DOI 10.1007/s10958-015-2607-y
Journal of Mathematical Sciences, Vol. 211, No. 3, December, 2015

AN ALGORITHM FOR DETECTING COMMUNITIES IN SOCIAL NETWORKS

M. I. Kolomeychenko, A. A. Chepovskiy, and A. M. Chepovskiy

UDC 004.421.2:519.178

Abstract. In this paper, we propose an algorithm to find subgraphs with given properties in large social networks. A computational experiment that confirms the effectiveness of the proposed algorithm is presented.

1. The Detection of Communities as a Problem of Graph Analysis

The key point is the recognition of hidden structures in real social networks, which is necessary to analyze the organization of complex networks [1, 3]. Complex systems are usually built in blocks that have specific roles and functions. Such blocks become a set of vertices with high density of internal links and at the same time low density of external links in the network representation. These blocks are called communities or modules and are commonly used in various structures of interrelated objects [5, 6].

Suitable representation may help to extract important information about the network, simplify perception and emphasize hidden structural characteristics of linked objects. Figure 1 shows an example of network partitioned into communities with high density of internal links and low density of external links between communities.

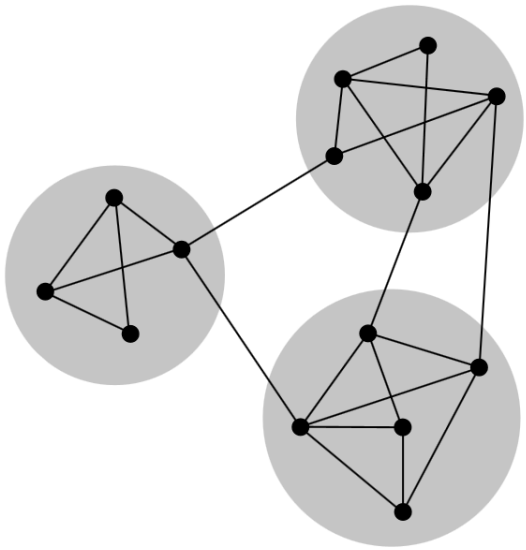

Fig. 1. An example of network partitioning.

Translated from Fundamentalnaya i Prikladnaya Matematika, Vol. 19, No. 1, pp. 21–32, 2014.

The detection of communities is a topical problem in graph theory. We consider the problem of finding a partitioning of vertices into communities for a given multigraph that minimizes the given target function.

There are many algorithms that use methods of different disciplines such as physics, biology, informatics, applied mathematics, and sociology [1, 3, 5, 6]. However, not all of them are reliable and may be applied for real problems.

Here are some algorithms proposed for the detection of communities before.

In the Girvan–Newman algorithm of hierarchical partition [6, 17], links are removed iteratively based on the value of one metric for links or vertices [1, 16]. In the most widespread realization, we use the links metric or "betweenness centrality" [1], defined as the number of shortest paths between all pairs of vertices passing through a given link:

$$\mathrm{betw}(l) = \sum_{s \neq t,\ s \in V,\ t \in V} \frac{\sigma_{st}(l)}{\sigma_{st}}, \tag{1}$$

where l is the given link, V is the set of graph vertices, σ_{st} is the total number of shortest paths between vertices s and t, and $\sigma_{st}(l)$ is the total number of shortest paths between vertices s and t passing through the link l.

In the most widespread realization, the link removing procedure stops when the modularity of the final partitioning reaches the maximum [15]:

$$Q = \sum_{i,j} \left[\frac{A_{ij}}{2m} - \frac{k_i k_j}{4m^2} \right] \sigma(c_i, c_j), \tag{2}$$

where m is the number of links, A is the adjacency matrix, k_i is the degree of the vertex i,

$$\sigma(c_i, c_j) = \begin{cases} 1, & c_i = c_j, \\ 0, & c_i \neq c_j, \end{cases}$$

and c_i is the number of classes where the vertex i belongs.

The Newman–Girvan modularity (2) determines the quality of network partitioning into communities.

Also we use a fast greedy modulatory optimization algorithm (Clauset, Newman and Moore [4]). This method is an accelerated realization of a previous technique proposed by Newman [14]. Starting with the set of isolated vertices, links of original graph are added iteratively to maximally increase the metric of modularity described above.

The fast method of modularity optimization (Blondel [2]) involves a multistage procedure based on the local Newman–Girvan modularity optimization for adjacent vertices. Super-vertices replace communities after graph partitioning, and as a result we get a network with lesser vertex degrees. This procedure continues iteratively until it is impossible to increase the modularity.

In addition to the mentioned algorithms, there are several popular approaches for the detection of communities in network structures [2, 7, 13, 18]. However, results of tests in [10] show that most of these algorithms cannot be applied for big networks due to low speed characteristics or low quality of community detection.

In this paper, we propose a method to effectively detect communities in a network. It is based on a modification of the algorithm for finding communities. This modification uses a graph modularity metric proposed in [8, 19–21].

2. A Technique for the Detection of Communities

The proposed method assumes data preprocessing that derives a graph from the given multigraph. Multiple links are replaced by one with weight equal to the number of links between the pair of vertices.

The problem of the detection of significant structures is comparable with the problem of data compression.

Let us consider the problem of obtaining unique names for vertices. A simple method to obtain unique names for vertices is to use Huffman coding, which provides data compression by assigning the shortest codewords to the most frequent events or objects and long codewords to less frequent ones. After that for a general network representation it is necessary to separate important structural information from insignificant details. The method relies on two-level partitioning of information. The method uses unique names for more common objects (communities) and duplicate names for minor details (the notation of vertices in each module).

A two-level description allows us to encode a random walk with less information than a single-level coding scheme. Furthermore, this approach helps to derive the network structure based on the assumption [12] that, in general, a statistically random walk will spend more time inside the vertex groups than for moves from group to group.

Thus, community partitioning is described by the upper level of a two-level coding. The basic idea is that there is no need to directly perform a two-level coding. It is enough just to count the upper limit of the codeword length that is needed to encode the vertices. The process of network partitioning into communities will minimize this metric.

Let M be a partitioning of n vertices into m modules:

$$\alpha = 1, \ldots, n, \quad i = 1, \ldots, m.$$

Let $L(M)$ be an upper bound of the codeword, serving as the quality of the derived partitioning metric. To calculate the metric $L(M)$ for a given partitioning M, first, we apply Shannon's source coding theorem, which states that when n codewords are used for the description of n states of a random variable X occurring with frequencies p_i, the average codeword length cannot be less than the entropy of the random variable X, represented as

$$H(X) = \sum_{i=1}^{n} p_i \log_2 p_i. \tag{3}$$

The formula (3) will be used for metric calculation $L(M)$. We fix a partition M into communities for a given network. Let us introduce the random variable $\mathcal{Q}$ that takes values from 1 to m (the number of communities) with probabilities $q^{(i)}$, where $i = 1, \ldots, m$. Let us consider a random variable $\mathcal{P}^i$ for each community i that takes values from 1 to n_i (the number of vertices in the community i) with probabilities ρ_k^i, where $k = 1, \ldots, n_i$. The calculation of the metric $L(M)$ is based on the calculation of the entropy of the introduced random variables $\mathcal{Q}$ and $\mathcal{P}^i$ in the following way:

$$L(M) = qH(\mathcal{Q}) + \sum_{i=1}^{m} p_i H(\mathcal{P}^i), \tag{4}$$

where

$$q = \sum_{i=1}^{m} q_i$$

is the probability of transition between communities at each step of a random walk, q_i is the probability to leave the community i, p_α is the probability to visit the vertex α, and

$$p_i = \sum_{\alpha \in i} p_\alpha + q_i$$

is the probability to stay in the community i. We have that

$$H(\mathcal{Q}) = -\sum_{i=1}^{m} \frac{q_i}{\sum\limits_{j=1}^{m} q_j} \log_2 \left(\frac{q_i}{\sum\limits_{j=1}^{m} q_j} \right), \tag{5}$$

$H(\mathcal{Q})$ is the entropy of transitions between modules, the lower limit of average codeword length for naming modules;

$$H(\mathcal{P}^i) = \frac{q_i}{q_i + \sum\limits_{\beta\in i} p_\beta} \log_2 \left(\frac{q_i}{q_i + \sum\limits_{\beta\in i} p_\beta} \right) - \sum_{\alpha\in i} \frac{p_\alpha}{q_i + \sum\limits_{\beta\in i} p_\beta} \log_2 \left(\frac{p_\alpha}{q_i + \sum\limits_{\beta\in i} p_\beta} \right), \tag{6}$$

$H(\mathcal{P}^i)$ is the entropy of transitions inside the module i, the lower limit of average codeword length for naming vertices in the module i. It is possible to get a more detailed description of the metric $L(M)$ (formula (4)) from the given formulas (5) and (6):

$$L(M) = \left(\sum_{i=1}^{m} q_i \right) \log_2 \left(\sum_{i=1}^{m} q_i \right) - 2 \sum_{i=1}^{m} q_i \log_2 q_i - \sum_{\alpha=1}^{n} p_\alpha \log_2 p_\alpha + \sum_{i=1}^{m} \left(q_i + \sum_{\alpha\in i} p_\alpha \right) \log_2 \left(q_i + \sum_{\alpha\in i} p_\alpha \right). \tag{7}$$

It is worth noting that $\sum\limits_{\alpha=1}^{n} p_\alpha \log_2 p_\alpha$ does not depend on the network partition into communities. Therefore, when we optimize the network partition, it will be necessary to keep all the changes: q_i is the probability of a random walk moving into and out of communities and $\sum\limits_{\alpha\in i} p_\alpha$ is the part of time that the random walk spends in each of the communities.

The quality metric for an obtained partition can be easily calculated for any partition; its update and recalculation is a high-speed operation.

Any numerical method developed for finding network partitioning that optimizes an objective function may be used for minimizing the metric $L(M)$.

Let us consider the method for undirected weighted networks. The relative weight of vertex α is defined as

$$w_\alpha = \sum_{l\in l_\alpha} w_l, \tag{8}$$

where l_α is the set of incident links for the vertex α and w_l is the weight of the link l. Then, using (8), we can set the relative weight of the community i:

$$w_i = \sum_{\alpha\in i} w_\alpha. \tag{9}$$

Let w_i^{exit} be the weight of exit from community i:

$$w_i^{\text{exit}} = \sum_{l\in l_i^{\text{exit}}} w_l, \tag{10}$$

where l_i^{exit} is the set of links coming out of community i.

We denote the total weight of links connecting modules as

$$w^{\text{exit}} = \frac{\sum\limits_{i=1}^{m} w_i^{\text{exit}}}{2}.$$

If we use undirected weighted networks, then the formulas (8)–(10) transform formula (7), so that the quality metric of the partition $L(M)$ will be as follows:

$$L(M) = w^{\text{exit}} \log_2 w^{\text{exit}} - 2 \sum_{i=1}^{m} w_i^{\text{exit}} \log_2 w_i^{\text{exit}} - \sum_{\alpha=1}^{n} w_\alpha \log_2 w_\alpha + \sum_{i=1}^{m} (w_i^{\text{exit}} + w_i) \log_2 (w_i^{\text{exit}} + w_i).$$

Now we consider the case of oriented weighted graphs. First of all, we need to introduce a new parameter, teleportation probability τ. The teleportation of a random walk is an equiprobable transition

to a random vertex at each time point during a random walk process;

$$p_\alpha = \frac{1}{n},$$

where p_α is the probability to start the random walk from the vertex α.

The probability of transition from a vertex α to a vertex β is defined as

$$w_{\alpha\beta}^{\text{norm}} = \frac{w_{\alpha\beta}}{\sum\limits_{l \in l_\alpha^{\text{out}}} w_l}, \tag{11}$$

where l_α^{out} is the set of vertices coming out of the vertex α, and $w_{\alpha\beta}$ is the weight of a link between α and β. If there is no link coming out from α to β, then $w_{\alpha\beta} = 0$.

At each step of the random walk process, the transition to an adjacent vertex is implemented with the probability $1 - \tau$ and to a random vertex of the network with the probability τ. The probability of moving from the community i, used in formula (7), under the condition that the network is oriented and weighted will be as follows:

$$q_i = \tau \frac{n - n_i}{n} \sum_{\alpha \in i} p_\alpha + (1 - \tau) \sum_{\alpha \in i} \sum_{\beta \notin i} p_\alpha w_{\alpha\beta}^{\text{out}}, \tag{12}$$

where n_i is the number of vertices in community i.

As a generalization of a random walk process, the parameter τ_α, the probability of teleportation to the vertex α, may be introduced, where $\sum\limits_{\alpha=1}^{n} \tau_\alpha = 1$. Then the probability of moving out from the community i (formula (12)) will be as follows:

$$q_i = \tau \left(1 - \sum_{\alpha \in i} \tau_\alpha\right) \sum_{\alpha \in i} p_\alpha + (1 - \tau) \sum_{\alpha \in i} \sum_{\beta \notin i} p_\alpha w_{\alpha\beta}^{\text{out}}.$$

Any greedy algorithm (fast, but not accurate) or the Monte Carlo method (accurate but slow) can be used to minimize $L(M)$.

In [2, 4, 14, 15], a similar method is used only for the structural separation of a network community. Often social networks contain many details [1,3] stored in different data types. Therefore, in the process of detection of communities it is necessary to use the information content of the network to improve the quality of the partition and its detailed adjustment.

Let S be the set of all strings, $T = \{\text{int}, \text{double}, \text{string}\}$ be the set of types, at $= (\text{name}, \text{type})$ be attribute type, where name $\in S$, type $\in T$, $TC = \{\text{at}_i\}$ be the set of attribute types, $D_{\text{at.type}}$ be the set of all allowable values of the attribute type at $\in TC$, $A = (t, \text{value})$ be an attribute, where $t \in TC$, value $\in D_{t.\text{type}}$, and $G = (V, E)$ be a multigraph, where $V = \{V_i\}$ is the set of vertices and $E = \{E_i\}$ is the set of edges such that $E_i \in V \times V$. Let $\varphi\colon TC \to 2^V$ be a map that defines the set of vertices with a given attribute, $\varphi_t\colon \varphi(t) \to D_{t.\text{type}}$, where $t \in TC$ is a map that defines the attribute values at the vertices that contain a given attribute type, $\psi\colon TC \to 2^E$ be a map that defines the set of links with a given attribute, $\psi_t\colon \psi(t) \to D_{t.\text{type}}$, where $t \in TC$, be a map that defines the attribute value at the links that contain a given attribute type. Let ST be a set of attribute types that must be taken into account for detecting communities. Let $\mu_{ST}(v_i, v_j) \in [0; 1]$ be the measure of closeness for two vertices by the given set of attribute types. Then for the case of an undirected graph, the community i weight (formula (9)) is modified as follows:

$$w_i = \left(\sum_{\alpha, \beta \in i} \frac{1 - \mu_{st}(\alpha, \beta)}{|i|^2} \right) \sum_{\alpha \in i} w_\alpha.$$

For the case of a directed graph the probability of a transition from the vertex α to the vertex β (formula (11)) is modified as follows:

$$w_{\alpha\beta}^{\text{norm}} = \frac{w_{\alpha\beta}}{\sum\limits_{l \in l_\alpha^{\text{out}}} w_l}\big(1 - \mu_{st}(\alpha, \beta)\big).$$

The methodology of the algorithm based on the balance between the quality of a network partitioning and the algorithm speed is as follows: adjacent vertices are connected into communities, which, in turn, are connected into super-communities, and so on. Then each vertex is moved randomly to an adjacent community in order to maximally minimize the target function. If it is impossible to minimize the target function, then the vertex remains in the original community. This procedure is repeated until there is no way to move any vertex and reduce the target function. At each step, a random sequence of selected vertices is generated. The algorithm will find a good network partition into communities in a short period of time.

The general scheme of the algorithm can be described as follows.

(1) Initially, each vertex is considered as a separate community. We calculate and memorize the target function $L(M)$. Go to the item (2).
(2) A random walk generates a sequence of vertices. Go to item (3).
(3) Subsets of vertices that are united in communities are chosen according to the frequency of vertex occurrences in the resulting sequence. Go to item (4).
(4) We calculate the metric $L(M)$ for a given partition. If the value decreased, then the partition M is saved and the algorithm's work continues (go to item (2)). Otherwise, if the target function value $L(M)$ did not decrease, go to item (5).
(5) The resulting partition of is considered as the result of the detection of communities in the network.

Detected communities will have the following property: The density of links within communities is much higher than the density of links between communities.

3. Computational Experiments

The testing of the algorithm was carried out in several stages: testing on generated graphs and testing on subgraphs of real social networks.

The first testing stage was conducted on generated graphs. To perform evaluation and test of the algorithm of detection of communities, we need graphs with a given structure of communities, which the algorithm will allocate. We can test the algorithm in different configurations of networks and communities in it with the help of the LFR-model introduced in [9, 11]. We consider the LFR-model as a special case of a planar partition into l communities, in which:

- communities have different size, which is distributed according to a power law. Let l be the number of communities in the network, $n_{\max}$ be the maximum degree of a community (the number of vertices belonging to the community), and $n_i \in \{1, 2, 3, \ldots, n_{\max}\}$ be the degree of the ith community. Then
$$P(n_i = n) = \frac{c_1}{n^{\tau_1}},$$
where $c_1 = \text{const}$, $\tau_1 = \text{const}$;
- vertex degrees are distributed according to a power law. Let N be the number of vertices in the network, $k_{\max}$ be the maximum vertex degree, and $k_i \in \{1, 2, 3, \ldots, k_{\max}\}$ be the degree of the ith vertex. Then
$$P(k_i = k) = \frac{c_2}{k^{\tau_2}},$$
where $c_2 = \text{const}$, $\tau_2 = \text{const}$.

Let us introduce the following notation. Let k_i^{in} be the number of vertices adjacent to the ith vertex that lie in the same community, k_i^{out} be the number of vertices adjacent to the ith vertex that lie in other communities, c be the community of the vertex i, n_c be the number of vertices lying in the community c, k_c^{out} (k_c^{in}) be the number of available links outside (inside) community c, the sum of the vertex degrees outside (inside) of the community, and $\langle k \rangle$ be the average vertex degree:

$$\langle k \rangle = \frac{\sum\limits_{i=1}^{N} k_i}{N}.$$

Then

$$k_c^{\text{out}} \sim (N - n_c)\langle k \rangle, \quad k_c^{\text{in}} \sim n_c \langle k \rangle.$$

The probability of a link of the vertex i with vertices from other communities is

$$p_i^{\text{out}} = \frac{k_i^{\text{out}}}{k_c^{\text{out}}} \sim \frac{k_i^{\text{out}}}{(N - n_c)\langle k \rangle}.$$

The probability of a link of the vertex i with vertices from its community is

$$p_i^{\text{in}} = \frac{k_i^{in}}{k_c^{\text{in}}} \sim \frac{k_i^{\text{in}}}{n_c \langle k \rangle}.$$

A speed test of the algorithm was performed on the generated graphs. We present the algorithm's execution time dependence on the average vertex degree, where the number of vertices is 10 000, the maximum node degree is equal to 40, the probability of a link between vertices from different communities is equal to 10%, the minimum community size is 100, and the maximum community size is 1000 (Fig. 2). All the networks in the test have different average node degree, which in turn affected the amount of links: a graph with average node degree "4" has 53 152 links, and a graph with average node degree "35" has 350 440 links.

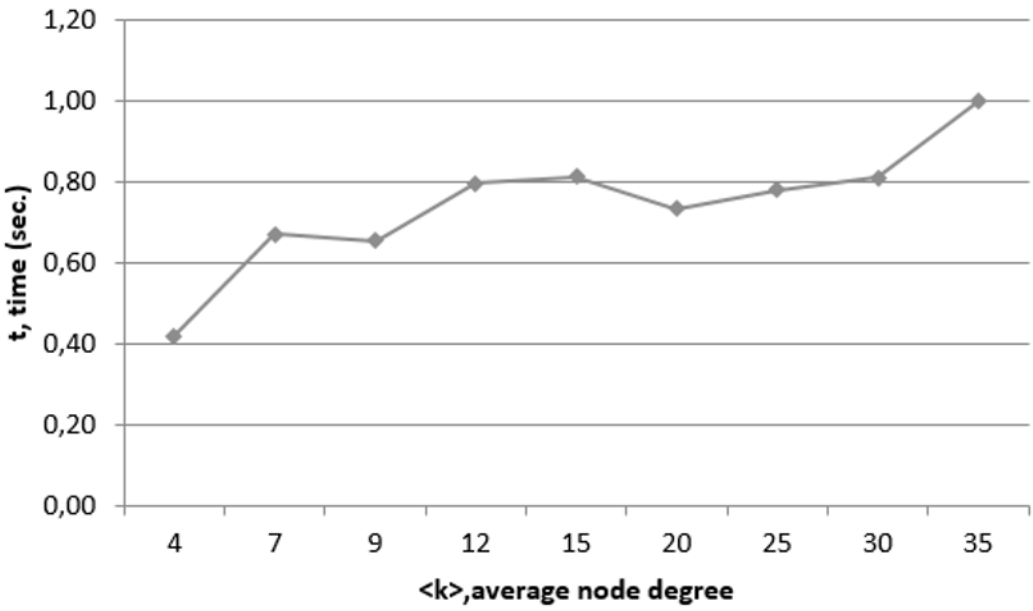

Fig. 2. The algorithm's execution time dependence on the average vertex degree.

Figure 2 shows that the presented dependence can be approximated by a linear function, but the analysis of the asymptotic behavior of the algorithm must take into account the structural features of the

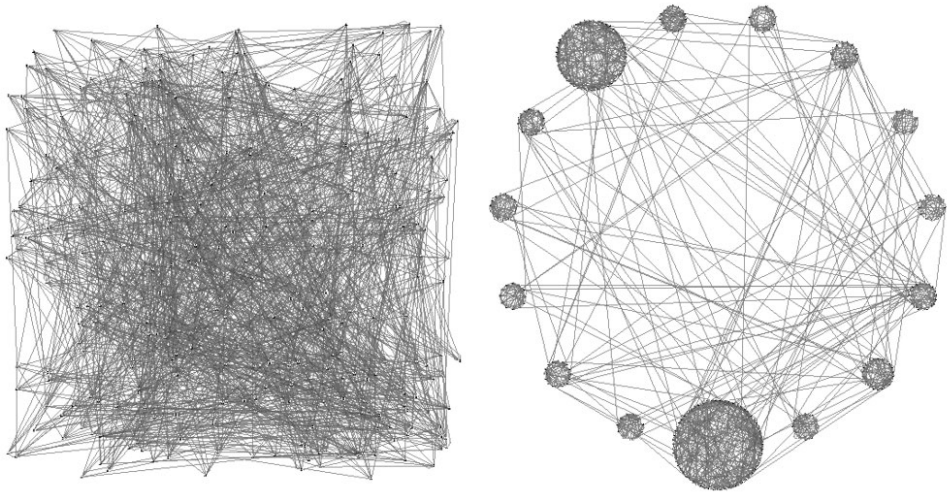

Fig. 3. Before and after the detection of communities. The number of vertices is 300; the number of links is 1357.

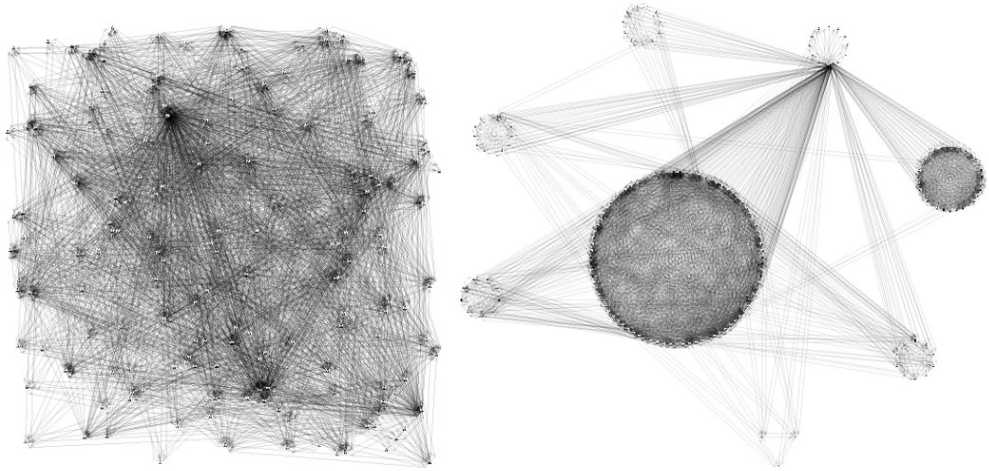

Fig. 4. Before and after the detection of communities. The number of vertices is 172; the number of links is 3026.

particular network. To confirm the linear dependence between the time of the algorithm's execution and the average vertex degree, we need more research on different network configurations.

Let us give examples of analysis of real social network subgraphs. The vertices are profiles of social networks and links are friendship relations between them. The data was taken from the social network *Vkontakte* (vk.com) using *API Vkontakte*.

Figures 3 and 4 show the results of the detection of communities in social networks. In the left part of each of the figures, vertices of the network are located randomly. In the right part, allocations are based on the results of the community detection algorithm. All elements of the same community are located relative to each other on the circular. Communities are also located in this way. The examples below (Figs. 3 and 4) demonstrate the detection of communities of a graph describing "friendship" relations. It is clearly demonstrated that most vertices from one community are linked to each other. At the same time, these vertices have weak links with vertices from other communities. Furthermore, it is understood that there are some objects linked with vertices of all communities in the network.

Therefore, the modification of existing methods proposed in this paper can be applied to big data of real social networks and efficiently (with linear complexity) solves the problem of detection of communities with regard to information content.

REFERENCES

1. T. V. Batura, "Methods of analysis of computer social networks," *Vestn. NGU, Ser. Inform. Technol.*, **10**, No. 4, 13–28 (2012).
2. V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "The Louvain method for community detection in large networks," *J. Stat. Mech. Theory Exp.*, 108–121 (2008).
3. A. N. Churakov, "Social network analysis," *SocIs*, **1**, 109–121 (2001).
4. A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev.*, **E 70**, 066111 (2004).
5. S. Fortunato, "Community detection in graphs," *Phys. Rep.*, **486**, 75–174 (2010).
6. M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proc. Natl. Acad. Sci. USA*, **99**, 7821–7826 (2002).
7. R. Guimera, M. Sales-Pardo, and L. A. N. Amaral, "Modularity from fluctuations in random graphs and complex networks," *Phys. Rev.*, **E 70**, 025101 (2004).
8. R. Lambiotte and M. Rosvall, "Ranking and clustering of nodes in networks with smart teleportation," *Phys. Rev.*, **E 85**, 1103–1012 (2012).
9. A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev.*, **E 80**, 016118 (2009).
10. A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Phys. Rev.*, **E 80**, 056117 (2009).
11. A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev.*, **E 78**, 046110 (2008).
12. L. Lovasz, "Random walks on graphs: a survey," in: D. Miklós, V. T. Sós, and T. Szőnyi, eds., *Combinatorics, Paul Erdős is Eighty*, Bolyai Soc. Math. Stud., Vol. 2, Budapest (1996), pp. 1–46.
13. C. P. Massen and J. P. K. Doye, "Identifying communities within energy landscapes," *Phys. Rev.*, **E 71**, 046101 (2005).
14. M. E. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev.*, **E 69**, 066133 (2004).
15. M. E. Newman, "Modularity and community structure in networks," *Proc. Natl. Acad. Sci. USA*, **103**, 8577–8582 (2006).
16. M. E. Newman, *Networks*: *An Introduction*, Oxford Univ. Press, Oxford (2010).
17. M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev.*, **E 69**, 026113 (2004).
18. F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proc. Natl. Acad. Sci. USA*, **101**, 2658–2663 (2004).
19. M. Rosvall, D. Axelsson, and C. T. Bergstrom, "The map equation," *Eur. Phys. J. Special Topics*, **178**, No. 1, 13–23 (2009).
20. M. Rosvall and C. T. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks," *Proc. Natl. Acad. Sci. USA*, **104**, No. 18, 7327–7331 (2007).
21. M. Rosvall and C. T. Bergstrom, "Maps of information flow reveal community structure in complex networks," *Proc. Natl. Acad. Sci. USA*, **105**, No. 4, 1118–1123 (2008).

M. I. Kolomeychenko, A. A. Chepovskiy, and A. M. Chepovskiy
National Research University "Higher School of Economics," Moscow, Russia
E-mail: achepovskiy@hse.ru