlapping communities. For example, variations of the algorithms Infomap [18, 19], Louvain [20] etc. can be used.

6. **Identify stars.** Choose the value of $\alpha$ and select the set of vertices $Star_\alpha(G)$, consisting of the stars of the graph. Check that stars are highlighted in the largest communities and such communities have a high $k_{Si}(G')$. If not, then change the $\alpha$.

7. **Detect the core.** Determine $\gamma$ and compose $Core_\gamma(G')$.

8. **Generate meta-graph.** Create a graph of meta-vertices $G_{1,1}$. Reveal communities inside $Core_\gamma(G)$ and other key meta-vertices, study their structure in accordance with the initial tasks of the researcher. If necessary, continue working with each meta-vertex separately, passing for them recursively to step 2.

9. **Create a meta-meta graph.** Consider the structure of $G_{2,1}$ in accordance with the original tasks.
2 Tool

The authors of this work have designed and implemented a special tool for graph analysis that supports automatic revealing of communities through built-in algorithms and visualization of the result, as well as other actions that allow implementing the previously indicated methodology.

Since the graph of interacting objects can be obtained from any social network and other sources, it was extremely important to fix a universal graph representation format that allows storing the attributes of its vertices and edges. A special XML-like unified markup format called AVS is used. The description of the vertices and edges of the graph is written to the AVS format file: their names, attribute names, data format in attributes. The description of the graph is followed by the description of each vertex and each edge.

An important feature of this AVS-format is its way of storing the attributes of the edges and vertices of the graph, which are large texts. In the AVS file itself, only the link to the file (or part of the file) is stored in the field of the corresponding attribute. And the text is stored in the file, which allows you to analyze texts separately, as well as perform data compression if necessary. Therefore, when exporting data from the developed application, the graph itself (vertices, edges and their attributes) is saved in AVS file, and user texts are stored in separate yaml files for each of them.

The developed software allows user to implement the following user scenarios:

- load an AVS file with a graph for visualization and point-by-point consideration of its vertices and connections;
- apply to the loaded graph any of algorithms to reveal communities available in the application and get acquainted with the statistics of the resulting partition;
- analyze partitions, including using meta-vertices, in order to change communities and/or weight functions;
- export the resulting set of communities for analysis or presentation of results in other systems or manually.

The graph uploaded by the user is displayed on the scene of the main application window (Fig. 1). Scene – is an area of the main application window used to display the graph and interact with its components.
Dividing the graph into communities is one of the most essential tools for their analysis. In the implemented application, two algorithms to reveal communities are integrated: Infomap and Louvain.

After applying the partition, the vertices of the graph belonging to the same community will be colored in the same color, each vertex will be added an attribute with its community number. Then vertices are arranged by revealed communities (Fig. 2).

In addition to these basic actions, this tool allows you to perform those that are important for presented methodology. One can remove edges with a predicate weight, remove isolated vertices, compose meta-vertices, define stars and select the core (using the statistics menu). To determine $\beta$ and remove “garbage” vertices, it is possible to hide the edges of the graph on the scene, according to the user-specified condition, and later either remove them completely from the graph or cancel the hiding and return the hidden edges to the scene. This functionality provides analysts with the ability to visually evaluate the graph that will be obtained for different $\beta$ and choose its optimal value. The user can specify a condition for hiding edges not only by weight, but also by any other attribute of the edge, if there are any in the graph $G$. To do this, you must specify the edge attribute, sign and constant value for comparison, as well as the type of comparison. The tool also provides the ability to remove isolated vertices, which is especially useful after removing edges, since, with the optimal choice of $\beta$, some of the vertices will become isolated and their removal will help to finally clear the graph of “garbage”. To create a meta-graph, the “Make meta-vertices” button is implemented, which initiates the creation, based on the current scene, of meta-vertices corresponding to the communities selected at the current step.

To create a graph of meta-vertices, first, for each community, the total weight of edges (with both vertices within the same community) is calculated. This gives the weight of the meta-vertices used to calculate the radius for displaying on the main scene. Then, for each pair of communities, the total weight of the edges between the vertices of these communities is calculated. This is how the weights of the new edges in the meta-graph are determined.

When analyzing a meta-graph, it may be relevant for a user to study the structure of a community representing one meta-vertex. To do this, the user can “fall” into the meta-vertex and see the subgraph of the original graph, composed only of the vertices of the given community. With this view of the subgraph, the user can save and open the subgraph as a separate project and work with it using the full functionality of the application. The statistics menu implemented in the application allows the user to study the topological indicators of the graph and its components (vertices, communities). Basic statistics, located in the text boxes in the lower left corner, will help user to calculate the graph interaction ratio needed in the early stages of the core method, as well as the community and $\gamma$-core density factors for later stages. Communities, sorted by the number of vertices included in them (the value is indicated in parentheses after the name of the community), will help user to quickly find the largest one and highlight the core.
3 An Example of Applying the Method on Data from Twitter

3.1 The Core Detection

As an example, in the evening of 05/27/2020, 8 relevant posts, regarding the actions of the Moscow city authorities in the fight against the new coronavirus infection Covid-19, were downloaded from Twitter. As well as related comments, likes, retweets. Among the posts were both the official statements of the city leadership in the media and their accounts, and highly social messages of a provocative nature from opposition-minded individuals. The download took place with the generation of a weighted graph based on interaction with the original posts, as well as other actions previously performed by users. Based on each of the interactions of users with each other (subscription, like, comment, retweet), the weight of the edges between them was formed.

First, a graph with 632 vertices and 1002 edges was obtained. Further, acting according to the previously described core method, the following actions were performed. First, isolated vertices were removed, and 459 vertices remained with 1002 edges, this will be graph $G$. Indicators of graph $G$: mean edges weight $= 2.767$, mean vertex degree $= 1.585$, max vertex degree $= 126$, $k_{int}(G) = 0, 0095$. Then we go to step 3 and “remove garbage vertices”: choosing the value $\beta = 1$, remove 249 edges with weights not exceeding $\beta$ and 34 vertices (which became isolated after removing edges), we get the graph $G'$. We calculate the coefficient $k_{int}(G') = 0, 0083$. The changes can be estimated as follows: $\frac{k_{int}(G')}{k_{int}(G)} = 0, 87$, which is in the recommended range. The total weight of the edges: $\sum_{e \in E'} w(e) = 2773$.

Next, we apply the Infomap algorithm to the graph $G'$ to reveal implicit communities. We get 43 communities (Fig. 2), 8 of which contain more than 15 vertices, and calculate the main indicators (Table 1). Four communities: $S_0, S_1, S_2$ and $S_4$ have both a high density coefficient and a high maximum internal degree. This indicates the presence of stars and active interaction within these communities. Communities $S_5$ and $S_7$ may also contain stars, while communities $S_3$ and $S_6$ rather do not have stars in their composition. However, the density of $S_3$ community is high.

Let’s look at the vertices of the graph $G'$ with the maximum weights (Table 2). Mean weighted vertex degree in $G'$ is 12, 08. The last column, obtained as the weight of the vertex divided by this value, shows well the stars-vertices with a given indicator above 14. Therefore, we will take $\alpha = 170 > 12, 08 \times 14 = 169, 12$.

Thus, we have found $Star_\alpha(G)$, and for $\alpha = 170$ this set consists of 5 vertices. The community $S_0$ has the greatest weight, so we take $Core(G) = S_0$, and define $\gamma = 0, 77$. Then, according to (3) $Core_\gamma(G')$ contains $S_0, S_1, S_2$ and $S_4$. We calculate $k_{Core_\gamma}(G') = \frac{2195}{2 \times 2773} = 0, 404$. A high value obtained indicates a correctly found core.

Now we can generate the meta-graph $G'_{\gamma,1}$. Further, inside the key meta-vertices $G'_{\gamma,1}$, including those from the $Core_\gamma(G')$, you can look at their structure from the inside.

3.2 The Structure of Meta-Vertices

Consider the partition into internal communities in $S_1$ (Fig. 3). This meta-vertex is a source of information – a star-vertex corresponding to the official media account and its
adjacent vertices. We will call such meta-vertices “constellations of the first kind”, and “planets” – its adjacent vertices. Thus, in our classification, \( S_1 \) is a constellation of the first kind, consisting of one vertex-star and 48 vertices-planets.

Let us consider in more detail the meta-vertex corresponding to \( S_0 \in \text{Core}_y(G) \). Let’s reveal internal communities in \( S_0 \) using Infomap algorithm (Fig. 4). The top-star is

**Table 1.** Communities with at least 15 nodes

| \( S_i \) | \(| S_i |\) | \( w(S_i) \) | \( w^*(S_i) \) | \( k^*_v(G') \) | \( \max_{v \in S_i} w(v) \) | \( \max_{v \in S_i} w^*(v) \) |
|---|---|---|---|---|---|---|
| \( S_0 \) | 44 | 710 | 382 | 0.068 | 445 | 187 |
| \( S_1 \) | 49 | 576 | 408 | 0.073 | 315 | 204 |
| \( S_2 \) | 33 | 537 | 296 | 0.053 | 258 | 129 |
| \( S_3 \) | 25 | 444 | 288 | 0.051 | 60 | 30 |
| \( S_4 \) | 44 | 372 | 306 | 0.055 | 171 | 128 |
| \( S_5 \) | 24 | 340 | 170 | 0.03 | 171 | 83 |
| \( S_6 \) | 24 | 337 | 180 | 0.032 | 64 | 27 |
| \( S_7 \) | 22 | 167 | 134 | 0.024 | 84 | 61 |

**Table 2.** Vertices of \( G' \) with top weight

| Encrypted vertex nick | Vertex degree | Weighted vertex degree | Weighted vertex degree in community | Weighted degree divided by mean |
|---|---|---|---|---|
| v_***ov | 126 | 445 | 187 | 37.08 |
| le***al | 84 | 315 | 204 | 26.25 |
| Vi***va | 76 | 258 | 129 | 21.5 |
| Mr***ay | 65 | 171 | 128 | 14.25 |
| Pr***at | 62 | 171 | 83 | 14.25 |
| Ol***13 | 27 | 84 | 61 | 7 |
| aa***an | 38 | 64 | 27 | 5.33 |
| 8a***Wn | 20 | 60 | 30 | 5 |
| ma***n_ | 25 | 57 | 29 | 4.75 |
| Se***us | 20 | 49 | 23 | 4.08 |
| ru***60 | 31 | 42 | 15 | 3.5 |
| NnA***36 | 13 | 41 | 18 | 3.41 |
| dj***ef | 12 | 40 | 15 | 3.33 |
clearly visible – this is a pronounced influencer and the adjacent users who had interaction with the original posts. It is worth noting that the composition of many of these users is ambiguous: basic nicknames of 15 random characters, which are created upon registration, the number of followers is zero or close to zero, photos are uploaded mostly without a face. Presumably, such users are bots or fakes of the respective influencer. Of course, there are also real users who share the leader’s views. They can even form their own communities, in this case there are two of them, but both of them consist of 2 vertices. Further there will be examples where additional communities are larger.

We will call such meta-vertices “constellations of the second kind”, “stars” in them – opinion leaders, and “planets” – other vertices (in general, not all of them will be adjacent to a star). Thus, in our classification $S_0$ is a constellation of the second kind, consisting of one vertex-star, with which 43 vertex-planets are connected.

Communities $S_2$ and $S_4$ also represent constellations of the second kind (Figs. 5, 6) consisting of one star, 32 and 43 planetary vertices, respectively. Other vertices, according to the value of $\alpha$ are not stars here.

Community $S_3$ is a “constellation of the third kind” (Fig. 7), there is no star here, but the density value $k_{S_3}(G')$ is quite high, the vertices are still competing for supremacy in this group. This means that with the further development of the social network over time, a star will appear here.
These constellation variants are not unique for the considered communities and can be found in other cases as well, for example, in the structure of communities $S_5$, $S_6$ and $S_7$. There will also be one of the three previous pictures. Among them, the star is only $S_5$ for the taken value of $\alpha$.

Thus, this technique distinguishes opinion leaders and communities, constituting support groups for the respective leaders. It should be noted that the qualitative analysis identified leaders of various opinions, both pro-government (Fig. 9) and opposition (Fig. 10).

Then we repeat the selection of the community in the graph of meta-vertices $G'_1$ and obtain the graph of meta-meta-vertices $G'_2$. If the graph $G'_1$ can be conventionally called a “constellation graph”, then the graph $G'_2$ – is a “galaxy graph” (Fig. 8). Five communities are distinguished on $G'_2$. Only one of them includes more than 1 meta-vertex. It consists of 39 meta-vertices and is the “Galactic core” for the initial graph $G'$. All the other 4 are composed of single meta-vertex, each of which, in turn, consists of several vertices of the original graph. It should be noted that there could be more interesting “graphs of galaxies”.

---

**Fig. 7.** Internal structure of community $S_3$

**Fig. 8.** $G'_2$ – the graph of meta-meta vertices with revealed communities

**Fig. 9.** Star vertex @Vi***va

**Fig. 10.** Star vertex @Mr***ay
4 Conclusions

This paper describes a core method that allows you to analyze weighted graphs and solve the problem of identifying opinion leaders and ways of disseminating information. Mathematical indicators for a weighted graph, which make it possible to assess the degree of interaction of network vertices, have been introduced. The functionality of the software implemented by the authors, which allows analysts to work within the framework of the suggested method, is described. An example of work by the method using the described application with a graph built on the basis of real data from the Twitter network is given and considered in detail.

The authors see a possible further development in this area in testing the hypothesis of self-organization of networks of interacting objects in the formation of implicit communities in the process of graph evolution based on interaction according to laws similar to the laws of physics and with the aim of bringing the system into a state of stable equilibrium.

References

1. Aggarwal, C.: Social Network Data Analytics. Springer, US (2011)
2. Chepovskiy, A., Lobanova, S.: Combined method to detect communities in graphs of interacting objects. Bus. Inf. 42(4), 64–73 (2017)
3. Coscia, M., Rossetti, G., Giannotti, F., Pedreschi, D.: Demon: a local-first discovery method for overlapping communities. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 615–623, ACM (2012)
4. Karimi, F., Lotfi, S., Izadkhah, H.: Multiplex community detection in complex networks using an evolutionary approach. Expert Syst. Appl. 146, 113184 (2020)
5. Lambiotte, R., Rosvall, M.: Ranking and clustering of nodes in networks with smart teleportation. Phys. Rev. E 85, 056107 (2012)
6. Newman, M., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69(2), 1–15 (2004)
7. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435, 814–818 (2005)
8. Lei, T., Huan, L.: Community detection and mining in social media. Synthesis Lectures on Data Mining and Knowledge Discovery, p. 137 (2010)
9. Xie, J., Szymanski, B.K., Liu, X.: SLPA: uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In: 2011 IEEE 11th International Conference, Data Mining Workshops (ICDM), pp. 344–349 (2011)
10. Yang Bo, Liu Dayou, Liu Jiming.: Discovering communities from social networks: methodologies and applications. Handbook of Social Network Technologies and Applications, pp. 331–346. Springer, Boston (2010)
11. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70, 066111 (2004)
12. Fortunato, S.: Community detection in graphs. Phys. Rep. 486(3), 75–174 (2010)
13. Girvan, M., Newman, M.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. 99(12), 7821–7826 (2002)
14. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. Phys. Rev. E 80, 056117 (2009)
15. Radicchi, F., Castellano, C., Loreto, V., Cecconi, F., Parisi, D.: Defining and identifying communities in networks. Proc. Natl. Acad. Sci. U.S.A. 101(9), 2658–2663 (2004)

16. Leschyov, D.A., Suchkov, D.V., Khaykova, S.P., Chepovskiy, A.A.: Algorithms to reveal communication groups. Voprosy kiberbezopasnosti. 32(4), 61–71 (2019). https://doi.org/10.21681/2311-3456-2019-4-61-71

17. Voronin, A.N., Kovaleva, J.B., Chepovskiy, A.A.: Interconnection of network characteristics and subjectivity of network communities in the social network Twitter. Voprosy kiberbezopasnosti. 37(3), 40–57 (2020). https://doi.org/10.21681/2311-3456-2020-03-40-57

18. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proc. Natl. Acad. Sci. U.S.A. 105(4), 1118–1123 (2008)

19. Rosvall, M., Bergstrom, C.T., Axelsson, D.: The map equation. Eur. Phys. J. Spec. Top. 178(1), 13–23 (2009)

20. Blondel, V., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large network. J. Stat. Mech. Theory Exp. 10, P10008 (2008)