BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **15**, No 2

Sofia • 2015

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2015-0029

Improving Community Detection in Time-Evolving Networks Through Clustering Fusion

Ran Jin[1,2], *Chunhai Kou*[3], *Ruijuan Liu*[1]

[1] *School of Information Science and Technology, Donghua University, Shanghai, 201620 China*
[2] *School of Computer Science and Information Technology, Zhejiang Wanli University, Ningbo, 315100 China*
[3] *School of Science, Donghua University, Shanghai, 201620 China*
Emails: ran.jin@163.com kouchunhai@dhu.edu.cn ruirui@163.com

Abstract: *Traditional community detection algorithms are easily interfered by noises and outliers. Therefore, we propose to leverage a clustering fusion method to improve the results of community detection. Usually, there are two issues in clustering ensembles: how to generate efficient diversified cluster members, and how to ensembles the results of all members. Specifically:* (*1*) *considering the time evolving characteristic of real world networks, we propose to generate clustering members based on the snapshot of networks, where the split based clustering algorithms are performed;* (*2*) *considering the difference in the distribution of the cluster centers in each clustering member and the actual distribution, we ensemble the results based on a maximum likelihood method. Moreover, we conduct experiments to show that our method can discover high quality communities.*

Keywords: *Time-evolving network, community detection, clustering fusion, network snapshot, maximum likelihood method, Expectation Maximization algorithm.*

1. Introduction

Networks have been one of the most popular forms of many real world systems, such as personal networks in social relationships, collaboration networks between scientists, epidemic networks and Internet networks [1]. Generally speaking, the networks are composed of a set of nodes which are connected in some way to form systems, where the nodes represent physical entities, and the connections are the relationships between the physical entities [2]. For example, in a collaboration

network between scientists, the nodes are scientists and the connections between the nodes are the collaboration relationships between them. In graph theory, a network can be abstracted as a graph with a set of vertices and edges. Usually, the above networks contain a large amount of entities and relationships in the system, so they are used to describe the system phenomena and the internal relations. Due to the complexity of networks, they are also called complex networks [3, 4].

As one of the most significant focuses in complex networks research, community detection has attracted large attention. However, in real world scenarios, the networks are typically time evolving. Therefore, community detection methods must take into consideration the time-evolving feature of the networks. Besides, traditional clustering methods are easily disturbed by outliers and noises, and thus the quality of the detected communities is affected.

Therefore, in this paper, considering the dynamic characteristics of time evolving networks, we propose to employ a clustering fusion algorithm for community detection. Clustering fusion is the method which integrates the results of multiple clusters to induce the final results [5]. Basically, clustering fusion has two major steps: first it generates a set of clustering members and then combines the results of the above clustering members. Accordingly, there are two challenges in this work: (1) how to efficiently generate clustering members of time evolving networks; and (2) how to combine the clustering results together, considering the issues, such as mapping between different types of labels.

In summary, the contributions of this paper are as follows:

1) we provide a model of time evolving networks and a unified description of each clustering;

2) we design a clustering fusion algorithm based on the idea of the maximum likelihood method;

3) we perform extensive experiments to evaluate the proposed algorithm, which is proved to be efficient for community detection in evolving networks.

2. Related work

Many efforts have been made in community detection. Typical methods for community detection are based on betweenness [6], modularity [7], random walk [8], information theory [9], etc. For example, Girvan and Newman [10] proposed a divisive clustering method for community detection. Cohn and Chang [11] used PHITS (Probabilistical Hyperlink Induced Topic Search) by extending HITS (Hyperlink Induced Topic Search) [12] through PLSA (Probabilistic Latent Semantic Analysis) [19] for community detection. Nallapati et al. [13] designed MMSB (Mixed Membership Stochastic Block) to analyze the links for community detection. Moreover, Tang et al. [14] extend the community detection from a single network to multiple networks. Along this line, Papalexakis, Akoglu, and Ience [15] proposed a multi-graph clustering technique based on minimum description length and tensor analysis. Deng et al. [16] analyzed different relationships in a social network and proposed a linear combination method to optimize the member relationships in learning communities.

Tang, Wang and Liu [17] proposed to deal with the noises in multi-media networks through combination of multi-source data.

There are also existing works on evolving networks. For example, Breiger [18] analyzed the connections changing over time. Leskovec, Kleinberg and Faloutsos [19] investigated the evolving characteristics of networks. Tantipathananandh, Berger-Wolf and Kempe [20] formulated the dynamic community detection as a graph coloring problem, and proposed a heuristic method as a solution. Falkowski, Bartelheimer and Spiliopoulou [21] analyzed the partitions of different timestamps based on a statistical method. Differing from the existing efforts, in this paper we propose a clustering fusion method to improve community detection.

3. Overview of the proposed method

In this paper we focus on community detection in time evolving networks.

In order to describe the time evolving characteristic of networks, we use snapshots to represent the network at specific time. Therefore, we simply select the clustering members from the clustering results of the network snapshots during a certain period.

The proposed method is composed of two stages: first, for each network snapshot, perform a base clustering algorithm to get the clustering result at a specific timestamp, which will be discussed in Section 4. Second, combine the clusterings during a time period together to generate the final clustering, as presented in Section 5. In this way we consider the dynamic evolving characteristic of the networks by the fusion of clusterings among different timestamps.

4. Base clustering

In order to perform clustering on a network dataset into clusters (i.e., communities) with varied densities and shapes considering isolated and noise data, we propose a divisive hierarchical clustering algorithm.

Usually the communities in complex networks are connected. Inspired by that, we formulate the community detection problem as a clustering task on a connected graph. Therefore, the first step of base clustering is to construct a set of connected graphs from the network. Then, a divisive hierarchical clustering algorithm is performed on each connected graph.

4.1. Constructing connected graphs

Let the threshold of nodes being within the same community be l. The selection of l is automatically computed based on the largest connected graph. That is, $l = \frac{1}{2} e^{\frac{x}{y}}$, where x, y are the numbers of edges and nodes in the largest connected graph respectively.

Only when the strength of connection is not smaller than l, the two nodes are in the same community. Indeed, in this way the problem is simplified beforehand by eliminating weak or no connections.

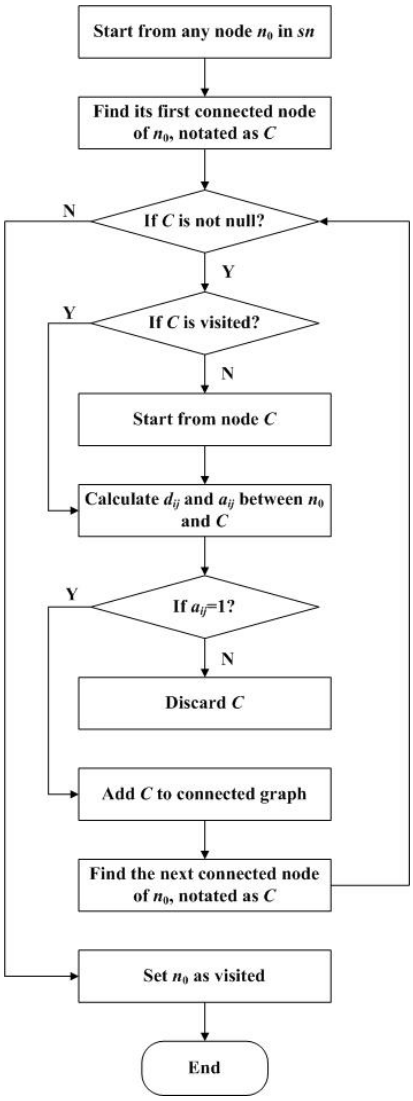

Fig. 1. Flow chart of constructing connected graphs

On the other hand, given the presence of noise data, typically the communities discovered are over connected. For example, even if entities a, b do not belong to the same communities in reality, the noise data might connect them together. Accordingly, different classes of objects are connected into one community through noise. Therefore, not only the connecting threshold, but the disconnecting threshold should also be set to ensure the efficiency of the connected graph. Denote the disconnecting threshold as h, which means that if the connection strength is larger than h, the connection is regarded as noise and must be discarded.

In network sn_t, the adjacency matrix between the nodes is notated as A, where each element a_{ij} is defined as

$$a_{ij} = \begin{cases} 1, & l \le d_{ij} \le h; \\ 0, & d_{ij} < l \text{ or } d_{ij} > h, \end{cases} \tag{1}$$

where d_{ij} is the distance between nodes *i* and *j*.

The procedure of constructing connected graphs is shown in Fig. 1. Basically, we employ a DFS (Depth-First-Search) strategy for generating eligible connected graphs. First of all, starting from any node n_0 in the network sn_t, and then traverse other nodes using DFS strategy. For each node C found, calculate the distance between n_0 and C as follows:

$$d(n_0, C) = \sum_{j \in (n_0, C)} (d(n_0, j) + d(j, C)), \tag{2}$$

where j is the intermediate node on the path (n_0, C).

Then compute $a(n_0, C)$ based on (1). If $a(n_0, C) = 1$, add the node to the connected graph; otherwise, delete node *C* and continue. The procedure finishes until all nodes in sn_t are visited.

4.2. Divisive hierarchical clustering

Now we have a set of connected graphs for each snapshot sn_t, so we need to further divide them into multiple partitions (i.e., communities).

The basic idea of divisive hierarchical clustering is to divide the network into two partitions based on a threshold and then update the threshold and perform the division recursively. First, we build a tree structured partition by divisive clustering. The steps of the algorithm are four and are described bellow.

Step 1. Initialize threshold *T* for division. The process is: randomly select N pairs of data points and calculate the distance d_{ij} between them; set T as $p \times (\mathrm{Ed}_{ij} + 0.25 \times \mathrm{Dd}_{ij})$, where $\mathrm{Ed}_{ij}, \mathrm{Dd}_{ij}$ are the expectation and deviation of distances, and p is a predefined percent. In this paper we set $p = 20\%$.

Step 2. For each connected graph in sn_t, find the initial clusters.

In this step we leverage the Grey Relational Analysis (GRA) [22] to measure the connections between nodes. Considering that the connections in real world situations are typically gray or fuzzy, GRA uses a gray correlation coefficient to measure the strength of connections. In our case, at time *t*, the gray correlation coefficient between nodes i, j is calculated as

$$g_{ij}(t) = \frac{\min\limits_j \min\limits_t \Delta_{ij}(t) + \alpha \max\limits_j \max\limits_t \Delta_{ij}(t)}{\Delta_{ij}(t) + \alpha \max\limits_j \max\limits_t \Delta_{ij}(t)}, \tag{3}$$

where $\alpha \in (0,1]$ is the resolution factor. The smaller α is, the better the resolution is. Typically, we set $\alpha = 0.5$ in this study. $\Delta_{ij}(t)$ is the difference between i, j at time t, which is computed as

$$\Delta_{ij}(t) = | w_i(t) - w_j(t) |, \tag{4}$$

where $w_i(t)$ is the importance of node i at time t:

$$w_i(t) = \sum_{j \in \text{neighbor}(i)} w_{ij}(t), \tag{5}$$

where $w_{ij}(t)$ is the value between node i and its neighbour.

Therefore, for any entity X we set its initial cluster as set C_0, where each element $Y \in C_0$ satisfies $g_{XY}(t) \geq 1 - T$. That is:

$$C_0(X) = \{Y \mid g_{XY}(t) \geq 1 - T\}. \tag{6}$$

If $C_0(X)$ is not null, start from X and get the initial cluster C_0. Now we divide the connected graph into two partitions, i.e., the initial cluster C_0 and the remaining nodes C_R.

Step 3. Update T as $T = T - \Delta$.

Step 4. If C_R is not null, divide C_R into two clusters C_L, C_R as in Step 2. Otherwise, the algorithm ends.

Now we have a list of tree structured partitions $C = \{C_1, C_2, ..., C_k\}$ and the next step is to determine the best partition results.

We measure the quality of the clusters by intra-cluster quality and inter-cluster separation, and the objective is to maximize the intra-cluster connection and minimize the inter-cluster connection, as illustrated in Fig. 2.

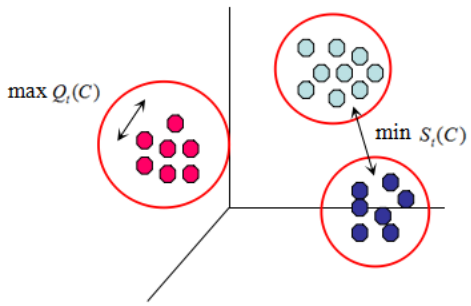

Fig. 2. Illustration of evaluating clusters

Let $Q_t(C)$ and $S_t(C)$ denote the intra-cluster distance and inter-cluster distance respectively, and:

$$Q_t(C) = \frac{1}{k} \sum_{i=1}^{k} \left(\frac{1}{|C_i|^2} \sum_{X,Y \in C_i} g_{XY}(t) \right), \tag{7}$$

(8) $$S_t(C) = \frac{1}{k(k-1)} \sum_{i=1, j \neq i}^{k} \left(\frac{1}{|C_i \| C_j|} \sum_{X \in C_i, Y \in C_j} g_{XY}(t) \right).$$

Therefore, in order to find the best partitions of the network snapshot, given a set of hierarchical clusters, for each layer of clusters, calculate:

(9) $$Q_t(C) - S_t(C),$$

and get the maximum value as clusters of sn_t.

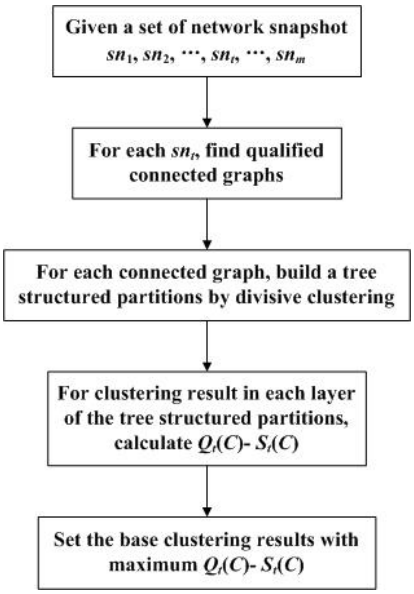

Fig. 3. The whole process of base clustering

The whole process of base clustering can be summarized in Fig. 3.

5. Clustering fusion

Now we have the base clustering results for each snapshot network. In this paper we propose to maximize the likelihood function of cluster labels for all snapshot networks to assign the final labels.

Denote the base clustering result for sn_t as $C_1^{\mathrm{sn}_t}, C_2^{\mathrm{sn}_t}, ..., C_{k_t}^{\mathrm{sn}_t}$, where k_t is the number of clusters for network sn_t. Therefore, for each entity x_i, $i = 1, 2, ..., N$, in sn_t, $t = 1, 2, ..., M$ we have a set of labels for different timestamps:

(10) $$x_i \rightarrow \{c_{x_i}^{\mathrm{sn}_1}, c_{x_i}^{\mathrm{sn}_2}, ..., c_{x_i}^{\mathrm{sn}_M}\}$$

where $c_{x_i}^{\mathrm{sn}_t}$ denotes the cluster label of x_i in the network sn_t.

Hence, the entities in the original dataset can be represented as

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & c_{x_1}^{\mathrm{sn}_1} & c_{x_1}^{\mathrm{sn}_2} & \cdots & c_{x_1}^{\mathrm{sn}_2} \\ x_{21} & x_{22} & \cdots & x_{2d} & c_{x_2}^{\mathrm{sn}_1} & c_{x_2}^{\mathrm{sn}_1} & \cdots & c_{x_2}^{\mathrm{sn}_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{N1} & x_{N2} & \cdots & x_{Nd} & c_{x_N}^{\mathrm{sn}_1} & c_{x_N}^{\mathrm{sn}_1} & \cdots & c_{x_N}^{\mathrm{sn}_1} \end{pmatrix}. \tag{11}$$

Consider the final cluster label of each entity y_i as a mixture model for the probability of $c_{x_i}^{\mathrm{sn}_t}$. That is, y_i is modeled upon the mixture of multi-variant components

$$P(\mathbf{y}_i \mid \Theta) = \sum_{t=1}^{M} \omega_t p_t(y_i \mid \theta_t), \tag{12}$$

where $\Theta = \{\theta_1, \theta_2, ..., \theta_M\}$, each component $p_t(y_i \mid \theta_t)$ is the distribution of each base clustering results for network snapshots, ω_t denoting the importance of *t*-th component. We assume that the more recent the snapshot is, the more significant the clustering is. Thus,

$$\omega_t = \frac{1}{|t - t_0|}, \tag{13}$$

where t_0 is the initial time.

Assume that the distribution of $Y = \{y_1, y_2, ..., y_N\}$ is independent and identical. In order to infer the value of y_i, we need to resolve the parameters Θ. Represent the log-likelihood function for Θ as follows [23]:

$$\log L(\Theta \mid Y) = \log \prod_{i=1}^{N} P(y_i \mid \Theta) = \sum_{i=1}^{N} \log \sum_{t=1}^{M} \omega_t p_t(y_i \mid \theta_t). \tag{14}$$

The objective is to find the estimation of Θ to maximize $\log L(\Theta \mid Y)$:

$$\hat{\Theta} = \arg\max \log L(\Theta \mid Y). \tag{15}$$

We employ the Expectation Maximization (EM) algorithm [24] to solve the estimate Θ. Let $f(\theta \mid y)$ be the posterior distribution density function, $f(\theta \mid y, c)$ be the posterior distribution added by the cluster labels of each individual base clustering. The procedure is described as follows.

E-step. Calculate the expectation of $\log f(\theta \mid y, c)$:

$$Q(\Theta, \theta_t) = E(\log f(\Theta \mid y, c) \mid y, \theta_t) = \frac{1}{M} \sum_{t=1}^{M} \log f(\Theta \mid y, c) f(c \mid y, \theta_t), \tag{16}$$

where $f(c \mid y, \theta_t)$ is the conditional distribution density function of *c*, given observation *y*.

M-step. Maximize $Q(\Theta, \theta_t)$ so that θ_{t+1} satisfies

$$\theta_{t+1} = \arg\max Q(\Theta, \theta_t). \tag{17}$$

Recursively perform EM steps until $|\theta_{t+1} - \theta_t| < \varepsilon$ or $|Q(\Theta, \theta_{t+1}) - Q(\Theta, \theta_t)| < \varepsilon$, where ε is a sufficiently small number.

6. Experiment

In this section we conduct experiments to evaluate the performance of the proposed clustering based community detection method. The dataset used in this experiment are achieved from [25], whose statistics are shown in Table 1.

We compare our proposed clustering fusion method with three baselines: (1) single *k*-means clustering, notated as KMC; (2) clustering fusion with k-means clustering on multiple snapshot networks, notated as F-KMC; (3) single divisive clustering as described in Section 4, notated as DC. Our proposed clustering fusion with divisive clustering is denoted as F-DC.

Tables 2 and 3 list the results of clustering for the four methods, including the convergence time, as well as the minimum, maximum, average and standard deviation values of log likelihood function. We can observe that though our proposed method F-DC is relatively time consuming in convergence, the performance of log likelihood function is better than the other three baselines. Besides, we can also see that the clustering fusion methods are better than single clustering on a specific snapshot network. Moreover, as shown in Fig. 4, the log likelihood function of F-DC convergence is approximately at iteration 15, which is satisfactory though.

Table 1. Statistics of the dataset in our experiment

Snapshot No	No of social nodes	No of social links	No of attri nodes	No of attri links	Crawled time
Snapshot 1	4,693,129	47,130,325	991,545	3,644,103	July 2011
Snapshot 2	17,091,929	271,915,755	3,108,141	14,693,125	August 2011
Snapshot 3	26,244,659	410,445,770	4,147,389	19,344,382	September 2011
Snapshot 4	28,942,911	462,994,069	4,443,631	20,592,962	October 2011

Table 2. Clustering results comparison of each snapshot without fusion

Snapshot No	Algorithm	Convergence time	Min	Max	Average	Standard deviation
Snapshot 1	KMC	6.1527	3.6694	5.8812	4.7753	11.5207
	DC	9.2464	3.7913	4.0789	3.9351	3.6813
Snapshot 2	KMC	7.2134	3.6321	5.9810	4.8066	12.9898
	DC	10.0021	3.7755	4.6824	4.2290	4.0215
Snapshot 3	KMC	11.1683	3.6706	6.0943	4.8825	13.0357
	DC	13.2464	3.7767	5.9633	4.8700	3.8848
Snapshot 4	KMC	12.2012	3.6543	5.7822	4.7183	13.1211
	DC	13.8994	3.7664	4.9925	4.3795	3.9697

Table 3. Clustering results in comparison with clustering fusion

Algorithm	Convergence time	Min	Max	Average	Standard deviation
F-KMC	4.2733	0.9102	0.9227	0.9165	1.9826
F-DC	15.2520	0.9025	0.9027	0.9026	0.0002

Moreover, we measure the performance of clustering using Dunn Index (DI) [26]:

$$\mathrm{DI} = \min_{1\le i\le n}\left\{\min_{1\le j\le n, i\ne j}\left\{\frac{\mathrm{dist}(i,j)}{\max_{1\le k\le n}\mathrm{dist}(k)}\right\}\right\}, \tag{18}$$

where $\mathrm{dist}(i, j)$ denotes the inter-cluster distance between clusters i and j, $\mathrm{dist}(k)$ is the intra-cluster distance within cluster k. The higher DI is, the better the clustering is.

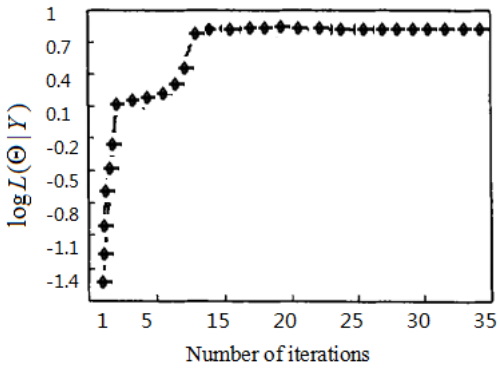

Fig. 4. Convergence curve of a log likelihood function of F-DC

We compare the DI values for the four methods in Fig. 5, where the instances are derived from the above four snapshots. We can observe that DI measure of the method herein proposed is the highest and it remains stable for different numbers of clusters. Therefore, we can conclude that our proposed clustering fusion method can detect high quality and accurate communities.

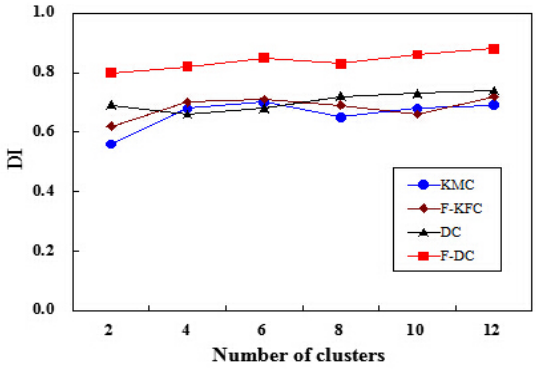

Fig. 5. Comparison of DI for different methods

7. Conclusion

In this paper we propose a clustering fusion method for detecting high quality communities. Specifically, first we generate the clustering members from different snapshots of a network and then employ a fusion method based on EM algorithm to maximize the likelihood of assigning a specific entity into some clusters. Though we consider different snapshots of evolving networks, we only use link based data for detecting communities. In future works we would like to explore the influence of emerging events on community detection.

***Acknowledgments*:** This work was supported by the Science and Technology Research Program of Zhejiang Province, under Grant No 2011C21036, and by the Shanghai Natural Science Foundation under Grant No10ZR1400100, as well as by the Science and Technology Innovation Team of Ningbo under Grant No 2012B82003, and by the National Undergraduate Training Programs for Innovation and Entrepreneurship under Grant No 201410876011.

References

1. Watts, D. J., S. H. Strogatz. Collective Dynamics of "Small-World" Networks. – Nature, Vol. **393**, 1998, No 6684, pp. 440-442.
2. Ahlswede, R., N. Cai, S-Y. R. Li, R. W. Yeung. Network Information Flow. – IEEE Transactions on Information Theory, Vol. **46**, 2000, No 4, pp. 1204-1216.
3. Strogatz, S. H. Exploring Complex Networks. – Nature, Vol. **410**, 2001, No 6825, pp. 268-276.
4. Boccaletti, S., V. Latora, Y. Moreno, M. Chavez, D.-U. Hwang. Complex Networks: Structure and Dynamics. – Physics Reports, Vol. **424**, 2006, No 4, pp. 175-308.
5. Frossyniotis, D., M. Pertselakis, A. Stafylopatis. A Multi-Clustering Fusion Algorithm. – In: Proc. of 2nd Hellenic Conference on Artificial Intelligence, Telos, Thessaloniki, Greece, Springer-Verlag, 2002, pp. 225-236.
6. Newman, M. E. J., M. Girvan. Finding and Evaluating Community Structure in Networks. – Physical Review E, Vol. **69**, 2004, No 2, pp. 56-68.
7. Newman, M. E. J. Finding Community Structure in Networks Using the Eigenvectors of Matrices. – Physical Review E, Vol. **74**, 2006, No 3, pp. 1-22.
8. Pons, P., M. Latapy. Computing Communities in Large Networks Using Random Walks. – In: Proc. of 20th International Symposium on Computer and Information Sciences, Turkey, Springer, 2005, pp. 284-293.
9. Santo, F., L. Vito, M. Massimo. A Method to Find Community Structures Based on Information Centrality. – Physical Review E, Vol. **70**, 2004, No 5, pp. 1-13.
10. Girvan, M., M. E. J. Newman. Community Structure in Social and Biological Networks. – Proceedings of National Academy of Sciences, Vol. **99**, 2002, No 12, pp. 7821-7826.
11. Cohn, D., H. Chang. Learning to Probabilistically Identify Authoritative Documents. – In: Proc. of 17th International Conference on Machine Learning, Omni Press, Haifa, Israel, 2000, pp. 167-174.
12. Kleinberg, J. M. Authoritative Sources in A Hyperlinked Environment. – Journal of the ACM, Vol. **46**, 1999, No 5, pp. 604-632.
13. Nallapati, R., A. Amr, X. Eric, W. C. William. Joint Latent Topic Models for Text and Citations. – In: Proc. of 14th ACM SIGKDD International Conference on Knowledge, Discovery and Data Mining, ACM, Las Vegas, USA, 2008, pp. 1-9.
14. Lei, T., X. Wang, H. Liu. Community Detection via Heterogeneous Interaction Analysis. – Data Mining and Knowledge, Discovery, Vol. **25**, 2012, No 1, pp. 1-33.
15. Papalexakis, E. E., L. Akoglu, D. Ience. Do More Views of A Graph Help? Community Detection and Clustering in Multi-Graphs. – In: Proc. of 16th International Conference on Information Fusion, IEEE, Istanbul, Turkey, 2013, pp. 899-905.

16. C a i, D., Z. S h a o, X. H e, X. Y a n, J. H a n. Community Mining from Multi-Relational Networks. – In: Proc. of 16th European Conference on Machine Learning, Oporto, Portugal, Springer, 2005, pp. 445-452.
17. T a n g, J., X. W a n g, H. L i u. Integrating Social Media Data for Community Detection. – In: Proc. of Modeling and Mining Ubiquitous Social Media, Springer, Boston, USA, 2012, pp. 1-20.
18. B r e i g e r, R. L. Social Structure from Multiple Networks. – The American Journal of Sociology, Vol. **81**, 1976, No 4, pp. 730-780.
19. L e s k o v e c, J., J. K l e i n b e r g, C. F a l o u t s o s. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. – In: Proc. of 11th ACM SIGKDD International Conference on Knowledge, Discovery and Data Mining, ACM, New York, USA, 2005, pp. 177-187.
20. T a n t i p a t h a n a n a n d h, C., T. B e r g e r-W o l f, D. K e m p e. A Framework for Community Identification in Dynamic Social Networks. – In: Proc. of 13th ACM SIGKDD International Conference on Knowledge, Discovery and Data Mining, New York, USA, ACM, 2007, pp. 717-726.
21. F a l k o w s k i, T., J. B a r t e l h e i m e r, M. S p i l i o p o u l o u. Mining and Visualizing the Evolution of Subgroups in Social Networks. – In: Proc. of 2006 IEEE/WIC/ACM International Conference on Web Intelligence, IEEE, Hong Kong, 2006, pp. 52-58.
22. C h a n, J. W. K, T. K. L. T o n g. Multi-Criteria Material Selections and End-of-Life Product Strategy: Grey Relational Analysis Approach. – Materials & Design, Vol. **28**, 2007, No 5, pp. 1539-1546.
23. S h i m o d a i r a, H. Improving Predictive Inference under Covariate Shift by Weighting the Log-Likelihood Function. – Journal of Statistical Planning and Inference, Vol. **90**, 2000, No 2, pp. 227-244.
24. D e m p s t e r, A. P., N. M. L a i r d, D. B. R u b i n. Maximum Likelihood from Incomplete Data via the EM Algorithm. – Journal of the Royal Statistical Society, Series B (Methodological), Vol. **39**, 1977, No 1, pp. 1-38.
25. G o n g, N. Z., W. X u, L. H u a n g, P. M i t t a l, E. S t e f a n o v, V. S e k a r. Evolution of Social-Attribute Networks: Measurements, Modeling, and Implications Using Google+. – In: Proc. of ACM Conference on Internet Measurement Conference, ACM, New York, USA, 2012, pp. 131-144.
26. D u n n, J. C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. – Journal of Cybernetics, Vol. **3**, 1973, No 3, pp. 32-57.