Graph Clustering Based on Mixing Time of Random Walks
Konstantin Avrachenkov, Mahmoud El Chamie, Giovanni Neglia

To cite this version:

Konstantin Avrachenkov, Mahmoud El Chamie, Giovanni Neglia. Graph Clustering Based on Mixing Time of Random Walks. IEEE International Conference on Communications (ICC 2014), Jun 2014, Sydney, Australia. pp.4089-4094, 10.1109/ICC.2014.6883961. hal-01087693

HAL Id: hal-01087693
https://inria.hal.science/hal-01087693
Submitted on 26 Nov 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Graph Clustering Based on Mixing Time of Random Walks

Konstantin Avrachenkov Mahmoud El Chamie Giovanni Neglia
INRIA Sophia Antipolis - Méditerranée
2004 Route des Lucioles, B.P. 93
06902 Sophia Antipolis, France
{konstantin.avratchenkov, mahmoud.el, giovanni.neglia}@inria.fr

Abstract—Clustering of a graph is the task of grouping its nodes in such a way that the nodes within the same cluster are well connected, but they are less connected to nodes in different clusters. In this paper we propose a clustering metric based on the random walks’ properties to evaluate the quality of a graph clustering. We also propose a randomized algorithm that identifies a locally optimal clustering of the graph according to the metric defined. The algorithm is intrinsically distributed and asynchronous. If the graph represents an actual network where nodes have computing capabilities, each node can determine its own cluster relying only on local communications. We show that the size of clusters can be adapted to the available processing capabilities to reduce the algorithm’s complexity.

I. INTRODUCTION

A community of nodes (or a cluster of nodes) in a network is a group of vertices that are well connected to each other, but are less connected with the remaining part of the network. Detecting clusters in networks has many applications. Communities in social networks are formed by people having common interest. Clusters in the web graph can group pages with similar topics. E-commerce, classification, computer vision, bioinformatics, and machine learning are only few areas of application of network clustering.

Comparing different possible graph clustering outputs and selecting the best outcome are carried out by introducing a quality metric that serves as an objective function. There is still no consensus in the literature on which quality metric for graph clustering is the best one. One of the most used metrics is the modularity [1] which gives a score to the cluster containing a node depending on the difference between the number of edges of a random graph having similar degree distribution and the slower it escapes from the cluster in the unconstrained case. Both effects can be quantified considering the eigenvalues of appropriate matrices. Beside introducing this new metric, we propose a randomized algorithm for clustering the network accordingly. The algorithm is local because it relies only on a partial view of the entire network. In particular, if the graph represents the topology of a network where nodes have computing capabilities, the algorithm can run in parallel at each node without the need of a central unit. Being local, clusters can be formed in parallel and the computation complexity is distributed among clusters. The algorithm can also find small clusters that are more difficult to be detected by the global clustering methods. We present in the next section the notation used across the paper.

II. NOTATION

Let $G = (V,E)$ be an undirected unweighted connected graph without self-loops, where $V = \{1, \ldots, n\}$ is the set of vertices and $E$ is the set of $m = |E|$ edges. Let $d_G(i) = |\{j \in V, (i,j) \in E\}|$ be the degree of a node $i$ in $G$, $D_G$ be a diagonal matrix having on its diagonal the degree of the nodes in $G$ and let $A_G$ be the adjacency matrix of the graph $G$ where $a_{ij} = 1$ if $(i,j) \in E$, and $a_{ij} = 0$ otherwise. For any set $S \subseteq V$, let $D_G(S)$ (resp. $A_G(S)$) be the sub-matrix of $D_G$ (resp. $A_G$) obtained considering only rows and columns corresponding to the vertices in $S$. Let $G(S) = (S,E(S))$ be the subgraph induced by $S \subseteq V$ where $E(S) = \{(i,j) \in E | i,j \in S\}$. Observe that in general $D_G(S) \neq D_G(S)$ because $D_G(S)$ contains the degree of nodes in the original graph $G$ which are different from their degrees in the induced subgraph.
When is not changed. If spectral gap $G$ in module of $G$. When $P$ is stochastic, let $s(P) = 1 - \lambda_2(P)$ be $1$ its spectral gap where $\lambda_2(P)$ is the second largest eigenvalue in module of $P$. Finally, $I$ is the identity matrix.

A clustering $C_G$ of a graph $G$ is a partition of the vertices such that $C_G = \{C_1, \ldots, C_k\}$ where $C_1 \cup \ldots \cup C_k = V$ and $C_u \cap C_v = \phi$ for all clusters $C_u$ and $C_v$ in $C_G$. Let $C(i) = \{C_u : i \in C_u\}$ be the cluster that contains node $i$. Let $f : V \rightarrow \mathbb{R}$ be a scoring function for the nodes, define a cluster score $f(C_u) = \sum_{i \in C_u} f(i)$, and a clustering algorithm score $f(C_G) = \sum_{i=1}^{k} f(C_u) = \sum_{i=1}^{n} f(i)$.

### III. The Random Walk Fitness Measure

In this section, we introduce a new scoring function $f(.)$ that can serve as a quality measure for a clustering algorithm. A good clustering algorithm identifies clusters that are well connected internally, but weakly connected with the rest of the network. Inspired by this intuitive definition, the function $f$ should have the following properties:

1) A cluster whose induced subgraph is disconnected should receive the minimum score.
2) A clique graph clustered as a single cluster should have the highest score among all clusterings for graphs with the same number of nodes.
3) For a given clustering, adding links within clusters should increase the score while removing them should only decrease the score.
4) For a given clustering, adding links between different clusters should decrease the score while removing them should increase the score.
5) Within a cluster, the higher the degree of a node, the more it contributes to the score.
6) Boundary nodes in a cluster that have links to other clusters have less score than internal nodes.

Given a graph clustering $C_G = \{C_1, \ldots, C_k\}$, we define the score of a vertex $i \in V$ is given by

$$f(i) = \alpha_i \times s(C(i)) \times \sigma(C(i)),$$

where $s(C(i))$ quantifies how fast a random walk on $G(C(i))$ (and then constrained to the cluster $C(i)$) reaches its steady state distribution, $\sigma(C(i))$ corresponds to the probability that a random walk on the whole graph $G$ that starts inside the cluster $C(i)$ keeps staying inside the cluster at a following step (see below for a more formal definition), and finally $\alpha_i$ differentiates among different nodes in the same cluster according to the last two properties. Given this definition of the scoring function, the score of cluster $C_u$ is:

$$f(C_u) = \left(\sum_{i \in C_u} \alpha_i\right) s_{C_u} \sigma_{C_u},$$

Below we define formally the different quantities $\alpha_i$, $s_{C_u}$ and $\sigma_{C_u}$ and show that $f(.)$ satisfies the required properties of a good clustering function.

First, we define $s_{C_u}$ as

$$s_{C_u} \triangleq s \left( (D_G(C_u) + I)^{-1}(A_G(C_u) + I) \right),$$

that is the spectral gap of the transition probability matrix of a simple random walk on the subgraph induced by the cluster nodes $C_u$. This value ranges between $0$ for a disconnected graph and $1$ for a fully connected network (a clique). Given a random walk starting at time $0$ from a node in the cluster, the difference between the probability distribution of the position of the random walker at time $t$ and its stationary distribution can be bounded by $A(1 - s_{C_u})^t$, with $A$ being an appropriate constant. Then the larger $s_{C_u}$, the faster the distribution converges to its stationary distribution, i.e. the faster the random walk mixes. The spectral gap of the transition probability matrix is then also a measure of how well connected the network within a cluster is. The presence of $s_{C_u}$ as a multiplicative factor in the scoring function guarantees that the first two properties are satisfied. Moreover, due to the interlacing property of eigenvalues [11], adding more links between the nodes of the same cluster usually increases the spectral gap while removing links decreases it, which supports the third property of a good clustering function.

Second, we define

$$\sigma_{C_u} \triangleq \sigma \left( (D_G(C_u))^{-1}A_G(C_u) \right).$$

Given that $D_G(C_u)$ considers the degrees of the nodes in the original graph $G$, $Q = D_G(C_u)^{-1}A_G(C_u)$ is a substochastic matrix. If we consider the transition probability matrix of a random walk on the whole graph $G$, $Q$ is the submatrix obtained by extracting only the rows and the columns corresponding to the nodes in $C_u$. Given a random walk on $G$ starting at a node $i$ in $C_u$, and assuming that $Q$ is a primitive matrix, it is possible to show [12] that the conditional probability distribution given that the random walk does not exit from $C_u$ converges to $\pi \in [0, 1]^{C_u}$ (we consider only the probabilities for the nodes in $C_u$, for all the other nodes the probability is clearly 0 under the conditioning event), that satisfies the following equation $\pi^TQ = \pi^T\sigma(Q)$. Then $\sigma(Q)$ can be interpreted as the probability that at each step the random walker does not exit from $C_u$, given that it has already spent a long time in $C_u$. The term $\sigma_{C_u}$ quantifies then the effect of outer links connecting the cluster $C_u$ to other clusters. Obviously, it ranges between $0$ and $1$. It is equal to $1$ when there is no link between nodes in $C_u$ and nodes in $V \setminus C_u$ and in particular when $C_u = V$ since the graph is connected.

It is equal to $0$ if the subgraph $G(C_u)$ has no links. Adding links between clusters can only decrease $\sigma$ while removing

---

1If $P$ is a scalar, we consider $s(P) = 1$ by convention.

2The inverse $D_G(C_u)^{-1}$ always exists because $D_G(C_u)$ is a diagonal matrix having strictly positive diagonal values ($d_G(i) \geq 1$ because $G$ is connected).

3Otherwise if we consider that the random walk initial position in $C_u$ follows the probability distribution $\pi$, $\sigma(Q)$ is simply the probability that the random walker does not exit from $C_u$ at each step.
them can only increase it. The factor $\sigma_{C_u}$ guarantees that the fourth property is satisfied.

Finally, $\alpha_i$ represents the contribution of a node to the final score depending on its connectivity to other clusters. To satisfy the last two properties required for the function $f$, the value $\alpha_i$ is chosen as follows:

$$\alpha_i \triangleq \frac{d_{in}^i}{1 + d_{out}^i},$$

where $d_{in}^i = d_{G(C(i))}(i)$ is the number of nodes in $C(i)$ connected to $i$ and $d_{out}^i = d_{G}(i) - d_{in}^i$ is the number of nodes in $V \setminus C(i)$ connected to $i$.

IV. Clustering Algorithm

The function $f$ presented in the previous section gives a scoring mechanism to evaluate a clustering algorithm. In particular, the optimal clustering algorithm can be written as follows:

$$\text{Argmax}_{C_G \in \{C_1, \ldots, C_k\}} f(C_G).$$

(1)

Let $C_G$ be the solution of (1) and $f^* = f(C_G^*)$ be its value. The optimal clustering and its value are computationally difficult to find, so we will give first some bounds on the optimal value $f^*$ and we will propose a local search clustering algorithm that can be implemented with an acceptable complexity and in a distributed way.

A. Bounds on $f^*$

**Proposition 1.** For the clustering optimization problem (1), the following bounds hold for the optimal value $f^*$:

$$2 \times m \times s_V \leq f^* \leq 2 \times m,$$

(2)

where $s_V$ is the spectral gap of the simple random walk on all the graph $G$ ($s_V = 1 - \lambda_2 ((D + I)^{-1}(A + I))$).

**Proof:** For any clustering $C_G = \{C_1, \ldots, C_k\}$ of the graph $G$ we have,

$$f(C_G) = \sum_i f(i) = \sum_i \frac{d_{in}^i}{1 + d_{out}^i s_C(i) \sigma_{C(i)}} \leq \sum_i \frac{d_{in}^i}{1 + d_{out}^i} \leq \sum_i d_{in}^i \leq \sum_i d_G(i) = 2 \times m,$$

where $m$ is the number of links in the graph $G$ and the first inequality follows from both $s_{C_u}$ and $\sigma_{C_u}$ being at most equal to one. From this upper bound, it follows that $f(C_G^*) \leq 2 \times m$.

The optimal clustering has a value greater than any possible clustering. Taking the graph as one cluster $C_G = \{V\}$ is a valid clustering of $G$. Thus, a lower bound on the optimal value is given as follows:

$$f^* \geq f(C_G = \{V\}) = \sum_i d_{in}^i \times s_V \times 1 = 2 \times m \times s_V,$$

where $s_V$ is the spectral gap of the simple random walk on all the graph $G$ ($s_V = 1 - \lambda_2 ((D + I)^{-1}(A + I))$).

We observe that both of the bounds are tight for the fully connected graph (let us denote it $K_n$). Indeed nodes in $K_n$ are grouped in a single cluster ($C_G = V$) and $f(V) = 2m$ since $d_{in}^i = 0$ for any vertex $i$ and $s_V = \sigma_V = 1$.

Due to the following proposition, the subgraph induced by a cluster of the optimal clustering is connected as long as it has at least an internal link.

**Proposition 2.** Let $C_G^* = \{C_1, \ldots, C_k\}$ be an optimal clustering for a graph $G$, then for any $C_u \in C_G^*$, if the subgraph $G(C_u)$ has at least one link, it is connected.

**Proof:** We sketch a proof of the proposition by contradiction. Suppose there exists a graph whose optimal clustering $C_G^*$ outputs a cluster $C_u$ such that $G(C_u)$ has at least one link, but it is disconnected. It follows that $f(C_u) = 0$ since $s_{C_u} = 0$ for disconnected graphs. However, there is a subset of vertices $H \subset C_u$ such that $|H| \geq 2$ and $G(H)$ is connected (because there is at least one link in $G(C_u)$) and it holds ($f(H) > 0$). Now if we replace $C_u$ with two clusters $H$ and $C_u - H$, the new clustering has a strictly higher value than $C_G^*$ (contradiction).

B. Local Search Clustering Algorithm

The optimal clustering can be computationally costly because calculating the spectral gap of a random walk has complexity $O(n^5)$. In this section, we present a local clustering algorithm that allows the clustering to be done in a distributed way. The algorithm applies the generic local search approach. Let $X$ be the set of all possible clusterings of graph $G$. We define two cluster $x$ and $y$ belonging to $X$ to be neighbors if and only if they differ only for a single vertex that belongs to two different clusters in $x$ and in $y$. A local search algorithm for clustering operates as follows:

1) Let $x$ be some initial clustering;
2) While there is a neighboring $G$-clustering $y$ with higher score value ($f(y) > f(x)$), set $x := y$
3) Return the final (locally optimal) solution $x$.

The algorithm is an iterative one. In our local clustering algorithm we follow the above steps but we add some randomness in choosing the neighbor in step two. In fact, at every iteration, a cluster, say it $C_u$, is chosen uniformly at random. This random cluster selects one of the outgoing links uniformly at random and proposes to the endpoint node $j$ in the adjacent cluster, to disconnect from that cluster and to join $C_u$. If joining $C_u$ can increase the value of the clustering then $j$ will accept the proposal, otherwise it will reject it and no change in the clustering will take place. In particular, a detailed description of the local clustering algorithm is given in Algorithm 1. The algorithm runs at most for $T_{stop}$ iterations, but it can easily changed so that it stops after a given number of consecutive iterations without any change of the clustering.

Algorithm 1 presents some interesting features. In fact, at every iteration, only two clusters are involved in the algorithm, while the others are idle. It is then simple to distribute the algorithm among the different clusters that can work asynchronously and in parallel as follows: at any time an
Algorithm 1 Local Clustering Algorithm

1: $G = (V, E)$ where $V = 1...n$ and $E = 1...m$.
2: Initial clustering $C^0_G = \{C_1, ..., C_n\}$ where $C_i = \{i\}$.
3: $E^+_{C_u} = \{(i, j) \in E | i \in C_u, j \notin C_u\}$ is the set of $C_u$'s outgoing links.
4: for $k = 1 : T_{stop}$ do
5:    $C^k_G = C^{k-1}_G$;
6:    let $C_u$ be a cluster chosen uniformly at random from $C^k_G$;
7:    let $(i, j)$ be a link chosen uniformly at random from $E^+_{C_u}$;
8:    let $C_j$ be the cluster containing $j$ (i.e. $C_v = C(j)$);
9:    $C_u$ proposes to $j$ to join (if it didn’t yet propose to $j$
10:   after the last change within $C_u$ occurred);
11:   if $f(C_u) + f(C_v) < f(C_u \cup \{j\}) + f(C_v \setminus \{j\})$ then
12:       $C_u \leftarrow C_u \cup \{j\}$;
13:       $C_v \leftarrow C_v \setminus \{j\}$;
14:   if $C(j) = \emptyset$ then
15:       Remove $C_v$ from $C^k_G$;
16:   end if
17: else
18:    $j$ rejects the proposal;
19: end if
20: $k \leftarrow k + 1$;
21: if all clusters don’t have any more proposals break;
22: end for
23: return $C_{k-1}^G$

inactive cluster can wake up and can propose to a node from another inactive cluster to join it, both clusters will become active until acceptance or rejection of the proposal. Several matching clusters can be active at the same time and the computations is distributed in a parallel way. Finally, being that the algorithm randomized, it is possible to run it multiple times and then select the best solution across the different runs.

Moreover, at every iteration, a cluster can increase by maximum one node. The complexity of the algorithm originates from calculating the function $f$ which in its turn depends on the number of nodes in the cluster. So depending on the available computational power, we can restrict the maximum number of nodes in a cluster. For example, if the calculation of the spectral gap is affordable for graphs with only few hundred nodes, then clusters reaching this limit will stop initiating the algorithm and stop proposing to other nodes to join.

In addition, the local clustering algorithm performs well on clique-like graphs. The following simple lemma will prepare the result:

**Lemma 1.** Let $g : A \rightarrow \mathbb{R}$ be a scalar strongly convex function, then for any $x$ and $y$ such that $x, x + 1, y, y - 1 \in A$ and $x \geq y$, we have:

$$g(x + 1) + g(y - 1) > g(x) + g(y).$$

**Proof:** Let $h(x) = g(x + 1) - g(x)$, since $g$ is strongly convex, then $g'(x)$ is strictly increasing, so

$$x + 1 > x,$$

$$g'(x + 1) > g'(x),$$

$$h'(x) > 0,$$

and adding that $x \geq y$ we can write:

$$x > y - 1,$$

$$h(x) > h(y - 1),$$

$$g(x + 1) - g(x) > g(y) - g(y - 1),$$

and the lemma follows. ■

**Proposition 3.** The local clustering Algorithm 1 calculates the optimal clustering for a clique graph $K_n$ in a finite number of iterations almost surely if $T_{stop}$ is large enough, i.e., Algorithm 1 on $K_n$ outputs a single cluster $C^*_G = \{V\}$.

**Proof:** First note that the optimal clustering on a clique $K_n$ is $C^*_G = \{V\}$ since $f(C_G) = |V|$ is an upper bound on $f^*$. It remains to prove that the local algorithm terminates with one cluster of all nodes. Let $C_u$ be any cluster in this graph, and let $n_u = |C_u|$ be its number of vertices, so $s_{C_u} = 1$ since the subgraph induced by $C_u$ is also a clique, and $\sigma_{C_u} = \frac{n_u - 1}{n_u}$ since the matrix $D_{C}(C_u)^{-1}A_{C}(C_u)$ has dimensions $n_u \times n_u$ and any of its elements has the value $\frac{1}{n_u}$ except the diagonal elements that are equal to 0, therefore

$$f(C_u) = \left(\sum_{i \in C_u} \frac{d_{in}^{(u)}}{1 + d_{out}^{(u)}}\right) s_{C_u}\sigma_{C_u}$$

$$= \left(\sum_{i \in C_u} \frac{n_u - 1}{1 + n - n_u}\right) \times \frac{n_u - 1}{n}$$

$$= \frac{n_u(n_u - 1)^2}{(n - 1)(n - n_u + 1)},$$

and it depends only on the size of the cluster. Let $g(n_u) = f(C_u)$, since $g(n_u)$ is strongly convex in $n_u$ when $n_u \in [1, n]$, then according to the algorithm and due to Lemma 1, any node $j$ (that belongs to the cluster $C_v$) receiving a proposal from a cluster $C_u$ will accept this proposal if $|C_u| \geq |C_v|$ and will reject otherwise due to the following equation,

$$f(C_u \cup \{j\}) + f(C_v \setminus \{j\}) = g(|C_u| + 1) + g(|C_v| - 1)$$

(3)

$$> g(|C_u|) + g(|C_v|)$$

(4)

$$= f(C_u) + f(C_v).$$

(5)

The transition from (3) to (4) is due to Lemma 1. Therefore, any proposal from the cluster with largest number of vertices to other nodes is accepted (let $C_{max}$ be the cluster with maximum number of vertices at iteration $k$), $|C_{max}|$ cannot decrease while it can increase by one with a probability larger
than $1/n$. The algorithm terminates when $|C^k_{\text{max}}| = n$, so with probability 1 there is an iteration $K$ such that all the nodes form a single cluster and the algorithm terminates. It is easy to check that $E(K) \leq n^2$.

V. NUMERICAL EXAMPLES

In this part, we study the performance of our local clustering algorithm. We consider real world networks whose ground truth is known. We apply our algorithm (we perform multiple independent runs of the Algorithm 1 and select the best local maximum) on these networks and compare the algorithm’s results with actual clustering. The results are shown using the graph visualization platform Gephi [13]. We also compare our results with the built-in modularity clustering algorithm [14] in Gephi. Our first example is the Zachary’s Karate Club [15], it is a social network of friendships between 34 members of a karate club at a US university in the 70s. Fig. 1 shows the partition of the karate club.

We apply our local clustering algorithm to the karate club network and the results are given in Fig. 2. Starting from 34 different clusters as initial input (every node is considered a cluster) and based on the connection and the spectral gap of the clusters, our algorithm identifies 3 clusters (one more than the ground truth). Moreover the two nodes 31 and 9 are not assigned to the correct cluster. Notice that this is just a local maximum for the optimization problem. For comparison, Fig. 3 shows the results of clustering using the modularity clustering algorithm, we see that it identifies even more clusters than our method (4) and node 10 is not correctly assigned in comparison to the ground truth.

The other example we consider is the network of American College football teams in Division I during Fall 2000 regular season [16]. Division I was made up by 115 teams divided in 12 conferences. A link in the graph corresponds to a game played between the two teams. Teams in the same conference are more likely to play games than teams from different conferences. Fig. 4 shows the teams grouped according to the conference they belong to. While most of conferences have good clustering properties (good connections inside the clusters and weak connections among them), there are some conferences for which this is not true. For conference 1 for example there is only one game (one link) among its members, and the clubs have played most of their games against teams in different conferences. In those cases we expect the clustering algorithm to classify the nodes into different clusters.

We applied our local clustering algorithm to this network. The results are shown in Fig. 5. The local clustering algorithm gives 14 clusters, and we see that the algorithm was able to find correctly most of the clusters. In particular, the difference
between the ground truth and the spectral gap clustering is as follows: cluster 7 was divided into two clusters, cluster 12 was also divided into two clusters. Even though conference 1 is very difficult to identify, our algorithm clustered together the only two connected nodes and clustered the disconnected nodes into different clusters. In total there are only 6 nodes that are not well clustered\(^4\)(out of the 115 nodes).

We also present the results of clustering using the modularity algorithm of [14]. This allows us to compare the performance with other clustering algorithm and to check if the errors were due to failure of the algorithm or due to the ground truth graph structure. In the Fig. 6, the modularity algorithm classified the network into only 10 clusters (2 less clusters than the ground truth). Cluster 7 nodes were divided between two already existing clusters. Cluster 1 disappeared. Note that the same 6 nodes that were miss-classified by our algorithm were also here miss-classified which suggests that the errors are due to the structure but not to the algorithm.

VI. CONCLUSION

In this paper we proposed a new clustering metric based on the spectral gap of a random walk on clusters. We also proposed a randomized local clustering algorithm that outputs a locally optimal clustering of the graph. The algorithm can be distributed in a network and clusters are iteratively updated on the basis of local communication and processing. One of the strengths of our algorithm is its ability to detect small clusters. The complexity can also be adapted to available processing capabilities.

\(^4\)The bad clustered nodes by Algorithm 1 in comparison to the ground truth are: 3 nodes in cluster 1, 2 nodes in cluster 9, and 1 node in cluster 5 which gives a total of 6 error nodes (without taken into consideration the split of the clusters 7 and 12).

Fig. 5. Clustering the American College football network by applying Algorithm 1. Nodes with the same color are classified as one cluster (the algorithm terminates with 14 clusters, 2 more than the ground truth).

Fig. 6. Clustering the American College football network by applying the modularity algorithm. Nodes with the same color are classified as one cluster (the algorithm terminates with 10 clusters).

REFERENCES