Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005

INCREMENTAL ALGORITHM FOR DETECTING COMMUNITY STRUCTURE IN DYNAMIC NETWORKS

BO YANG, DA-YOU LIU

College of Computer Science and Technology, Jilin University, ChangChun 130012, China
Open Symbol Computation and Knowledge Engineering Laboratory of State Education Department, Jilin University, Changchun 130012, China
E-MAIL: yangbojlu@sina.com, boyang@mail.jlu.edu.cn

Abstract:

Community structure is an important property of networks. Being able to identify communities can provide invaluable help in exploiting and understanding both social and non-social networks. Several algorithms have been developed up till now. However all these algorithms can work well only with small or moderate networks with vertexes of order 10^4. Additionally, all existing algorithms are off-line and can not work well with highly dynamic networks such as web, in which web pages will be updated frequently every day. When the already clustered network is updated, the entire network including original and incremental parts has to be recalculated, even only slight changes involved. To address this problem we develop an incremental algorithm which allows for detecting community structure in large-scale and dynamic networks. Based on the community structure it detect previously our algorithm will take little time to reclassify the entire network including both original and incremental parts. Furthermore, our algorithm is faster than most existing algorithms such as Girvan and Newman's algorithm and its improved versions. Also our algorithm can help to visualize these community structures in network and provide a new approach to research the evolving process of dynamic networks.

Keywords:

Social network; Community; Incremental algorithm

1. Introduction

Many systems in our world take the form of networks, sets of vertices joined together in pairs by edges [1]. Examples include social networks [2, 3] such as acquaintance networks [3] and scientific collaboration networks [4], technological networks such as the Internet [5], World Wide Web [6], and power grids [3], and biological and ecosystems networks such as neural networks [3], food webs [7], epidemiology networks[8, 9] and metabolic networks [10].

Recent research on networks among mathematicians, physicists and computer scientists has focused on a number of distinctive statistical properties that most networks seem to share, for example small world effect[3], power-law distribution of degree[1,11] and network transitivity[3,12].

In this paper, we consider another important and common property of generic networks, community structure, which is the topological property of networks, i.e. the division of network nodes into some groups within which the network connections are dense, but between which they are sparser. Communities in a social network might represent real social groupings, perhaps by interest or background; communities in a citation network might represent related papers on a single topic; communities in a metabolic network might represent cycles and other functional groupings; communities in Web might represent pages on related topics. In the last case, the ability to identify all communities of world wide web is highly useful for the implementation of next generation search engines, content filtering, automatic classification, and the automatic realization of ontologies[13]. So being able to identify these communities can provide invaluable help in understanding and visualizing the structure of networks, and also can help us to exploit these networks more effectively. However, this task is highly nontrivial.

All of existing algorithms to perform this task can be divided into two groups, *bisection* and *hierarchical*.

With bisection algorithms, one finds the best division he can of the complete graph into two groups, and then further subdivides those two until he has the required number of groups. Two most dominated bisection algorithms are the spectral bisection method[14,15], which is based on the eigenvectors of the *Laplacian* matrix of network, and the Kernighan-Lin algorithm[16], which is aimed to optimize one partition function predefined using greedy algorithm. A quite different approach proposed recently by Wu and Huberman[17] is also a bisection algorithm fundamentally.

The idea behind hierarchical clustering algorithm is to

0-7803-9091-1/05/$20.00 ©2005 IEEE

create a hierarchical structure of all communities in a given network, so called *dendrogram*. According to the order of building dendrograms, the hierarchical clustering algorithm can be divided into two main approaches, *agglomerative*, a bottom-up method, and *divisive*, a top-down method.

Agglomerative clustering algorithms are developed by sociologists to analyze many kinds of social networks. [18] The main idea of the method is to develop a measure of similarity for every pair of nodes in the network. Recently Newman present a faster communities detection algorithm with time complexity O(*mn*) using greed strategy to optimize the evaluation function $Q = \sum_i (e_{ii} - a_i^2)$, which is also an agglomerative clustering method essentially[19].

In the divisive hierarchical clustering algorithms, the order of constructing dendrogram is reversed. The crucial point of divisive algorithm is the selection of the edges to be cut, which have to be those connecting communities and not those within them. Girvan and Newman present a famous GN algorithm to solve this problem based on the value of *edge betweenness*[20], a generalization of the v*ertex betweenness* introduced by Freeman[21]. The main disadvantage of the GN algorithm is its speed. Since there are *m* edges to be removed in total and each iteration of the algorithm takes O(*mn*) time, the worst-case running time of the algorithm is O(m^2n), or O(n^3) on a sparse network. To address this issue several approximate version of basic GN algorithm have been presented, such as the algorithms of Tyler[22] and of Radicchi [23].

All of the algorithms discussed above are centralized and only can work well when applied to small or moderate networks, i.e. all vertexes of networks are not more than 10^4. Roughly estimated, it will take these algorithms several years to finish detecting community structure in networks around 10^5 vertexes. While up till now, the total number of existing web pages has increased to more than 8 billions.

Additionally, all of existing algorithms are off-line, namely, they can not work well with highly dynamic networks. When the already clustered network is changed, for example, adding some new vertexes or edges to the network, deleting some vertexes or edges from the network, or changing some weights of edges, the entire network, including original and incremental parts, has to be input into these algorithms again and be calculated again, even only slight changes involved, in the extreme case, only one vertex and one edge are added. Obviously the strategy of recalculating entire network after each modification in disregard of its original community structure is infeasible for large-scale and highly dynamic networks such as web, in which web pages will be updated frequently every day.

To address this big problem we develop an on-line or incremental algorithm for identifying community structure in generic networks.

2. Algorithm

2.1. Basic idea and concepts

This method is inspired by *n*-body simulation, which imitates a real physical world such as celestial, chemical, or atomic system by computers.

Entire network with *n* vertices and *m* edges is considered as a 4-dimension (4-D) mechanics world of $E^3 \times T$, in which E^3 is the three dimensional Euclidean space with fixed orientation and *T* is the time dimension.

Each vertex v_i in network is seemed as a particle which has its own physical attributes such as position x_i , velocity, acceleration and mass m_i, and will diffuse in such 4-D world and interact with other particles through forces. Generally, m_i is equal with the degree of v_i, which means vertex with more centrality has more impact on others. Also m_i might represent the weight of vertex in some special networks.

Each edge (v_i,v_j) of networks is seemed a virtual spring, called *edge spring*, which can be extended or compressed but can not exert really force to each vertex that it connects. When the extended or compressed edge spring is released it will return to its original length, called *natural length* l^0 , while provided its current length l_{ij}^t is not too great, i.e. not over certain threshold, otherwise the edge spring will be broken. Threshold δ_{ij}^t is proportional to its weight w_{ij} and can be calculated by equation (1) and (2), where μ and α are two constant.

$$\delta_{ij}^t = \mu \cdot w_{ij}^{\alpha} \cdot \max(l^0, E(l^t)) \tag{1}$$

$$E(l^t) = \frac{1}{m} \cdot \sum_{(v_p,v_q) \in E} l_{pq}^t \tag{2}$$

However, the broken edge spring could be reconnected when its current length is below threshold again resulting from the decreasing displacement of related vertices.

Such 4-D mechanics world of networks complies with following its own laws.

Law I. Such world complies with Newton-Laplace principle of determinacy: the state of mechanical system at any fixed moment of time uniquely determines all of its motion.

Mathematically, Law I can be formulated as equation 3

$$U_t(v_0) = \Psi(\sum_i F(x_0, x_i)) \tag{3}$$

where $U_t(v_0)$ is a physical quantity of v_0 at time *t* which can

be obtained by summing the pairwise interactions $F(x_0, x_i)$ over the particles of system, and Ψ is a mapping function. For instance, in our 4-D mechanics world of networks, $F(x_0, x_i)$ is the forces between two vertices, $U_t(v_0)$ is the position of vertex v_0 at moment t, and Ψ is an integrator.

Law II. Every pair of vertices repulses each other but not attracts in contrast to Newtonian universal gravitational law. The repulsive forces to vertex v_i at position x_i can be calculate by equation 4:

$$F_r(x_i) = \sum_{v_j \in V} m_i m_j (l^0)^2 \frac{x_i - x_j}{\left|x_i - x_j\right|^2} \tag{4}$$

where l_0 is the natural length of edge spring, and can be calculated by equation (5).

$$l_0 = \beta \cdot \sqrt{\frac{volume\ of\ E^3}{n}} \tag{5}$$

where β is a constant. We would like the vertices to be uniformly distributed in the Euclidean space, and l^0 is the radius of the empty sphere around a vertex.

Law III. Only the pair of vertices connected by an edge spring in networks can attract each other like Newtonian universal gravitational law. The attractive forces to vertex v_i at position x_i can be calculate by equation 6:

$$F_a(x_i) = \sum_{(v_i, v_j) \in E} m_i m_j \frac{\left|x_i - x_j\right|^2}{l^0} \cdot \frac{x_i - x_j}{\left|x_i - x_j\right|} \tag{6}$$

Figure 1 illustrates the relations among repulsive force, attractive force and their sum versus distance between two vertices. The point where the sum of the attractive and repulsive force crosses the distance-axis is where the two forces would exactly cancel each other out, and this is at l^0, the natural length of edge spring.

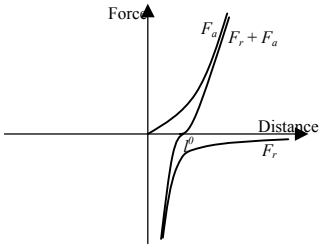

Figure 1. Forces versus distance between two vertices. (i) $|F_a| = |F_r|$ when $|x_i\text{-}x_j| = l_0$; (ii) $|F_a| > |F_r|$ when $|x_i\text{-}x_j| > l_0$; (iii) $|F_a| < |F_r|$ when $|x_i\text{-}x_j| < l_0$.

With such ideal 4-D mechanics world of networks modeled above, the idea behind our algorithm to detect community structure in network can be described as follows.

Initially, in the 4-D mechanics world of networks, each vertex is specified a random position, zero velocity and zero acceleration; each edge is set to its nature length and zero force on it.

Once the 4-D mechanics world of networks is activated it will evolve continuously and autonomously directed only by the three laws above. All vertices diffuse simultaneously in E^3 space driven by the forces exerted to them, and all edges extend or compress resulting from the displacement of the two neighbor vertices connected by it.

Because the edges within communities are denser, the vertices within same community would attract each other rather than repulse. The higher the density of edges of one community the shorter the distances among its vertices, and thus the entire community including all its vertices and edges can be seemed as a single pole. In contract, because the edges within communities are sparser, the vertices between two different communities trend to repulse each other rather than attract, and each community acts as a repulsive pole that push all the vertices in the other community far away and stretches solitary connections between them. Those edges would be broken when the forces exerting on them are too great and exceed the maximum value that they can endure. Two communities will appear after all edges between them are broken. After the 4-D mechanics world of networks converge its equilibrium state, in which the sum of displacement of all vertices is less than a predefined error ε, all communities will be formed automatically.

In order to imitate the evolution of such 4-D mechanics world of networks we need to calculate the diffusing tracks of all particles in the world.

According to Newton's second law, the position of each particle in 4-D mechanics world of networks can be calculated by equation 7

$$\frac{\partial^2}{\partial x^2} x_i(t) = \frac{F(x_i(t))}{m_i} \tag{7}$$

Equation 7 is known as *ordinary differential equation*, or ODE because it relates to the second derivative of $x_i(t)$, the position of vertex v_i at time t. Generally, solving ODEs that come up with exact formulas is hard, even for the problem only with two vertices. While fortunately, numerical method turns out to be efficient and accurate enough for our problem of detecting community structure in networks. From equation 7 we have

$$x_i(t) = \frac{1}{m_i} \iint F(x_i(t)) dt dt \tag{8}$$

Solving integral equation (8) is also difficult, since x_i is present on both sides. Also, x_i is dependent on t, which means we have a system of n coupled integral equations for each time step.

A discrete numerical approach to solve equation 8 is described by equation 9, which is a linear combination of the function F evaluated at different time points.

$$\Delta x_i = \frac{1}{m_i}\sum_{k=1}^{\infty}\gamma \cdot F(x_i(t+h_k)) \qquad (9)$$

Different discrete integration schemes yield different coefficients γ and h_k. A commonly used integrator is the so-called *Leapfrog* integration scheme.

2.2. Algorithm description

Based on the idea, concepts and equations discussed in previous section, the incremental algorithm for detecting community structure in networks is given in table 1.

Table 1. The incremental algorithm for detecting community structure in dynamic network

PROCEDURE IA-DCS
volume of $E^3 = 0.75\pi R^3$;
INPUT(N); {$N = (V, E), |V| = n, |E| = m$}
$l_0 = \beta \cdot \sqrt[3]{\frac{volume\ of\ E^3}{n}}$
$t = 0$
LOOP
Errors = 0
FOR $\forall v_i \in V$ DO
BEGIN
$F_r(x_i) = \sum_{v_j \in V} m_i m_j (l^0)^2 \frac{x_i - x_j}{|x_i - x_j|^2}$
$F_a(x_i) = \sum_{(v_i, v_j) \in E} m_i m_j \frac{|x_i - x_j|^2}{l^0} \cdot \frac{x_i - x_j}{|x_i - x_j|}$
$F(x_i) = F_r(x_i) + F_a(x_i)$
$\Delta x_i = \frac{1}{m_i} \cdot \gamma \cdot F(x_i)$
Errors = Errors + Δx_i
END
{diffusing all vertices}
FOR $\forall v_i \in V$ DO
$x_i = x_i + \Delta x_i$
{checking each edge spring}
FOR $\forall (v_i, v_j) \in E$ DO
IF $|x_i - x_j| \geq \delta_{ij}$ THEN
BEGIN
{broken edge spring}
$E = E - (v_i, v_j)$;
$\overline{E} = \overline{E} + (v_i, v_j)$;
END
END
FOR $\forall (v_i, v_j) \in \overline{E}$ DO
IF $|x_i - x_j| < \delta_{ij}$ THEN
BEGIN
{reconnected edge spring }
$\overline{E} = \overline{E} - (v_i, v_j)$;
$E = E + (v_i, v_j)$;
END
END
{continue iteration }
$t = t + 1$;
UNTIL($t \geq iteration$ || Errors $< \varepsilon$)
OUTPUT(N);
END PROCEDURE

2.3. Algorithm analysis

Calculating repulsive forces takes time O(n^2); Calculating attractive forces takes time O(m); diffusing all vertices takes time O(n); Checking each edge spring takes time O(m); totally, each iteration of our algorithm needs time O(n^2+m). So the entire time of the algorithm is O($iterations \times (n^2+m)$), or O($iterations \times n^2$), in both dense and sparse networks.

But exactly how many iteration steps at least are required to detect all communities of network in problem? This problem, i.e. the value of *iterations*, is closely related to the initial layout of networks.

The rationale of our algorithm is somewhat like the general law of nature that homogeneities attract each other while heterogeneities repulse each other. The vertices in the same community are seemed as homogeneous, otherwise heterogeneous. Because of the impact of repulsive or attractive forces, homogeneous vertices trend to approach closer and closer, and synchronously, heterogeneous vertices trend to depart from each other further and further. When the distance between two sub-graphs are far enough, they are taken as two heterogeneous communities.

Based on the rationale, the community structure of network is decided exclusively by its final topological structure in Euclidean space. So the time required to detecting community structure is just the time to find the final topological structure of network we need. To do so, in our algorithm, all vertices will diffuse simultaneously and repeatedly until the final topological structure is found.

Obviously, the final topological structure will be found quickly if the initial layout of network is "good", namely, all vertices already have been close to their correct

positions according to their affiliations of community. In other word, it will take much less time to find community structure starting from a "half-baked" network than from a scratch. It is the fact that contributes to the incremental feature of our algorithm.

So with our algorithm, a network which has been clustered previously will be reclassified quickly through few iteration steps after it is updated incrementally. The ability will enable our algorithm to work well with lager networks, especially dynamic networks such as WWW. Also our algorithm can be used to research the evolving process of such dynamic networks.

2.4. Speeding Algorithm

Fundamentally, our algorithm is a particle-particle algorithm, so it has to take time $O(n^2)$ to calculate the repulsive forces between each pair of vertices. However, it is easy to speed our basic algorithm using techniques of n-body simulation which permit not only particle-particle but also particle-cluster and cluster-cluster interactions. Barnes-Hut(BH), Particle-Mesh(PM) and Fast Multi-pole Method(FMM) algorithms are three most popular quick algorithms which can reduce the time of calculating repulsive forces of all vertices from $O(n^2)$ to $O(nlogn)$ or even $O(n)$. So the best time complexity of our improved algorithm would be $O(iterations \times n)$.

3. Experiments

3.1. Detecting communities in social network

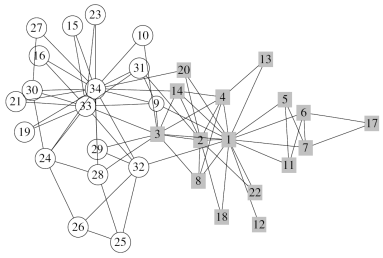

Figure 2. The friendships network of Zachary's karate club study [24].

Firstly, we test our algorithm against the friendship network data from Zachary's karate club study[24]. Over two years in 1970s, Wayne Zachary observed social interactions between the members of a karate club at an American university. Based on their social interactions within the club he constructed networks of ties between members of the club. During the course of his study, a dispute arose between the club's administrator and its principal karate teacher by chance, and as a result the club eventually split in two roughly equal size clubs, centered around the administrator and the teacher respectively, as shown in figure 2.

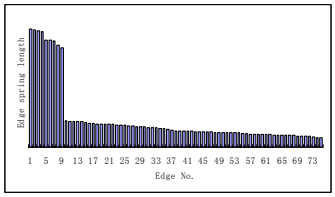

Figure 3. In the figure x-axis represents the edge no. and y-axis represents the edge spring length in the final topological structure of network detected by our algorithm. The 9 broken edges including (1,32), (2,31), (14,34), (20,34), (3,10), (3,33), (3,29), (3,28), (3,9) are just between the two communities, and much longer than other 66 edges within communities.

In this experiment, nine longest edges are broken and two communities are detected correctly by our algorithm after 127 iteration steps. The length of each edge spring at equilibrium state is shown in decreasing order in figure 3.

3.2. Detecting communities in dynamic network

In this experiment we test our algorithm using an artificial network presented by Newman[19]. The network consists of n = 128 vertices divided into 4 groups of 32. Each vertex has z_{in} edges connecting it to members of the same group and z_{out} edges to members of other groups, with the sum $z_{in} + z_{out}$ = 16. As z_{out} grows from 0 to 16, the community structure in the network becomes less and less well defined, i.e. more and more ambiguous. Algorithm is considered to be successful if the four communities are detected, each node is classified in the right community, and the four communities are not further subdivided. Experiment results are shown in figure 4 and figure 5.

Form figure 4 we can see that both algorithms work very well when $z_{out} < 6$, correctly identifying more than 90% of vertices, whereas they perform less and less well when z_{out} is greater than 6 and approaches to 16. Comparing with GN algorithm, our algorithm performs well at some case when $z_{out} < 7$, while slightly worse when $z_{out} > 7$.

From figure 5 we can see that: (1) The time of GN algorithm increases with z_{out} increasing from 0 to 16, but changing slightly, because the total number of edges and vertices is fixed; (2) the average time of our algorithm is less than that of GN algorithm; (3) the time of our

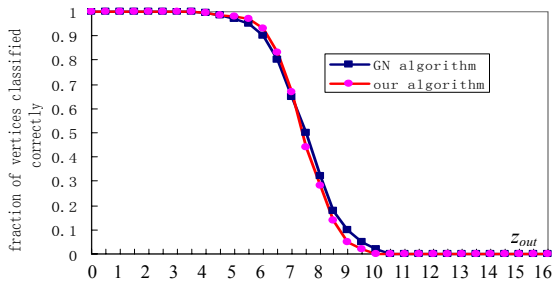

Figure 4. Comparison of the fraction of successes between our algorithm and GN algorithm in an dynamic random network with 128 vertices and 2048 edges. The degree of each vertex is 16 = z_{in}+ z_{out}.

algorithm at z_{out}= 0 is slightly more than that of GN algorithm because it need to detect community from scratch; (4) but the time of our algorithm decrease sharply at z_{out}= 0.5 and almost does not increase during z_{out} changing from 0.5 to 4 because the previously clustered network is a good initial input of next run and can be adjusted quickly; (4) during z_{out} changing from 5 to 12 the time of our algorithm increase smoothly because more and more edges of inter-community are added and the layout of previously clustered network becomes chaos rather than half-baked. Consequently, most of vertices have to take more time to diffuse in order to find their new correct position in current run.

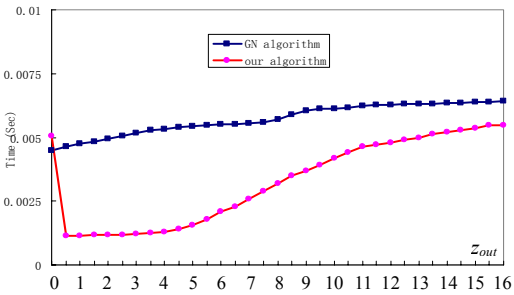

Figure 5 Comparison of the time needed to detect four communities by our algorithm and GN algorithm in an dynamic random network with 128 vertices and 2048 edges. The degree of each vertex is 16 = z_{in}+ z_{out}. The runs are performed on a desktop computer with a 3.0G CPU.

4. Conclusions

In this paper, we presented our algorithm which allows for detecting community structure in dynamic networks. Our algorithm is inspired by the force interaction of Newtonian mechanics, and is quit different from all existing ones for its incremental feature. Based on previously clustered networks our algorithm will take little time to reclassify the updated network including both original and incremental parts. With the exclusive ability our algorithm is very competent for dealing with large, dynamic networks such as WWW. Additionally, Our algorithm with time $O(iterations \times n^2)$ is faster than most existing algorithms such as GN algorithm with time $O(m^2n)$[20], their improved version with $O(mn)$[19], and Radicchi's algorithm with time $O(m^4/n^2)$[23]. Not only can our algorithm identify community structures of networks quickly but also it can visualize these structures and provide a new approach to research the evolving process of dynamic networks.

References

[1] S. H. Strogatz. Exploring complex networks. Nature, 2001(410), 268-276.
[2] J. Scott. Social Network Analysis: A Handbook. Sage Publications, London, 2nd edition, 2000.
[3] D. J. Watts, S. H. Strogatz. Collective dynamics of small-world networks. Nature, 1998(393), 440-442.
[4] M. E. J. Newman. The structure of scientific collaboration networks. Proc. Natl. Acad. Sci. USA 98, 404–409.2001.
[5] M. Faloutsos, P. Faloutsos, C. Faloutsos. On power-law relationships of the internet topology. Computer Communications Review 1999(29), 251-262.
[6] R. Albert, H. Jeong, A.L.Barabasi. Diameter of the world-wide web. Nature, 1999(401), 130-131.
[7] R. J. Williams and N. D. Martinez, Simple rules yield complex food webs. Nature, 2000(404), 180-183.
[8] R. P.Satorras, A. Vespignani. Epidemic spreading in scale-free networks. Phys. Rev. Letter, 2001(86), 3200–3203.
[9] R. M. May and A. L. Lloyd, Infection dynamics on scale-free networks. Phys. Rev. E, 2001(64), 066112.
[10] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barab´asi, The large-scale organization of metabolic networks. Nature, 2000(407), 651–654.
[11] A.L.Barabasi, R.Albert. Emergence of scaling in random networks. Science, 1999(286), 509-512.
[12] M.E.J.Newman, S.H.Strogatz, D.J.Watts. Random graphs with arbitrary degree distributions and their applications. Phys. Rev. E, 2001(64), 026118.
[13] G. W.Flake, S.R.Lawrence, C. L.Giles, F. M.Coetzee. Self-organization and identification of web communities. IEEE Computer, 2002(35), 66–71.
[14] M. Fiedler. Algebraic connectivity of graphs. Czech.Math. 1973(23), 298-305.

[15] A.Pothen, H.Simon, and K.P.Liou. Partitioning sparse matrices with eigenvectors of graphs. SIAM Jounal of Matrix Analysis and Application. 1990(11), 430-452.
[16] B. W. Kernighan, S. Lin. An effcient heuristic procedure for partitioning graphs. Bell System Technical 1970(49), 291-307.
[17] F. Wu, B. A. Huberman. Finding communities in linear time: A physics approach. Preprint condmat/0310600, 2003.
[18] J. Scott. Social Network Analysis: A Handbook. Sage,London, 2nd edition.2000.
[19] M. E. J. Newman. Fast algorithm for detecting community structure in networks. Preprint cond-mat/0309508, 2003.
[20] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. Proc. Natl. Acad. Sci.USA 99, 7821-7826.2002.
[21] L. C. Freeman. A set of measures of centrality based upon betweenness. Sociometry. 1977(40), 35-41.
[22] J.R.Tyler, D.M.Wilkinson, B.A.Huberman. Email as spectroscopy: Automated discovery of community structure within organizations. In M.Huysman, E.Wenger, V.Wulf (eds.), Proceedings of the First International Conference on Communities and Technologies, Kluwer, Dordrecht, 2003.
[23] F.Radicchi, C.Castellano, F.Cecconi, V.Loreto, D.Parisi, Defining and identifying communities in networks. Proceedings of the National Academy of Science(PNAS), 2004,101(9),2658-2663.
[24] W.W.Zachary. An information flow model for conflict and fission in small groups. Journal of Anthropological Research,1977(33), 452–473.