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_chamie, 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.

To Compare different graph clustering outputs, it is necessary to introduce a quality metric, that is also called the objective function. There is still no consensus on which metric is the best one. One of the most used metrics is modularity [1] which gives a score to the cluster by comparing the number of edges falling inside the clusters with the number of edges of a random graph having similar characteristic as the original one. One of its drawbacks is that it cannot distinguish small clusters having links of order $O(\sqrt{m})$ where m is the total number of links [2]. The silhouette index [3] uses distances between the nodes presented in the cluster and those outside it, its drawback being its high computational cost as it requires to compute the shortest path between all node pairs. Another approach [4] evaluates a clustering score by using the concept of inter-cluster conductance, but it ignores internal cluster density. Using this metric, some graph partitioning algorithms based on PageRank vectors of a graph have been proposed in [5] to find a cut with a certain conductance in the graph. All these metrics turn out to be biased toward large communities [6]. Many practical algorithms have been proposed as hierarchical clustering [7], Markov clustering [8], bisecting K-means, and spectral clustering [4]. Their drawback is that they are global clustering methods which require as input the entire graph to calculate the clustering. Moreover their output is biased toward equal size clusters (so small communities tend to disappear using these algorithms). A complete survey of fitness measures and clustering is given in [9].

In this paper, we introduce a new fitness measure for evaluating a clustering algorithm based on random walks' properties. Roughly speaking, our fitness index is higher the faster a random walk constrained to the cluster reaches its stationary 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.

The organization of the paper is as follows: in Section II we present the notation used across the paper, in Section III we introduce our new fitness measure. Section IV describes the local clustering algorithm. Section V compares its performance on different networks. Section VI concludes the paper.

II. Notation

Let $G = (V, E)$ be an undirected unweighted connected graph withoug self-loops, where $V = \{1, \dots n\}$ is the set of vertices and E is the set of $m = |E|$ edges. Let $d_G(i) = |\{j \in V \text{ such that } (j, i) \in E\}|$ 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 $G(S)$. Conversely, $A_G(S) = A_{G(S)}$ as the adjacency matrix is not changed. If P is a substochastic matrix (a square matrix with nonnegative entries so that every row adds up to at most

1), let $\sigma(P) = |\lambda_1(P)|$ be the largest eigenvalue in module of P. When P is stochastic, let $s(P) = 1 - |\lambda_2(P)| \in [0, 1]$ be its spectral gap[1] where $\lambda_2(P)$ is the second largest eigenvalue in module of P. Finally, I is the identity matrix.

A clustering $\mathcal{C}_G$ of a graph G is a partition of the vertices such that $\mathcal{C}_G = \{C_1, \dots C_k\}$ where $C_1 \cup ... \cup C_k = V$ and $C_u \cap C_v = \phi$ for all clusters C_u and C_v in $\mathcal{C}_G$. Let $C(i) = \{C_u \in \mathcal{C}_G; i \in C_u\}$ be the cluster that contains node i. Let $f : V \to \mathbb{R}$ 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 $f(\mathcal{C}_G) = \sum_{u=1}^{k} f(C_u) = \sum_{i=1}^{n} f(i)$.

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.

The new scoring metric we propose satisfies the properties above. Given a graph clustering $\mathcal{C}_G = \{C_1, ..., C_k\}$, 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 α_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 α_i, s_{C_u} and σ_{C_u} and show that $f(.)$ has all the required properties.

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

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 adding self-loops [10]. 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[2] 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 $\pi^T Q = \pi^T \sigma(Q)$. 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[3]. The term σ_{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 $V \backslash C_u$ and then in particular when $C_u = V$ since the graph is connected. It is equal to 0 if the subgraph $G(C_u)$ has no link. Adding links between clusters can only decrease σ while removing them can only increase it. The factor σ_{C_u} guarantees that the fourth property is satisfied.

[2]The 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).

[3]Otherwise if we consider that the random walk initial position in C_u follows the probability distribution π, $\sigma(Q)$ is simply the probability that the random walker does not exit from C_u at each step.

Finally, α_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 α_i is chosen as follows:

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

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

$$\underset{\mathcal{C}_G=\{C_1,...,C_k\}}{\text{Argmax}} \quad f(\mathcal{C}_G) \tag{1}$$

Let $\mathcal{C}_G^*$ be the solution of (1) and $f^* = f(\mathcal{C}_G^*)$ be its value. Finding the optimal clustering and its value can be very expensive, 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^*:*

$$2 \times m \times s_V \leq f^* \leq 2 \times m, \tag{2}$$

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

Proof: For any clustering $\mathcal{C}_G = \{C_1, ..., C_k\}$ of the graph G we have,

$$\begin{aligned} f(\mathcal{C}_G) &= \sum_i f(i) = \sum_i \frac{d_i^{in}}{1+d_i^{out}} s_{C(i)} \sigma_{C(i)} \\ &\leq \sum_i \frac{d_i^{in}}{1+d_i^{out}} \leq \sum_i d_i^{in} \leq \sum_i d_G(i) \\ &= 2 \times m, \end{aligned}$$

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

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

$$\begin{aligned} f^* \geq f(\mathcal{C}_G = \{V\}) &= \sum_i d_i^{in} \times s_V \times 1 \\ &= 2 \times m \times s_V, \end{aligned}$$

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

We observe that both 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 ($\mathcal{C}_G = V$) and $f(V) = 2m$ since $d_i^{out} = 0$ 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 $\mathcal{C}_G^* = \{C_1, ..., C_k\}$ be an optimal clustering for a graph G, then for any $C_u \in \mathcal{C}_G^*$, 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 $\mathcal{C}_G^*$ 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 $\mathcal{C}_G^*$ (contradiction). ■

B. Local Search Clustering Algorithm

The optimal clustering can be computationally costly because calculating the spectral gap of a random walks has complexity $O(n^3)$. In this section, we present a local clustering algorithm that allows the clustering to be done in a distributed way. In particular clusters can be determined in parallel. 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

Algorithm 1 Local Clustering Algorithm

$G = (V, E)$ where $V = 1...n$ and $E = 1...m$.
Initial clustering $\mathcal{C}_G^0 = \{C_1, ..., C_n\}$ where $C_i = \{i\}$.
$E^+_{C_u} = \{(i,j) \in E | i \in C_u, j \notin C_u\}$ is the set of C_u's outgoing links.
for $k = 1 : T_{stop}$ **do**
  $\mathcal{C}_G^k = \mathcal{C}_G^{k-1}$;
  let C_u be a cluster chosen uniformly at random from $\mathcal{C}_G^k$;
  let (i,j) be a link chosen uniformly at random from $E^+_{C_u}$;
  let C_v be the cluster containing j (i.e. $C_v = C(j)$);
  C_u proposes to j to join (if it didn't yet propose to j after the last change within C_u occurred) ;
  if $f(C_u) + f(C_v) < f(C_u \cup \{j\}) + f(C_v \backslash \{j\})$ **then**
    j accepts the proposal;
    $C_u \leftarrow C_u \cup \{j\}$;
    $C_v \leftarrow C_v \backslash \{j\}$;
    if $C(j) = \phi$ **then**
      Remove C_v from $\mathcal{C}_G^k$;
    end if
  else
    j rejects the proposal;
  end if
  $k \leftarrow k + 1$;
  If all clusters don't have any more proposals **break**;
end for
return $\mathcal{C}_G^{k-1}$

asynchronously and in parallel as follows: at any time an 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. It is also possible that at every time each cluster is matched to another one and evaluates the possibility to acquire/yield a node to it. several matching clusters can be active at the same time and the computations is distributed in a parallel way. Finally, being the algorithm randomized, it is possible to run it multiple times and then select the best solution across all 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 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 $g : A \to \mathbb{R}$ be a scalar strongly convex function ($g''(x) > 0$), 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 convex, then $g'(x)$ is strictly increasing, so

$$\begin{aligned} & x + 1 > x, \\ \Rightarrow\ & g'(x+1) > g'(x), \\ \Rightarrow\ & h'(x) > 0, \text{ so } h(x) \text{ is strictly increasing,} \end{aligned}$$

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

$$\begin{aligned} & x > y - 1, \\ \Rightarrow\ & h(x) > h(y-1), \\ \Rightarrow\ & g(x+1) - g(x) > g(y) - g(y-1), \end{aligned}$$

and the lemma follows. ∎

Now we show the following proposition:

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 $\mathcal{C}_G = \{V\}$.*

Proof: First note that the optimal clustering on a clique K_n is $\mathcal{C}_G^* = \{V\}$ since $f(\mathcal{C}_G = \{V\}) = 2m$ that 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 $\sigma_{C_u} = \frac{n_u - 1}{n-1}$ since the matrix $D_G(C_u)^{-1} A_G(C_u)$ has dimensions $n_u \times n_u$ and any of its elements has the value $\frac{1}{n-1}$ except the diagonal elements that are equal to 0, therefore

$$\begin{aligned} f(C_u) &= \left(\sum_{i \in C_u} \frac{d_i^{in}}{1 + d_i^{out}} \right) s_{C_u} \sigma_{C_u} \\ &= \left(\sum_{i \in C_u} \frac{n_u - 1}{1 + n - n_u} \right) \times 1 \times \frac{n_u - 1}{n - 1} \\ &= \frac{n_u (n_u - 1)^2}{(n-1)(n - n_u + 1)}, \end{aligned}$$

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,

$$\begin{aligned} f(C_u \cup \{j\}) + f(C_v \backslash \{j\}) &= g(|C_u| + 1) + g(|C_v| - 1) && (3) \\ &> g(|C_u|) + g(|C_v|) && (4) \\ &= f(C_u) + f(C_v). && (5) \end{aligned}$$

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}^k be the cluster with maximum number of vertices at iteration k), $|C_{max}^k|$ cannot

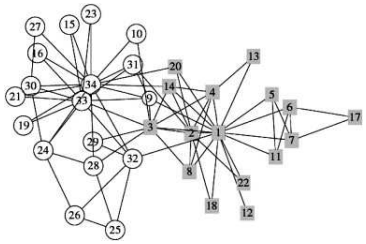

Fig. 1. The network of social relationship between the members of the Karate Club. After the split, the members represented by a square belongs to one sub-club and the members represented by a circle to the other sub-club (the image is taken from [1]).

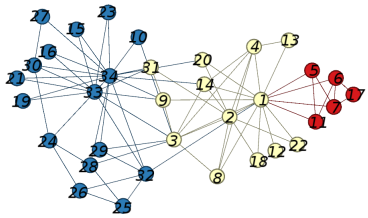

Fig. 2. Clustering the karate club by applying Algorithm 1.

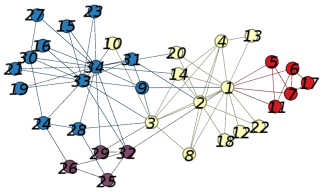

Fig. 3. Clustering the karate club by applying modularity algorithm.

decrease while it can increase by one with a probability larger than $1/n$. The algorithm terminates when $|C^k_{max}| = n$, 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 $\mathbb{E}(K) \leq n^2$. ■

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 on these networks and compare the algorithm's results with actual clustering. Our first example will be the Zachary's Karate Club [13], it is a social network of friendships between 34 members of a karate club at a US university in the 70s. Links in this network represent social interactions outside of the club itself. Due to a conflict, the club was split into two sub-clubs in which the members moved to one of the two new generated groups. 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 [14], 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 [15]. 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 week 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

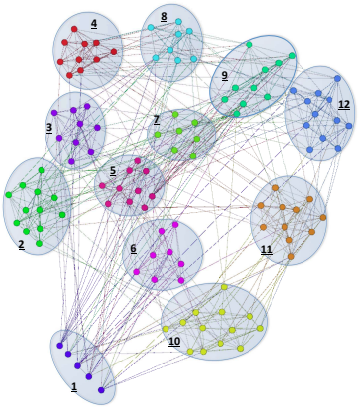

Fig. 4. The ground truth of the conferences (clusters) in the American College football network.

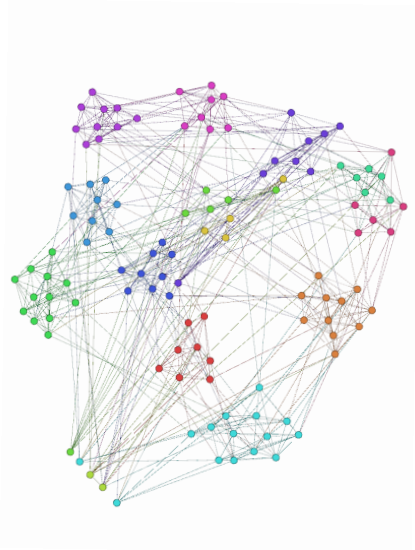

Fig. 5. Clustering the American College football network by applying Algorithm 1. Nodes with the same color are classified as one cluster (the algorithm terminates with 14 clusters, 2 more than the ground truth).

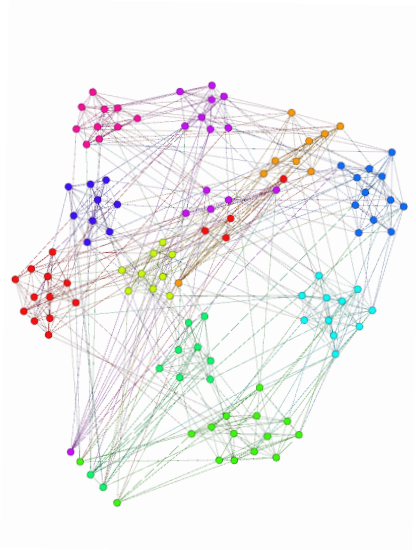

Fig. 6. Clustering the American College football network by applying the modularity algorithm. Nodes with the same color are classified as one cluster (the algorithm terminates with 10 clusters).

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).

References

[1] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, Feb 2004.

[2] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, Jan. 2007.

[3] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[4] R. Kannan, S. Vempala, and A. Veta, "On clusterings-good, bad and spectral," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, ser. FOCS '00. Washington, DC, USA: IEEE Computer Society, 2000, p. 367.

[5] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Foundations of Computer Science, 2006. FOCS '06. 47th Annual IEEE Symposium on*, 2006, pp. 475–486.

[6] H. Almeida, D. Guedes, W. Meira, and M. J. Zaki, "Is there a best quality metric for graph clusters?" in *Proceedings of the 2011 ECML PKDD'11*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 44–59.

[7] M. Krivánek and J. Morávek, "Np-hard problems in hierarchical-tree clustering," *Acta Informatica*, vol. 23, no. 3, pp. 311–323, 1986.

[8] S. Van Dongen, "Graph clustering via a discrete uncoupling process," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 1, pp. 121–141, Feb. 2008.

[9] S. E. Schaeffer, "Survey: Graph clustering," *Comput. Sci. Rev.*, vol. 1, no. 1, pp. 27–64, Aug. 2007.

[10] H. Landau and A. Odlyzko, "Bounds for eigenvalues of certain stochastic matrices," *Linear Algebra and its Applications*, vol. 38, no. 0, pp. 5 – 15, 1981.

[11] G. Chen, G. Davis, F. Hall, Z. Li, K. Patel, and M. Stewart, "An interlacing result on normalized laplacians," *SIAM J. Discret. Math.*, vol. 18, no. 2, pp. 353–361, Feb. 2005.

[12] J. N. Darroch and E. Seneta, "On quasi-stationary distributions in absorbing discrete-time finite markov chains," *Journal of Applied Probability*, vol. 2, no. 1, pp. pp. 88–100, 1965. [Online]. Available: http://www.jstor.org/stable/3211876

[13] W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, pp. 452–473, 1977.

[14] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[15] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.