SSDE-Cluster: Fast Overlapping Clustering of Networks Using Sampled Spectral Distance Embedding and GMMs

Malik Magdon-Ismail
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
magdon@cs.rpi.edu

Jonathan Purnell
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
purnej@cs.rpi.edu

***Abstract*—**

Clustering social networks is vital to understanding online interactions and influence. This task becomes more difficult when communities overlap, and when the social networks become extremely large. We present an efficient algorithm for constructing overlapping clusters, (approximately linear). The algorithm first embeds the graph and then performs a metric clustering using a Gaussian Mixture Model (GMM). We evaluate the algorithm on the DBLP paper-paper network which consists of about 1 million nodes and over 30 million edges; we can cluster this network in under 20 minutes on a modest single CPU machine.

I. Introduction

Social, collaboration and interaction networks (Facebook, DBLP, Amazon, etc.) are a source of vast, multimodal information. Such networks are used in order to communicate ideas broadly and quickly. Identifying communities in a network is a fundamental task, where nodes represent the entities and edges represent the interactions between nodes (e.g. users and similarity of purchases). This task is complicated by the fact that entities may belong to more than one community. Clustering algorithms have provided the prominent tools for identifying communities and have been extended for allowing overlap. However, these networks are growing larger in size and detail every day. Thus algorithms must be scalable since even quadratic runtimes can become impractical when networks have over 1 million nodes and tens of millions of edges.

Since there are a plethora of definitions for *community* in a social network, we construct an intuitive algorithm that uses a particular metric. This metric is defined by available data and assumptions made on the structure of a community. Our metric is based on homophily [1], favoring communities that are internally similar while dissimilar from other communities. The algorithm uses the metric to assign nodes to overlapping clusters. Validation is whether, in real social networks, the algorithm can produce good intuitive results.

Overview of SSDE-Cluster The input is a (typically sparse) social network graph $G = (V, E)$ on n nodes. Edges are weighted relative to the relationship between the nodes. Typically, the definition of the edge weights is application-specific and a determining factor for the quality of the results.

Our approach has two phases: The first phase efficiently embeds this metric in $\mathbb{R}^d$, so that Euclidean distances reasonably approximate the graph metric. The second phase is to cluster the embedded data. We use a Gaussian Mixture Model (GMM) because it can be adapted to give overlapping clusters.

II. Related Work

Finding communities in social networks has been rapidly researched during the past decade [2], [3], [4]. Early work used 'small world', power-law, and network transitivity properties. More recently proposed is the connectedness of communities, where communities are densely connected subgraphs that are sparsely connected to each other [5], [6].

To address efficiency, algorithms have been extended to work within subspaces [7], identify overlapping groups using local optimality [8], and remove the constraints on the number of communities [9]. One of the most prominent methods is clique-percolation [10], [11].

In this paper we propose an algorithm with the following features:

- *Efficiency:* Scales linearly with the number of entities, edges, dimensions of embedding, and communities.
- *Overlapping:* The GMM provides a measure of similarity of each entity to each community. Entities can be properly assigned to multiple communities, during or after the clustering.
- *Unsupervised:* Limited apriori knowledge is needed. SSDE-Cluster discovers the community structure given only the number of communities in the network. We also discuss how to estimate this number.

The work in [12], [13] is similar to ours in that it starts with spectral properties, similar to multi-dimensional scaling (MDS). However, they construct the Laplacian, which has $\Omega(n^2)$ runtime. A comprehensive review of community detection is given in [2].

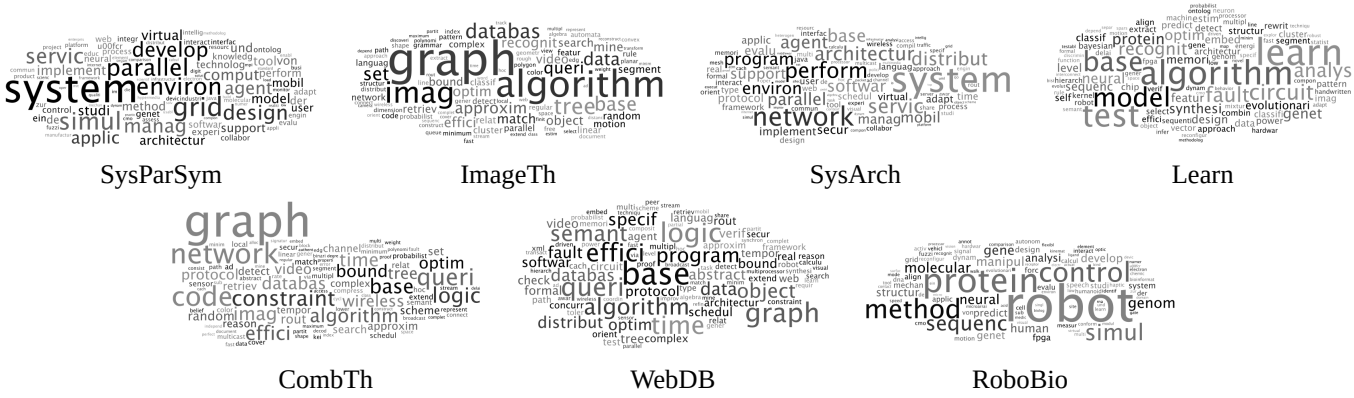


Fig. 1. Keyword Clouds for the 7 clusters. For each cluster, the titles of each included paper are processed into a word cloud where a word's font size increases proportionately to its frequency.

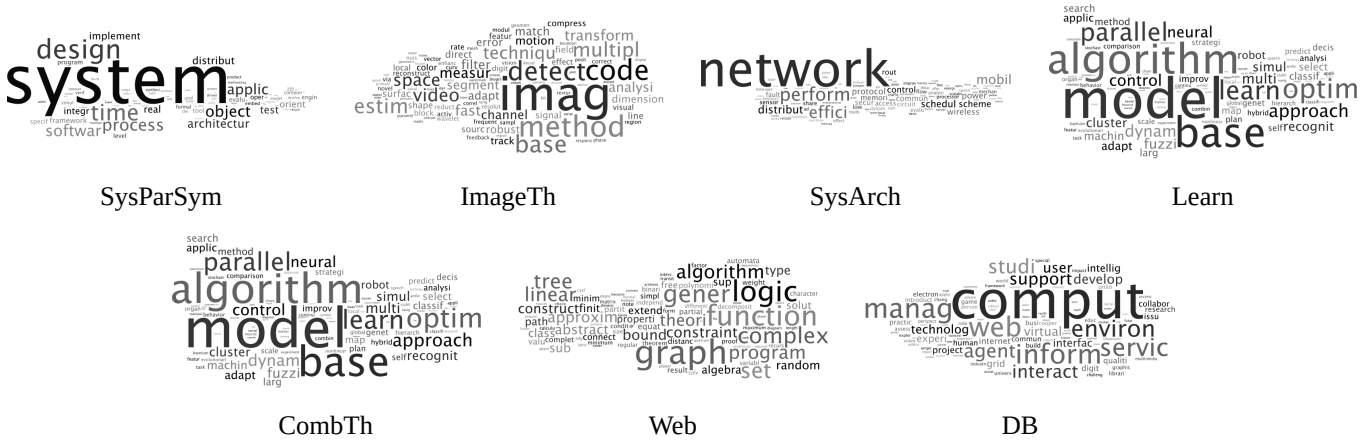


Fig. 2. LDA conference clouds for 7 clusters. These are conference clouds that were generated based on the groups that LDA discovered.

III. The SSDE-Cluster Algorithm

The input is a weighted graph, $G = (V, E)$. The algorithm can be broken down as follows.

1. Run SSDE (spectral embedding) to approximately embed the graph in $d \ll n$ dimensions.
2. Fit a GMM to the embedded data; determine K, the number of clusters, by comparing the marginal value of adding a cluster on the real data with random data.
3. Use the GMM posterior probabilities to construct overlapping clusters; determine the degree of cluster overlap by locally optimizing a cluster density.

A. Sampled Spectral Distance Embedding

Sampled Spectral Distance Embedding (SSDE) is an approximation to classical multidimensional scaling which was introduced in the context of fast graph drawing [14]. The distance matrix $\mathbf{D}$ is the symmetric $n \times n$ matrix containing all the pair-wise distances. Suppose we position vertex v_i at $\mathbf{x}_i \in \mathbb{R}^d$. We are seeking a positioning that approximates the graph theoretical distances with the Euclidean distances, i.e.,

$$||\mathbf{x}_i - \mathbf{x}_j|| \approx D_{ij}, i, j = 1, 2, \ldots, n \tag{1}$$

After squaring and some manipulation (see [14]), one obtains the MDS equation:

$$\mathbf{Y}\mathbf{Y}^T \approx -\frac{1}{2}\gamma\mathbf{L}\gamma = \mathbf{M} \tag{2}$$

where the embedding $\mathbf{Y}$ is an $n \times d$ matrix containing the coordinates of the points, $L_{ij} = D_{ij}^2$ (the squares of the distances) and the centering projection matrix $\gamma = \mathbf{I}_n - \frac{1}{n}1_n 1_n^T$.

In a previous work [14], two lemmas were combined to show that any n-point finite metric is approximately embeddable in $d = O(\ln n/\epsilon^2)$ dimensions, which means that the "numerical" rank of $\mathbf{L}$ is roughly $d + 2 = O(\ln n/\epsilon^2)$. Practically, this means that a small number (say $c = 100$) suitably chosen rows of $\mathbf{L}$ captures all the information in $\mathbf{L}$. SSDE works in three steps (for details and the lemmas, see [14]). The spectral decomposition can be computed in $O(nc^2 + c^3)$. In practice, we may only need (say) the top 5 dimensions, and so a power iteration could be used in lieu of SVD (as was done in [14]).

B. Fitting a Gaussian Mixture Model (GMM)

Our next task is to cluster the nodes. We choose a standard GMM trained using the E-M algorithm. A GMM is a

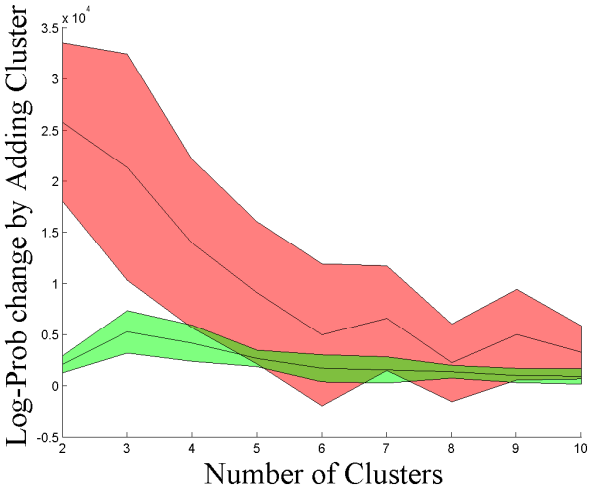


Fig. 3. Determining the # clusters. As additional clusters are added, the rate of change in the model's log-prob decreases. The green region shows changes in a model fit to random data; the red region for actual data.

weighted sum of M component densities $p_1(\mathbf{x}), \ldots, p_M(\mathbf{x})$; each component density is a d variate Gaussian function with mean $\boldsymbol{\mu}_k$ and covariance matrix Σ_k:

$$p_k(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\Sigma_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)}.$$

The GMM is $p(\mathbf{x}) = \sum_{k=1}^{M} \pi_k p_k(\mathbf{x})$, where the mixture weights satisfy $\sum_{k=1}^{M} \pi_k = 1$.

We use a simple E-M procedure for training. We initialize the GMM by selecting $\boldsymbol{\mu}_k$ randomly without replacement, and setting $\pi_k = 1/M$ and $\Sigma_k = \mathbf{I}_d$. In the expectation step, the probabilities for node i to belong to cluster k are calculated by $p_{ki} = \pi_k p_k(\mathbf{x}_i)$. We then update the parameters using these probabilities during the maximization step. These two steps are repeated until some end criterion is reached. Details of these steps are well-known [15]. One iteration of E-M is $O(nMd^2)$, but [16] discusses methods for reducing this to $O(nMd)$ using low rank perturbations to the covariance matrix. The resulting set of posterior probabilities $\{p_{ki}\}$ are used to compute *overlapping* clusters.

Determining the Number of Clusters Using too many clusters results in over-fitting the data and breaking the true communities into several smaller communities. Too few causes communities to be lumped together. We use a simple heuristic for estimating the proper number of clusters in the network. This involves comparing the real data clusters to random data clusters.

We generate a random data set whose points are randomly and uniformly distributed over the same space as the embedded data. A model is fitted to the random data and we measure the increase in log-likelihood of the model as we add clusters. Since there are no clusters in the random data, the increase in log-likelihood by adding an additional cluster is purely from over-fitting. When the benefit from adding a cluster to the real data is not significantly more than that of adding a cluster to random data, we argue that it is time to stop adding clusters. Figure 3 shows the successive gains of adding a cluster to the DBLP data versus to the random data; from this, we discern that about 7-11 clusters is appropriate (we used 7 in our experiments).

C. Assigning Nodes to Overlapping Clusters

Every node i is assigned to its most likely cluster, $\mathrm{argmax}_k\, p_{ki}$. This constructs a partition of the nodes into clusters. We now extend this so that clusters may overlap. For node i, define $\alpha_{ki} = p_{ki}/\max_k p_{ki}$; α_{ki} measures how diffuse node i's membership is with respect to its most likely cluster. Assume we have a cluster metric which measures the quality of a cluster. In our experiments, we used

$$E(C) = \lambda \frac{W_{\text{in}}(C)}{W_{\text{in}}(C) + W_{\text{out}}(C)} + (1-\lambda)\frac{W_{\text{in}}(C)}{|C|(|C|-1)};$$

W_{in} (resp. W_{out}) is the sum of similarities within the cluster (resp. from within to outside); $E(C)$ combines similarity internally and to the outside with the average internal similarity. We used $\lambda = \frac{1}{2}$.

We use α_{ki} to define an ordering over node-cluster assignment pairs, starting with the highest α. When evaluating a node-cluster pair, we also calculate the expected change in the metric value for the cluster. This is determined by finding the change in adding an *average* node; a node which has degree and edge weights equal to the respective global averages and has $\frac{|C|}{n}$ of its edges connected to nodes in the cluster. We process all node-cluster pairs sequentially in descending value of α to ensure that the more probable assignments are made first. This is similar in spirit to the local optimizations performed in [8], [17].

IV. The DBLP Network

In general, the social network is constructed by defining the "agents" (nodes) and the interactions or relationships between them. We apply our algorithm to the Digital Bibliography and Library Project (DBLP) data [18]; we choose papers as nodes, and two papers are related if they have common authors. We use the Jaccard index of the author sets to define similarity; for two papers i, j, let A_i, A_j be their respective author sets. Then

$$s_{ij} = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}; \qquad d_{ij} = \frac{1}{s_{ij}}$$

Most clustering algorithms for social networks work with the similarities $\{s_{ij}\}$. Since our algorithms are metric based and need a difference measure, we use the inverse-similarity.

A. Validating Clusters

The recurring problem with applying a clustering algorithm to real data (where the "definition" of the cluster is the "result of the algorithm") is to validate these clusters as good. We use human judgment based on the title and venue information of the papers. We preprocess title texts by removing stop words and stemming [19]. Visually, we depict the descriptive words using word clouds, and compare with an LDA analysis of the title data. Note that in general networks, such an LDA analysis is not possible.

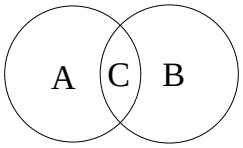
	SysParSim	ImageTh	SysArch	Learn	CombTh	WebDB	RoboBio
SysPS	**2.5**	2.4,1.7,1.3	2.4,1.7,1.3	2.4,1.7,1.3	2.4,1.6,1.3	2.5,1.4,1.0	2.4,1.7,1.2
ITh		**2.3**	2.4,1.8,1.5	2.4,1.7,1.3	2.4,1.7,1.5	2.4,1.6,1.3	2.3,1.7,1.5
SysA			**2.4**	2.4,1.7,1.5	2.4,1.8,1.5	2.4,1.8,1.4	2.4,1.8,1.5
Lrn				**2.4**	2.4,1.6,1.4	2.4,1.5,1.2	2.4,1.7,1.4
CTh					**2.4**	2.4,1.7,1.2	2.4,1.8,1.5
WbDB						**2.5**	2.4,1.9,1.4
RBio							**2.4**

TABLE I
CLUSTER QUALITY. BETWEEN TWO CLUSTERS, WE HAVE THREE MEASURES: THE AVG. WEIGHT OF EDGES WITHIN THE CLUSTERS, THE AVG. WEIGHT OF EDGES BETWEEN A CLUSTER AND THE INTERSECTION OF THE TWO CLUSTERS, AND THE AVG. WEIGHT OF EDGES BETWEEN THE CLUSTERS. IDEALLY THESE VALUES DECREASE IN ORDER.

V. EXPERIMENTAL RESULTS

For our study, we constructed the DBLP network as described above and clustered the largest connected component, which consisted of about 900K papers (nodes) and over 30 million (weighted) edges. We chose $c = 25$ for the SSDE phase and then chose $d = 5$ for the embedding. We clustered using the GMM into 7 clusters (as discussed earlier). The process of embedding and clustering took under an 20 min on a single 2.0Ghz CPU machine with 2GB of memory.

Cluster and Overlap Quality. A typical pair of clusters is illustrated here. With respect to the paper-paper similarity, we can measure the average inside a set (A and B) as compared with the average between a set and the intersection (A, C and B, C) and the average between sets (A, B). We expect:



$$\frac{1}{2}(A, A+B, B) \geq \frac{1}{2}(A, C+B, C) \geq (A, B) \qquad (3)$$

where X, Y is the average weight over edges that have one node in X and one in Y. This can be observed for our overlapping clusters (see Table I), which lends credence to their validity.

Since the clustering was done with paper-paper similarities based on authorship overlap, we may validate the clusters by looking at the text-based and conference based topics they represent (as discussed earlier). Figure 1 shows the 7 word clouds representing the clusters. Additionally, we show the resulting word clouds from using LDA in Figure 2. Again, we note that while LDA needs text, SSDE-Cluster does not use it in any way. Qualitatively, we see that the results are similar.

VI. CONCLUSION

We presented an algorithm to efficiently find overlapping communities in very large social networks with very little apriori knowledge. Our algorithm involves two fast and linear-time phases (ignoring a log factor in the SSSP task for weighted graphs). The graph is only used as a proxy for computing distances; SSDE-Cluster can be applied with any other proxy for distance. Efficient implementations of all our algorithms can be found online [20].

Acknowledgement This material is based upon work partially sponsored by: U.S. DHS through ONR grant number N00014-07-1-0150 to Rutgers University and continues under the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053.

REFERENCES

[1] M. McPherson, L. Smith-Lovin, and J. Cook, "Birds of a feather: Homophily in social networks," *Rev. Soc.*, vol. 27, pp. 415–444, 2001.
[2] S. Fortunato, "Community detection in graphs," *Phys. Rep. 486, 75-174 (2010)*, vol. 486, pp. 75–174, 2010.
[3] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks." *PNAS*, vol. 99, no. 12, pp. 7821–7826, June 2002.
[4] M. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.
[5] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, no. 2, pp. 1–15, February 2004.
[6] S. Gregory, "An algorithm to find overlapping community structure in networks," *Proc. PKDD*, pp. 91–102, 2007.
[7] K. Sequeira and M. Zaki, "SCHISM: a new approach to interesting subspace mining," *Bus. Intel. & Data Min.*, vol. 1, no. 2, p. 137, 2005.
[8] J. Baumes, M. Goldberg, and M. Magdon-Ismail, "Efficient Identification of overlapping communities," *ISI*, vol. 1, no. 2, pp. 27–36, 2005.
[9] M. Al Hasan, S. Salem, B. Pupacdi, and M. Zaki, "Clustering with Lower Bound on Similarity," in *Proc. 13th PAKDD*, 2009, p. 122.
[10] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society." *Nature*, vol. 435, no. 7043, pp. 814–8, June 2005.
[11] M. Zaki, M. Peters, I. Assent, and T. Seidl, "Clicks: An effective algorithm for mining subspace clusters in categorical dataset," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 51–70, January 2007.
[12] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *NIPS*, vol. 2, pp. 849–856, 2002.
[13] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *Proc. 15th SIGKDD*, 2009.
[14] A. Civril, M. Magdon-Ismail, and E. Bocek-Rivele, "SSDE: Fast graph drawing using sampled spectral distance embedding," in *Graph Drawing*, 2006, pp. 30–41.
[15] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.
[16] M. Magdon-Ismail and J. Purnell, "Approximating the covariance matrix with low rank perturbations," in *Proc. IDEAL"*, 2010.
[17] J. Baumes, M. Goldberg, M. Krishnamoorthy, M. Magdon-Ismail, and N. Preston, "Finding communities by clustering a graph into overlapping subgraphs," in *IADIS*, 2005, pp. 97–104.
[18] M. Ley, "DBLP: Computer science bibliography," 1993.
[19] C. van Rijsbergen and S. Robertson, "New models in probabilistic information retreival," *London British Library*, 1980.
[20] J. Purnell, "LRGMM code," *www.cs.rpi.edu/~purnej/code.php*, 2010.